Сутки профиля

На одной странице

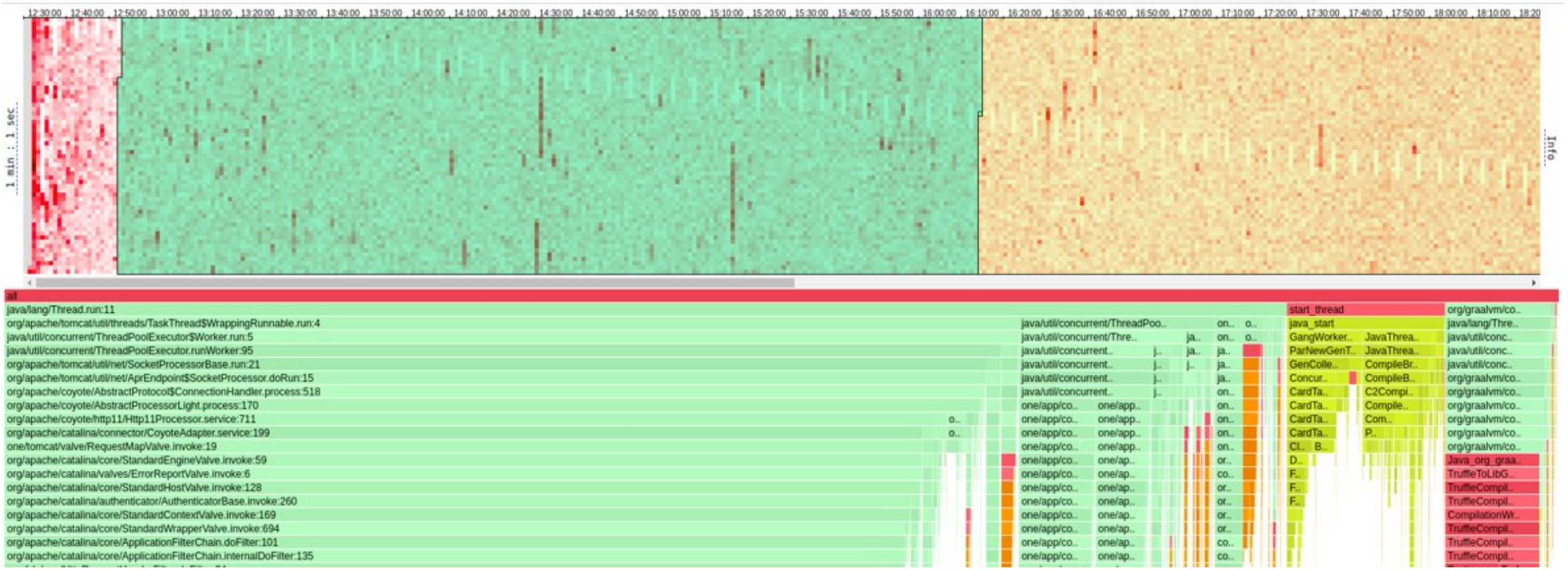
Артём Дроздов

Одноклассники

- 12K серверов в 7 ЦОД
- 50K контейнеров в one-cloud*
- >1 EB данных
- 4 Tbps исходящего трафика
- до 100K rps на контейнер

* https://www.youtube.com/watch?v=eJShMFzDV6g
One-cloud: ОС уровня дата-центра в Одноклассниках

Профилируем 24/7



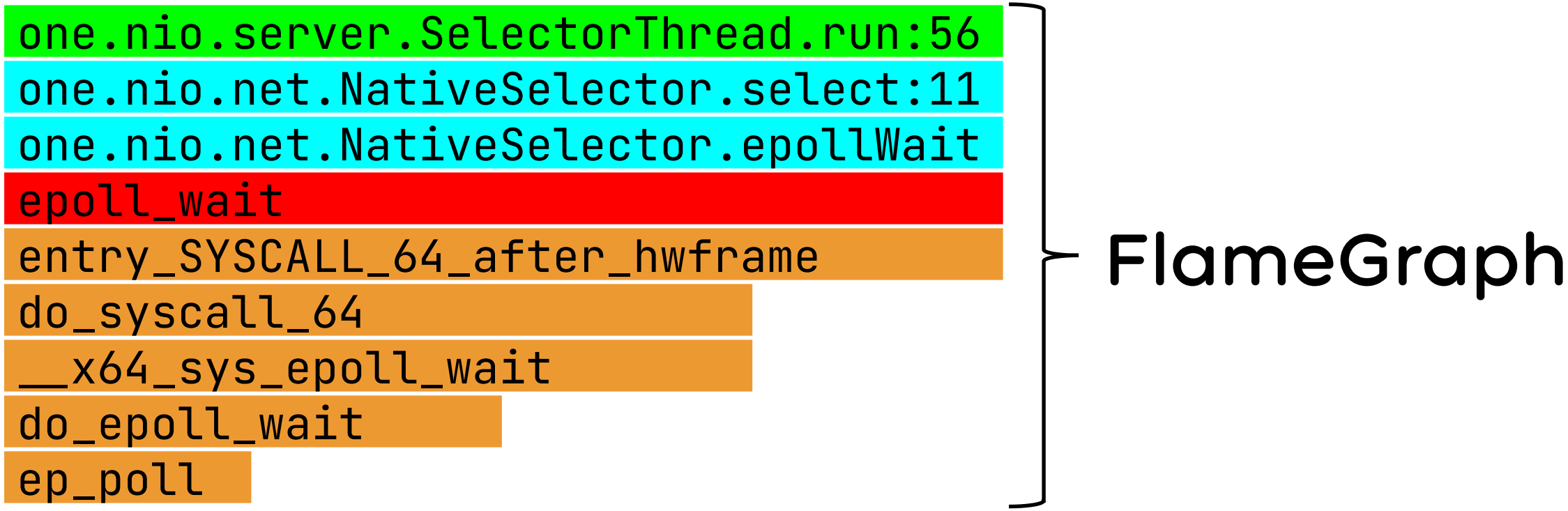
* https://www.youtube.com/watch?v=EcmR2BOeVZE

Непрерывное профилирование в облаке с помощью eBPF

Что ты такое?

Введение

Введение


one.nio.server.SelectorThread.run:56
one.nio.net.NativeSelector.select:11
one.nio.net.NativeSelector.epollWait
epoll_wait
entry_SYSCALL_64_after_hwframe
do_syscall_64
__x64_sys_epoll_wait
do_epoll_wait
ep_poll
FlameGraph

Введение

```
one.nio.server.SelectorThread.run:56
one.nio.net.NativeSelector.select:11
one.nio.net.NativeSelector.epollWait
epoll_wait
entry_SYSCALL_64_after_hwframe
do_syscall_64
__x64_sys_epoll_wait
do_epoll_wait
ep_poll
```

Frame

Class: ...SelectorThread
Method: run
Line: 56
Type: JIT-compiled

M1

Введение

```
one.nio.server.SelectorThread.run:56
one.nio.net.NativeSelector.select:11
one.nio.net.NativeSelector.epollWait
epoll_wait
entry_SYSCALL_64_after_hwframe
do_syscall_64
__x64_sys_epoll_wait
do_epoll_wait
ep_poll
```

Frame

Class: ...NativeSelector
Method: select
Line: 11
Type: Inlined

M2

Введение

```
one.nio.server.SelectorThread.run:56
one.nio.net.NativeSelector.select:11
one.nio.net.NativeSelector.epollWait
epoll_wait
entry_SYSCALL_64_after_hwframe
do_syscall_64
__x64_sys_epoll_wait
do_epoll_wait
ep_poll
```

Class: ...NativeSelector
Method: epollWait
~~Line: ???~~
Type: Inlined

M3

Введение

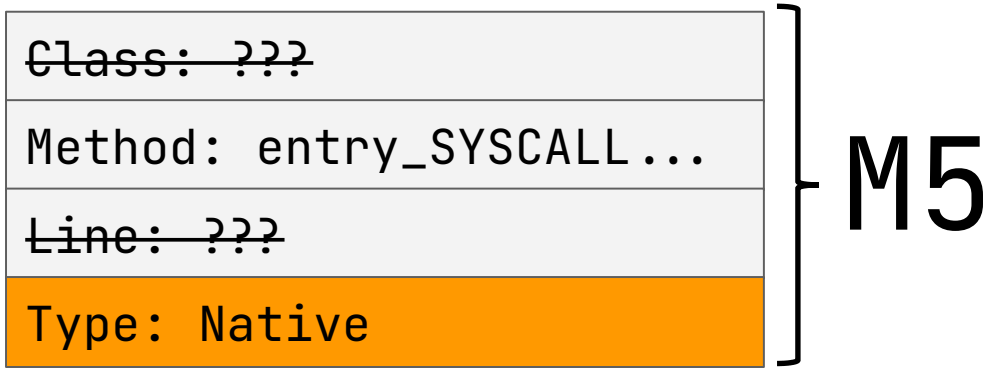
```
one.nio.server.SelectorThread.run:56
one.nio.net.NativeSelector.select:11
one.nio.net.NativeSelector.epollWait
epoll_wait
entry_SYSCALL_64_after_hwframe
do_syscall_64
__x64_sys_epoll_wait
do_epoll_wait
ep_poll
```

~~Class: ???~~
Method: epoll_wait
~~Line: ???~~
Type: Native

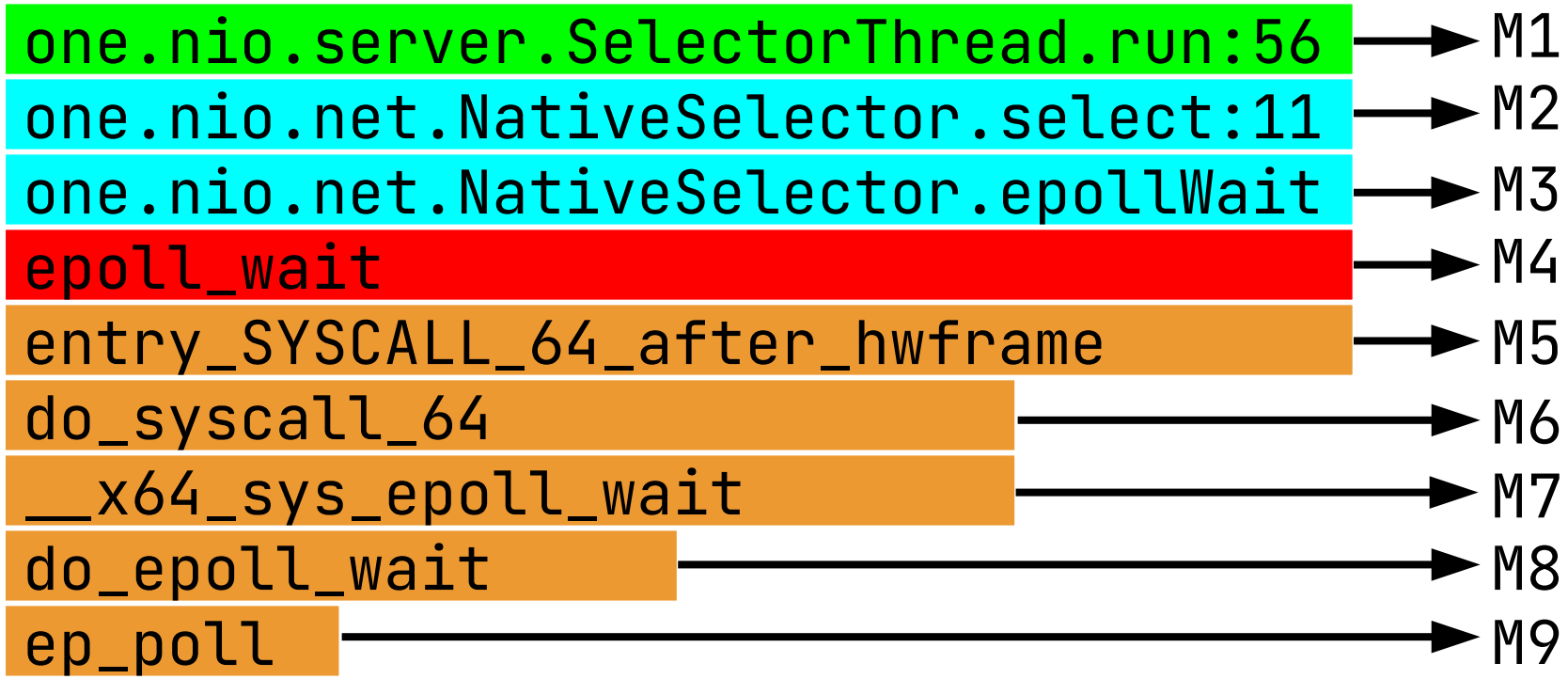
M4

Введение

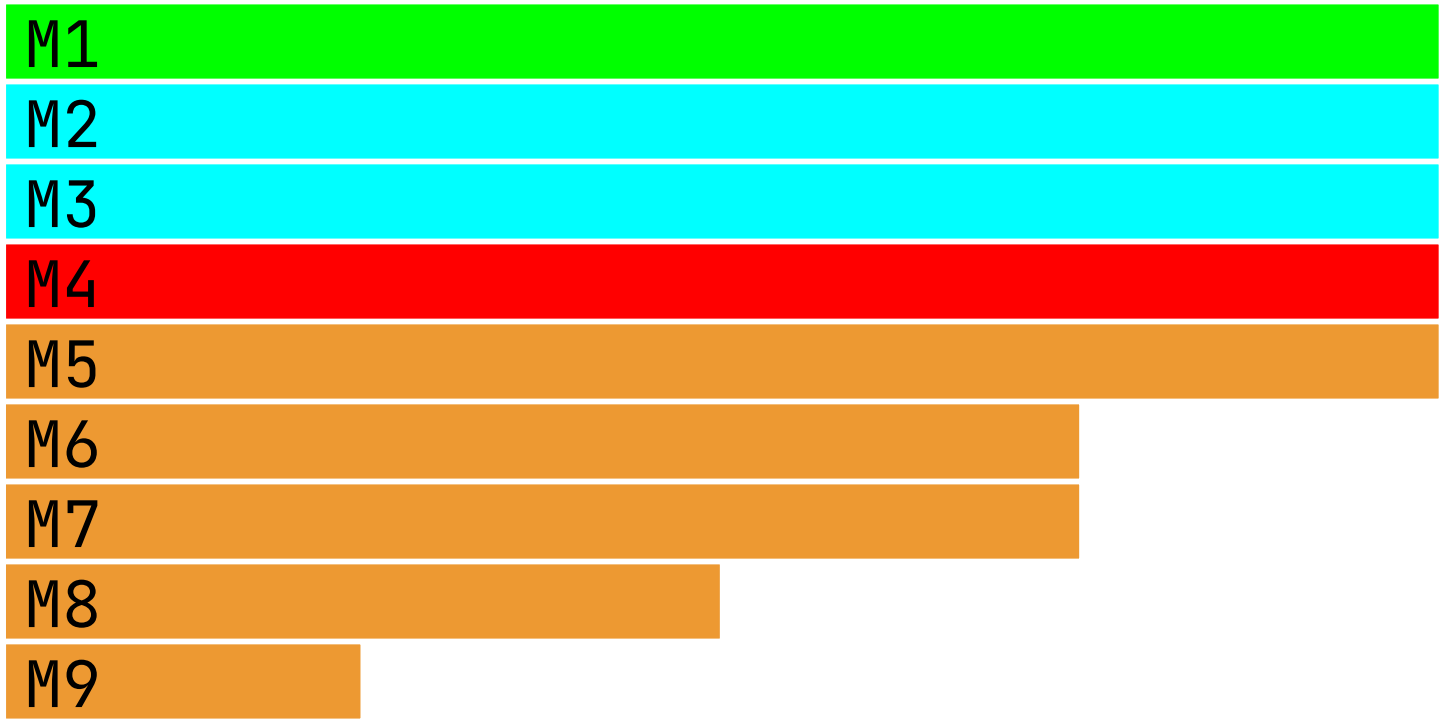
```
one.nio.server.SelectorThread.run:56
one.nio.net.NativeSelector.select:11
one.nio.net.NativeSelector.epollWait
epoll_wait
entry_SYSCALL_64_after_hwframe
do_syscall_64
__x64_sys_epoll_wait
do_epoll_wait
ep_poll
```



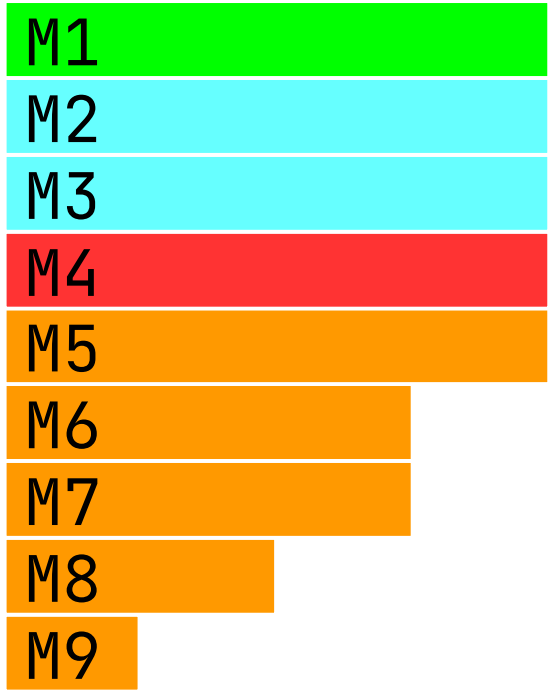
Введение


one.nio.server.SelectorThread.run:56
one.nio.net.NativeSelector.select:11
one.nio.net.NativeSelector.epollWait
epoll_wait
entry_SYSCALL_64_after_hwframe
do_syscall_64
__x64_sys_epoll_wait
do_epoll_wait
ep_poll
M1
M2
M3
M4
M5
M6
M7
M8
M9

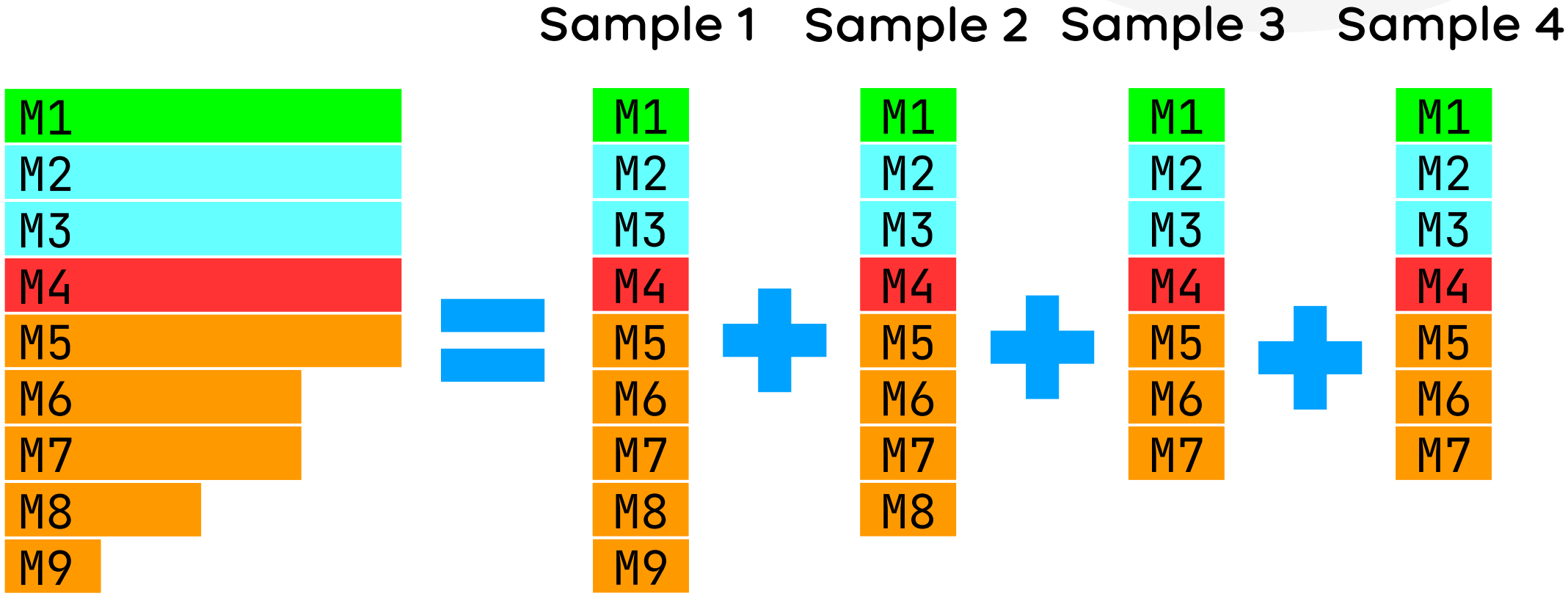
Введение



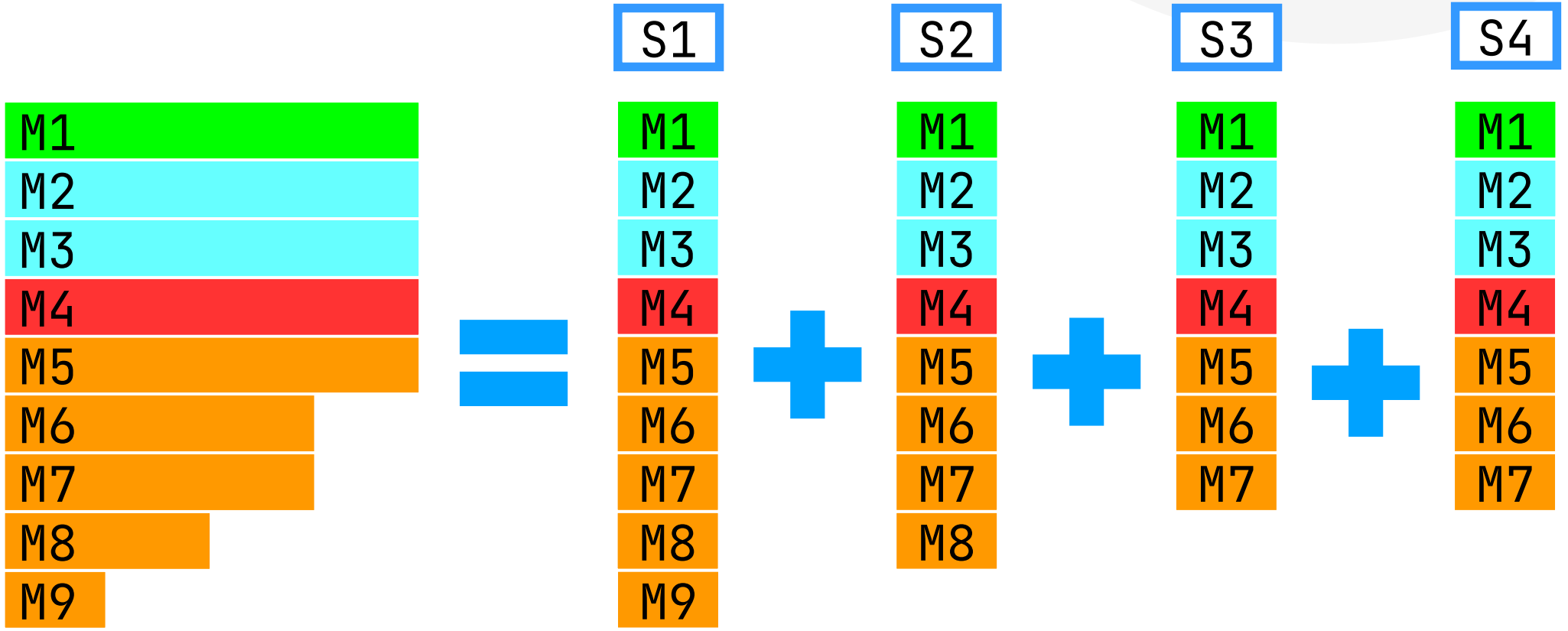
Введение



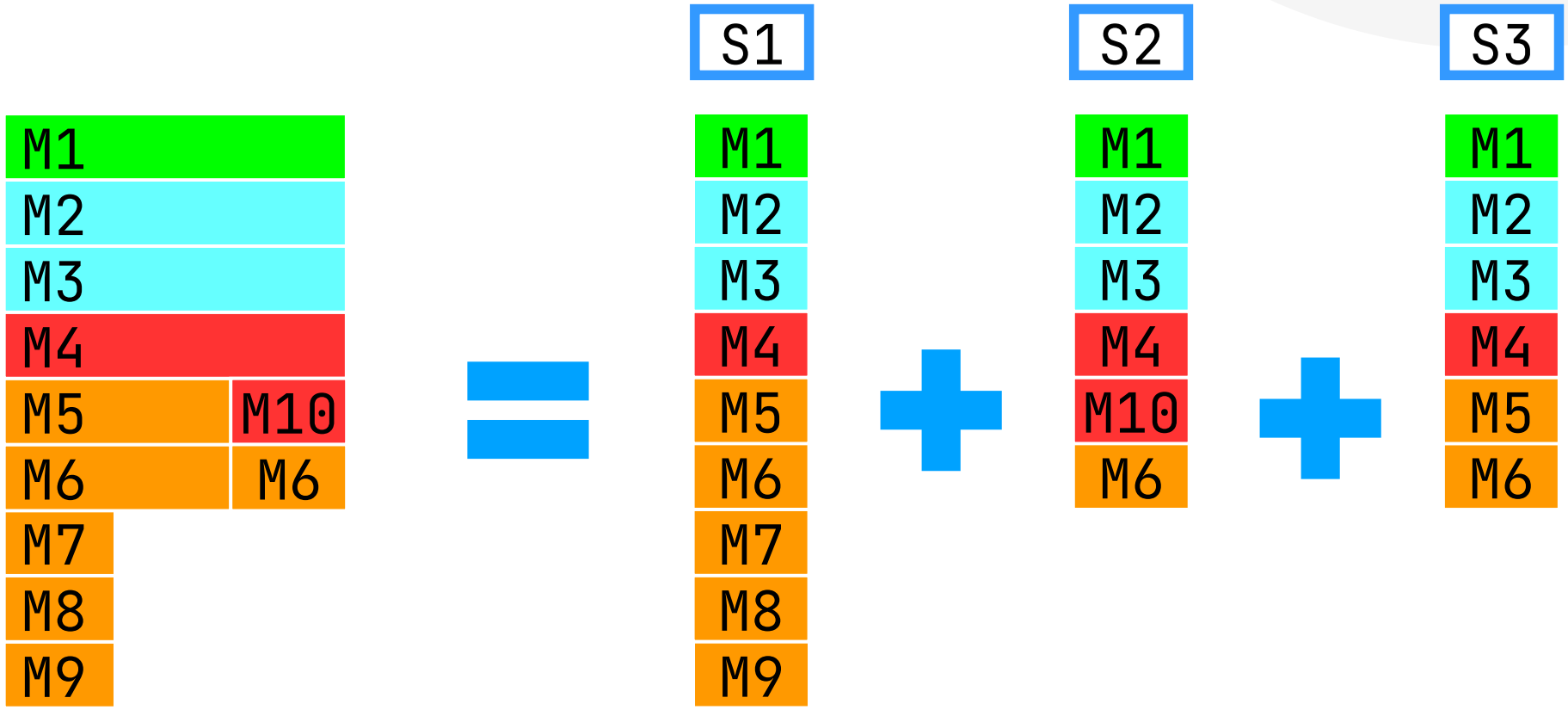
Введение


Sample 1
Sample 2
Sample 3
Sample 4
M1
M2
M3
M4
M5
M6
M7
M8
M9
=
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
M1
M2
M3
M4
M5
M6
M7
M8
+
M1
M2
M3
M4
M5
M6
M7
+
M1
M2
M3
M4
M5
M6
M7

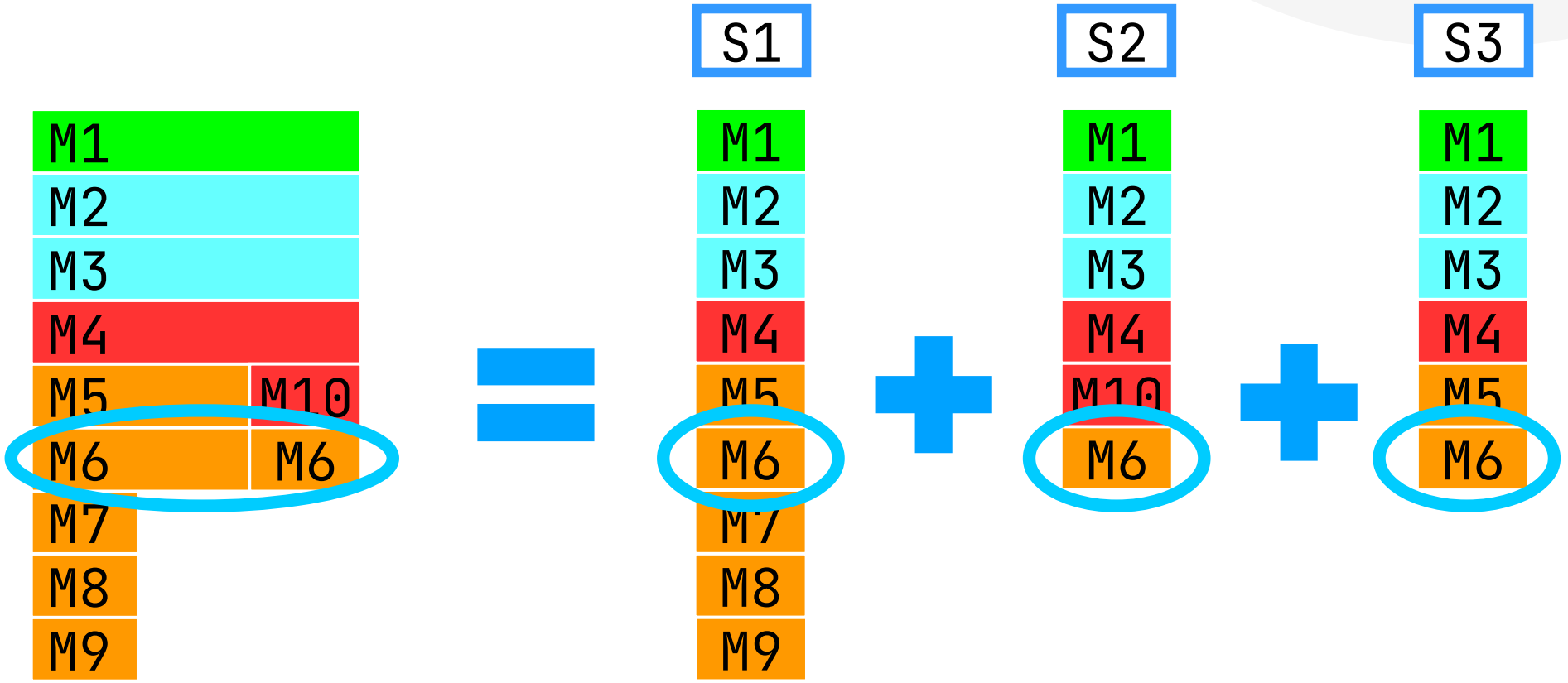
Введение


M1
M2
M3
M4
M5
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
S2
M1
M2
M3
M4
M5
M6
M7
M8
+
S3
M1
M2
M3
M4
M5
M6
M7
+
S4
M1
M2
M3
M4
M5
M6
M7

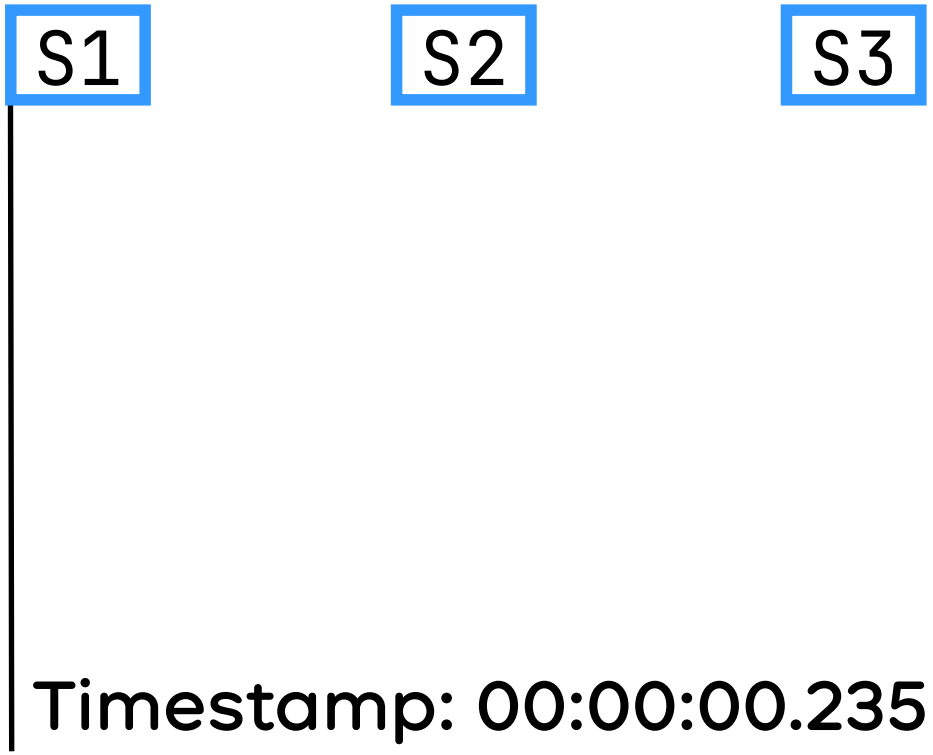
Введение


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
S2
M1
M2
M3
M4
M10
M6
+
S3
M1
M2
M3
M4
M5
M6

Введение

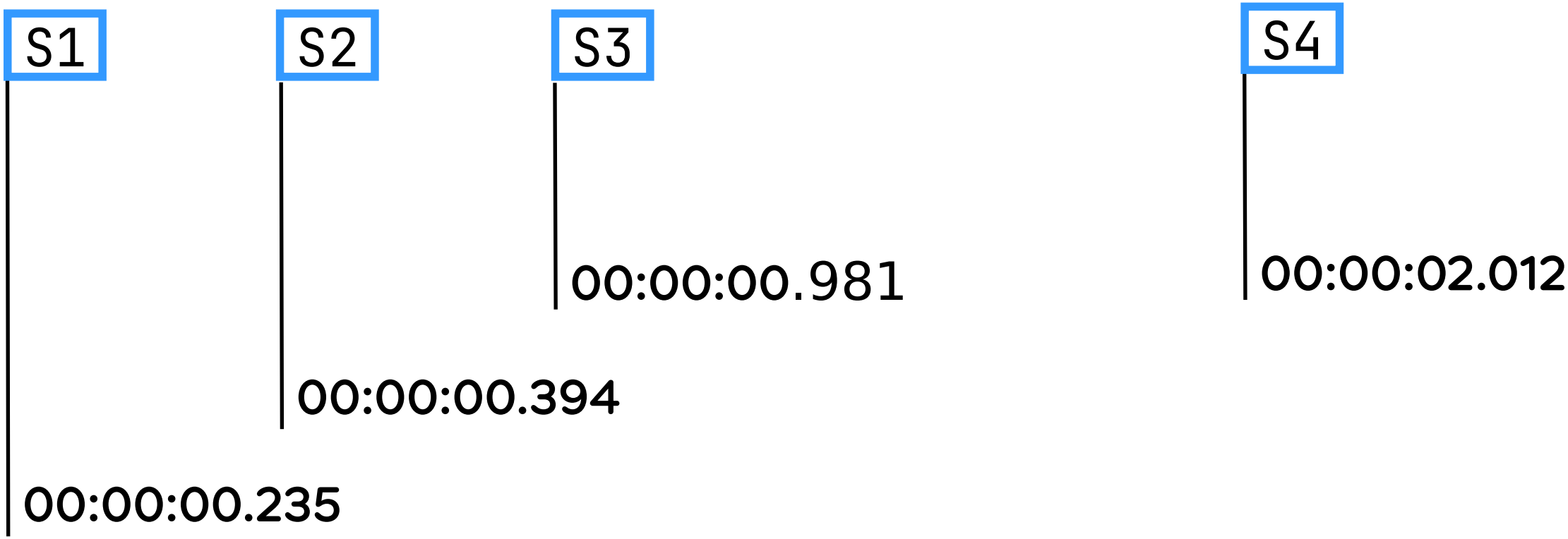

M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
S2
M1
M2
M3
M4
M10
M6
+
S3
M1
M2
M3
M4
M5
M6

Введение

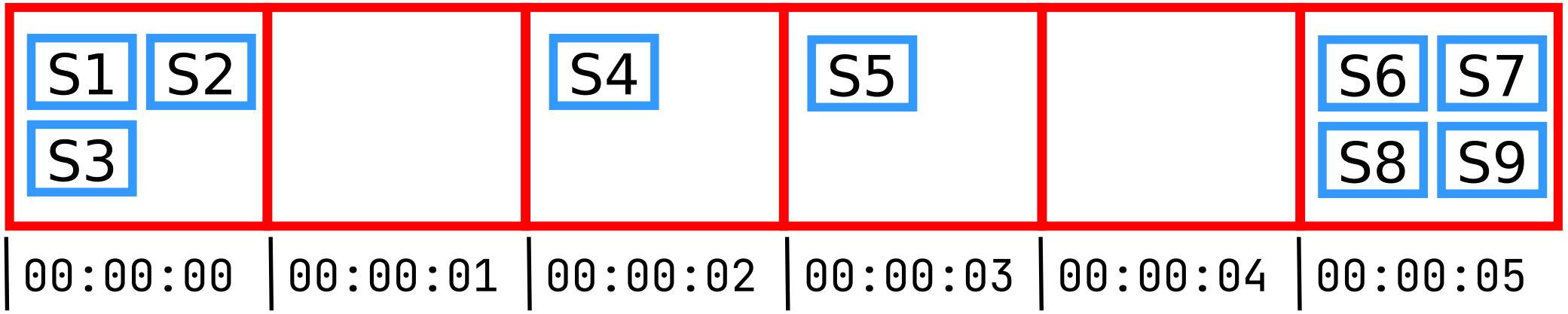


Введение

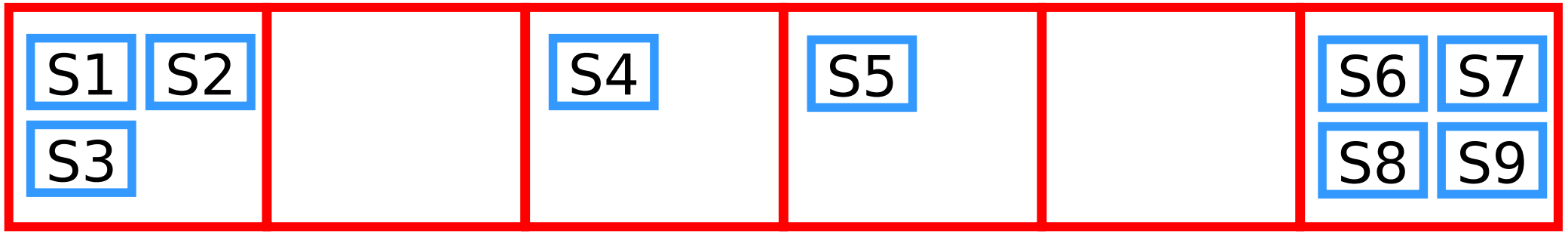
Введение

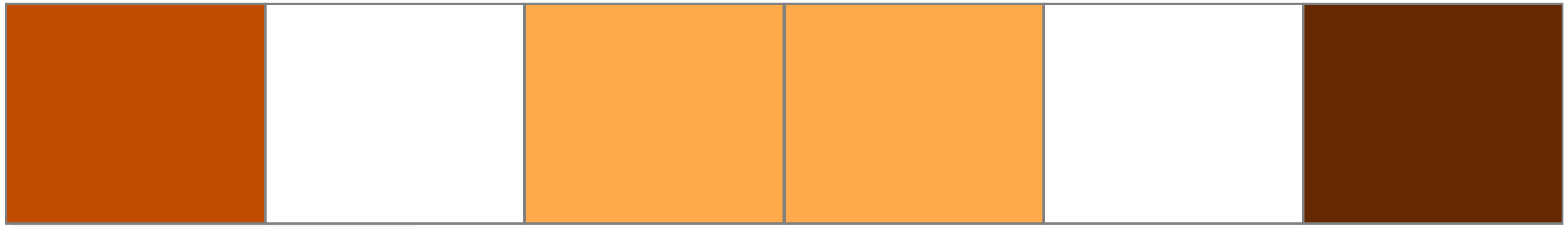


Введение

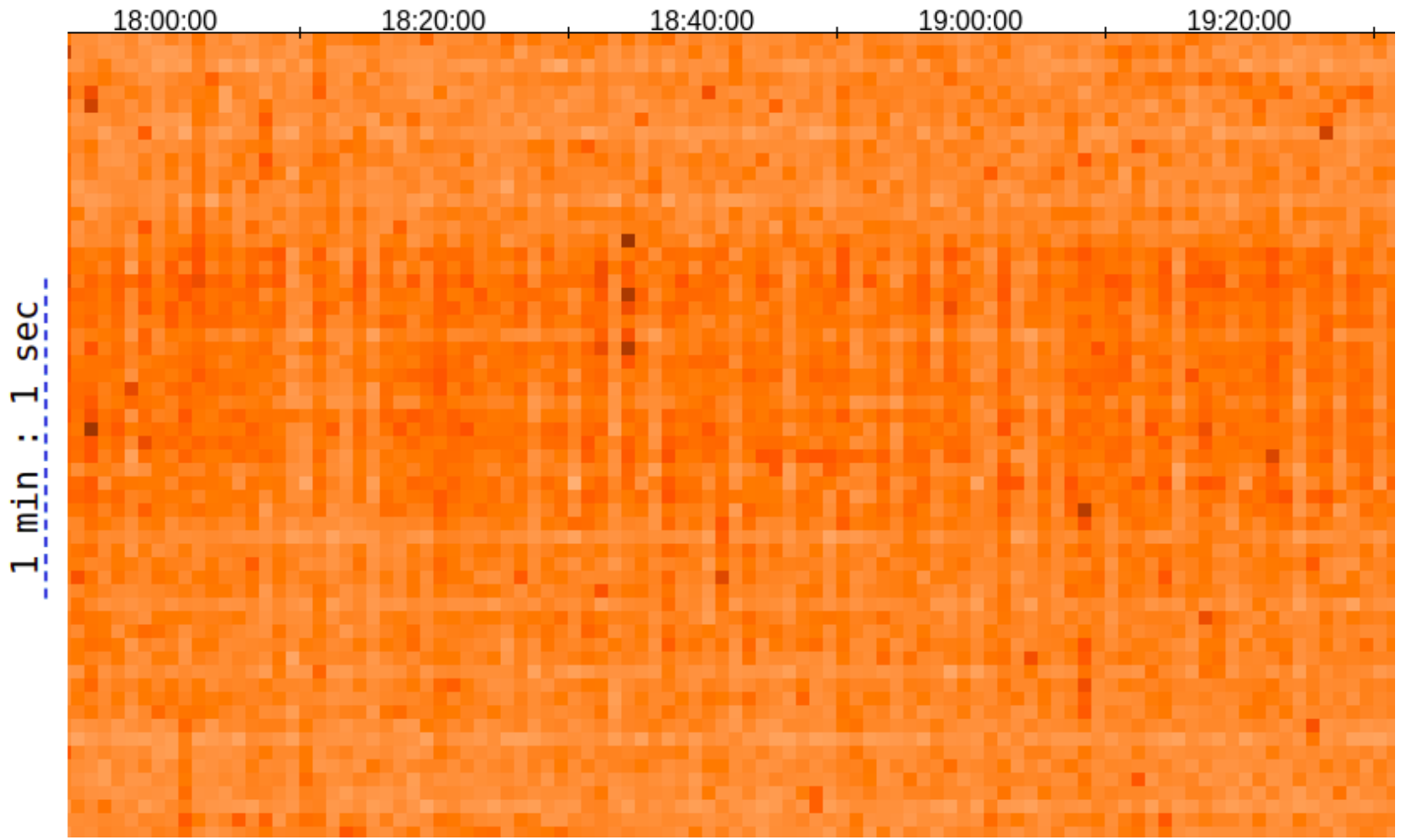

S1
S2
S3
S4
S5
S6
S7
S8
S9
00:00:00
00:00:01
00:00:02
00:00:03
00:00:04
00:00:05

Введение

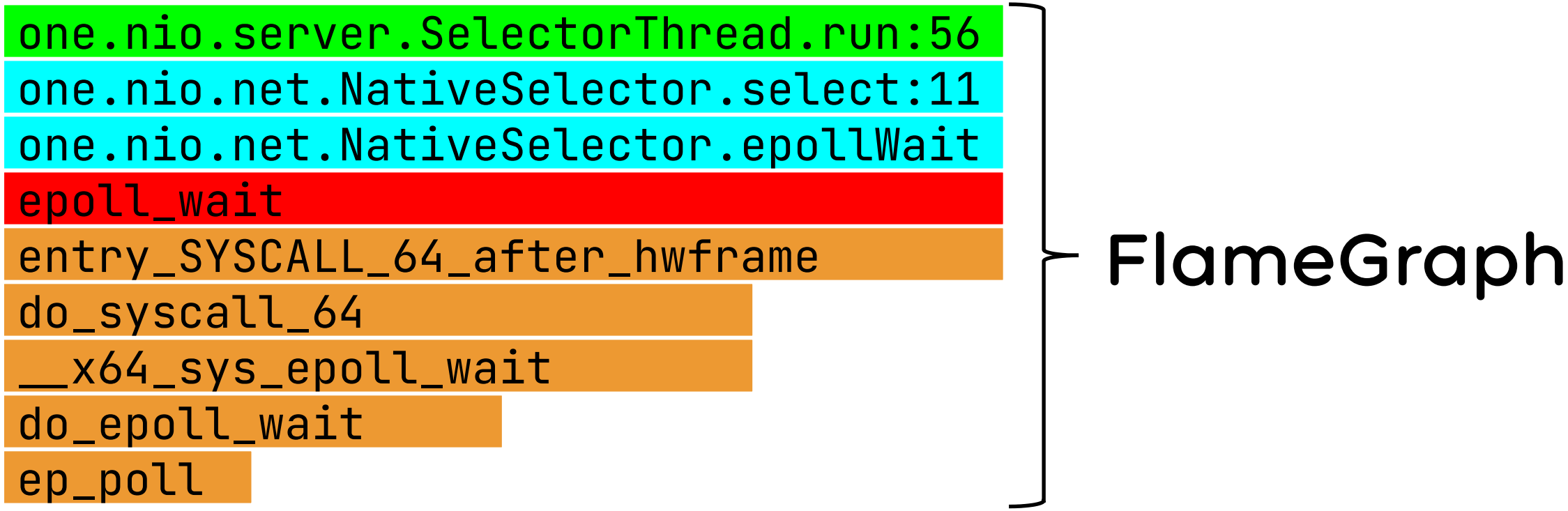




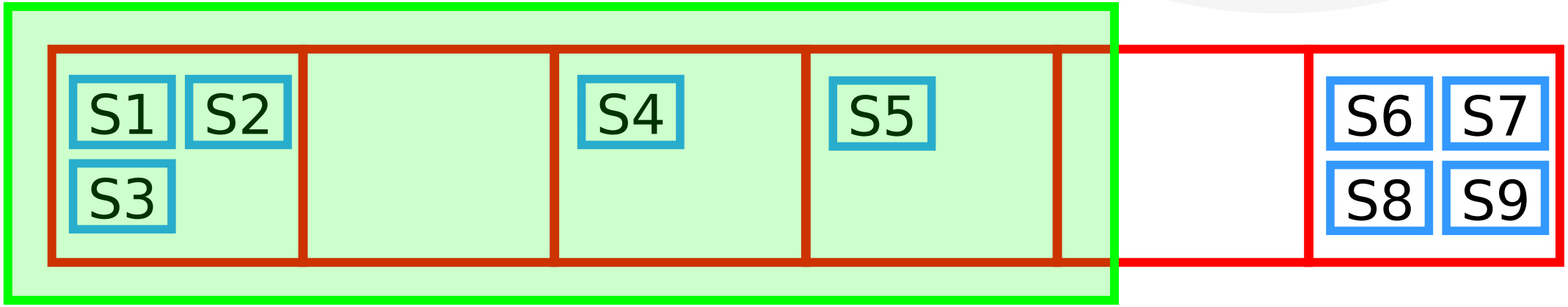
Введение



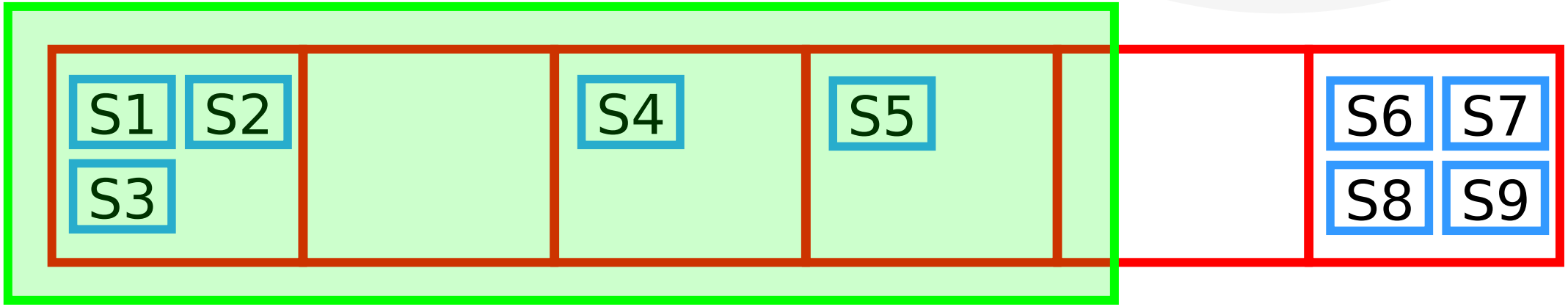
Введение


one.nio.server.SelectorThread.run:56
one.nio.net.NativeSelector.select:11
one.nio.net.NativeSelector.epollWait
epoll_wait
entry_SYSCALL_64_after_hwframe
do_syscall_64
__x64_sys_epoll_wait
do_epoll_wait
ep_poll
FlameGraph

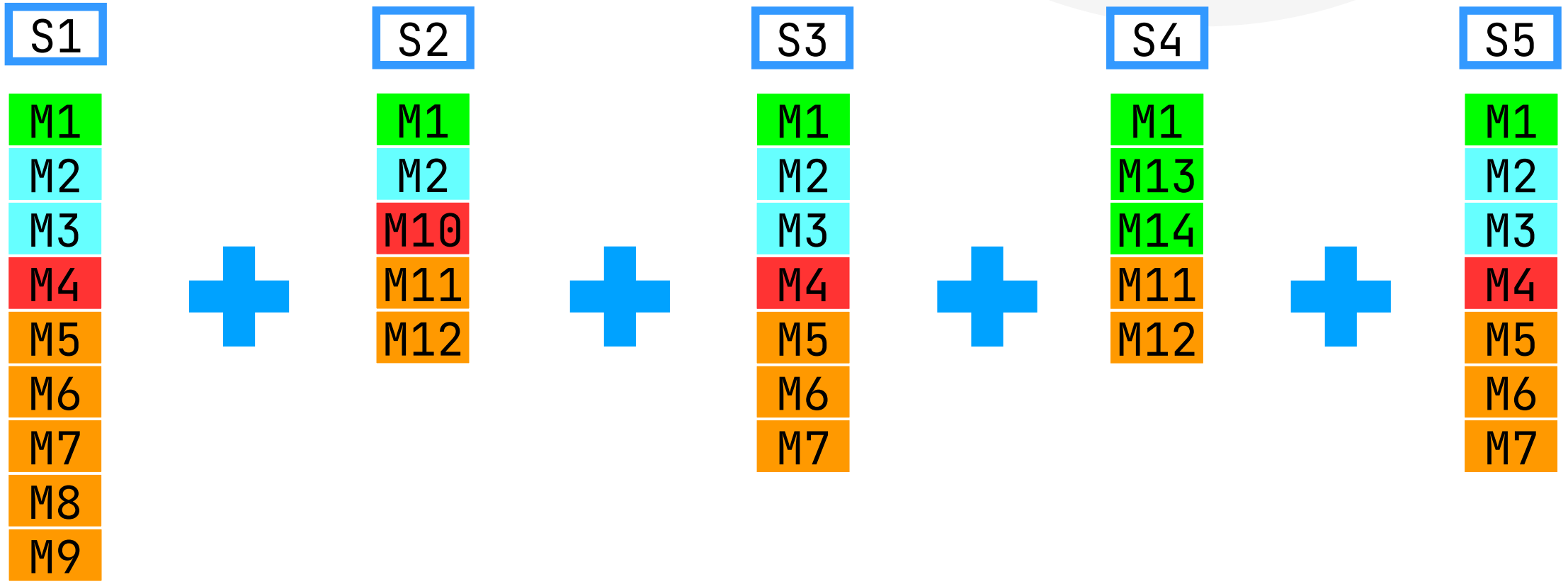
Введение



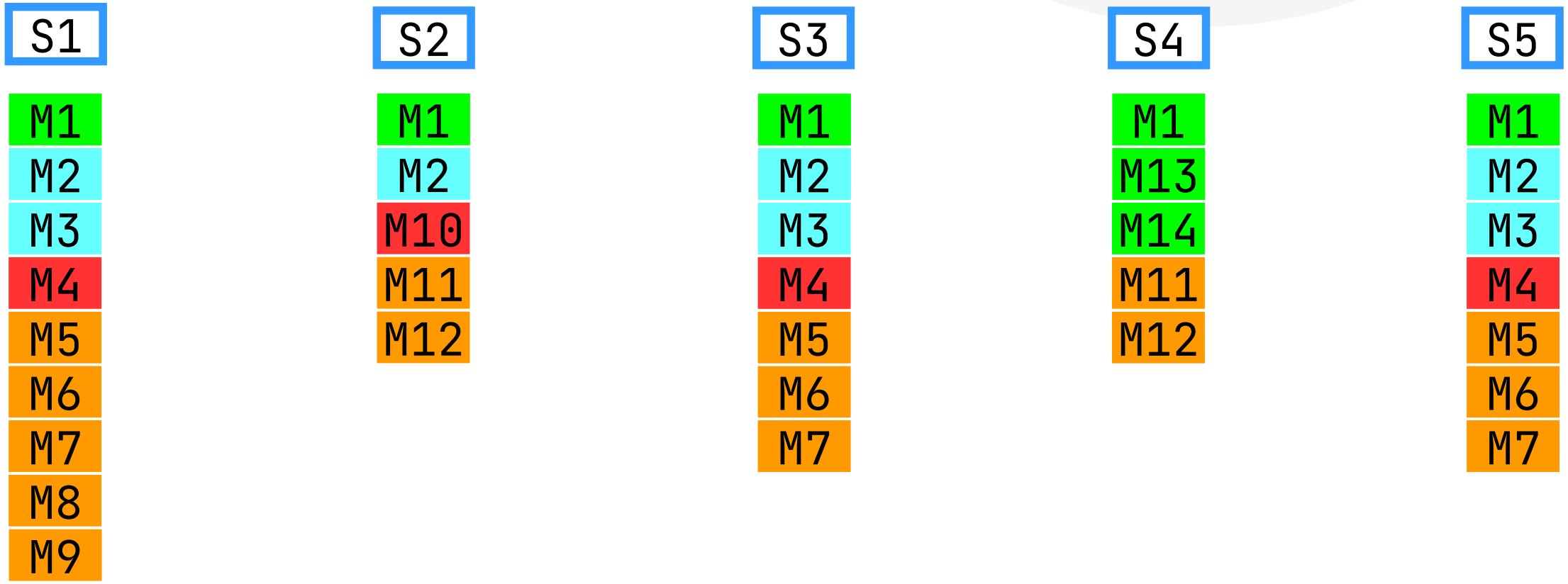
Введение



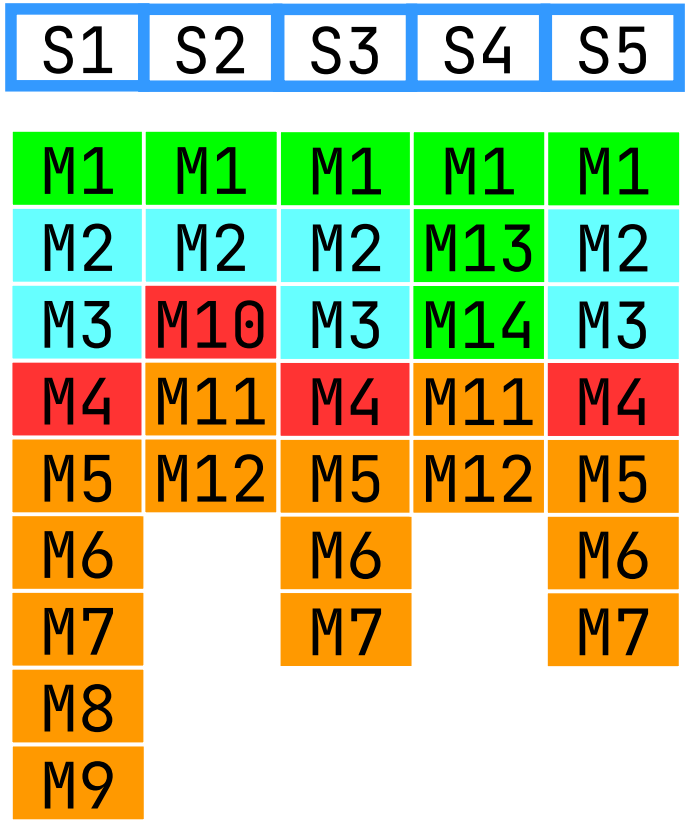
Введение


S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M1
M2
M10
M11
M12
S3
M1
M2
M3
M4
M5
M6
M7
S4
M1
M13
M14
M11
M12
S5
M1
M2
M3
M4
M5
M6
M7

Введение

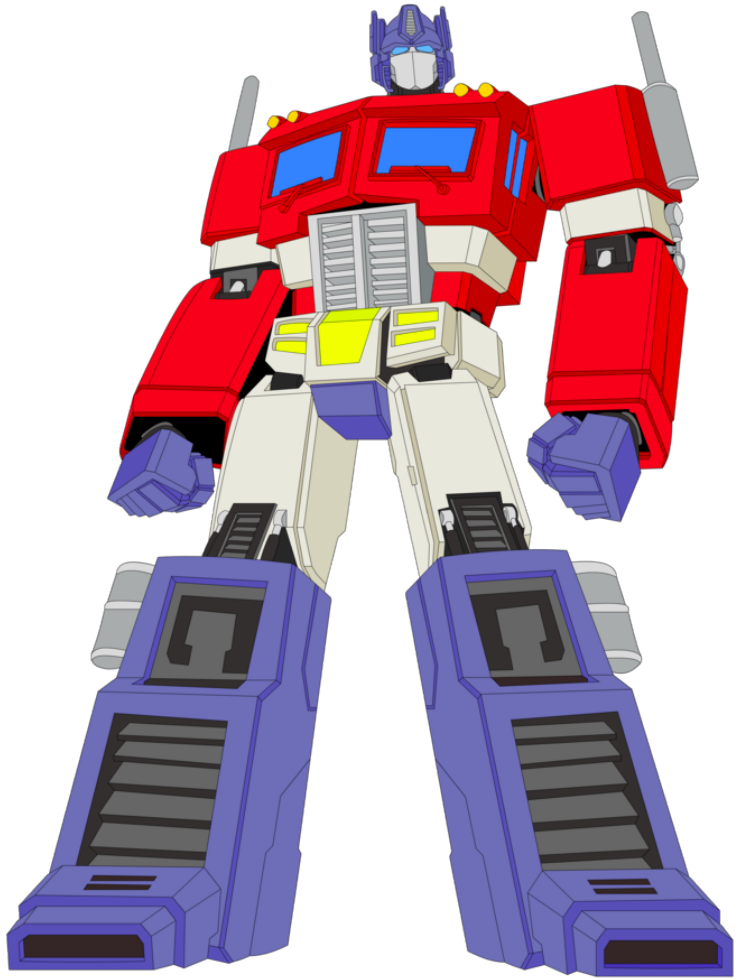

S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M1
M2
M10
M11
M12
S3
M1
M2
M3
M4
M5
M6
M7
S4
M1
M13
M14
M11
M12
S5
M1
M2
M3
M4
M5
M6
M7

Введение

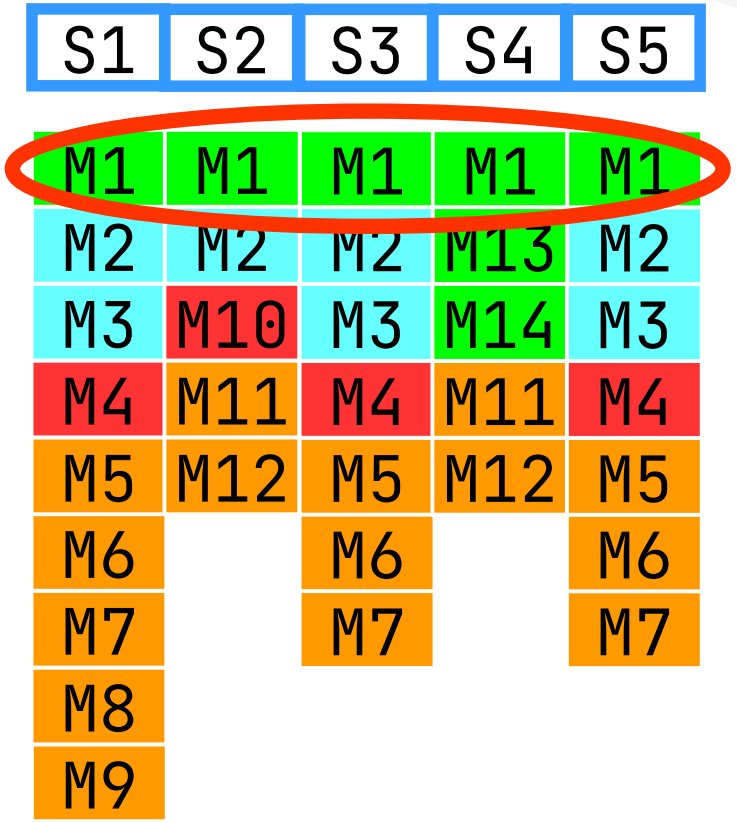

S1
S2
S3
S4
S5
M1
M1
M1
M1
M1
M2
M2
M2
M13
M2
M3
M10
M3
M14
M3
M4
M11
M4
M11
M4
M5
M12
M5
M12
M5
M6
M6
M6
M7
M7
M7
M8
M9

Введение

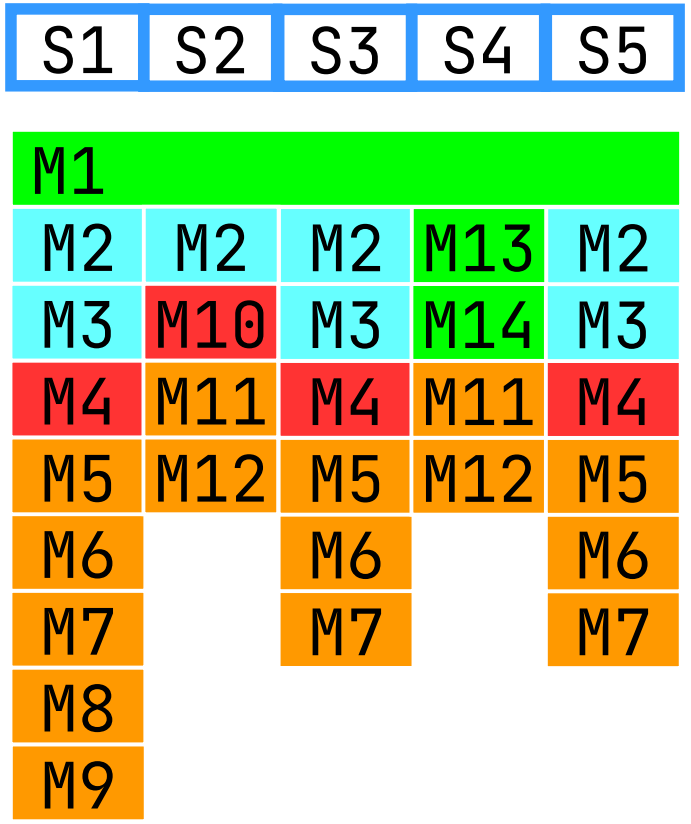
S1	S2	S3	S4	S5
M1	M1	M1	M1	M1
M2	M2	M2	M13	M2
M3	M10	M3	M14	M3
M4	M11	M4	M11	M4
M5	M12	M5	M12	M5
M6		M6		M6
M7		M7		M7
M8				
M9				



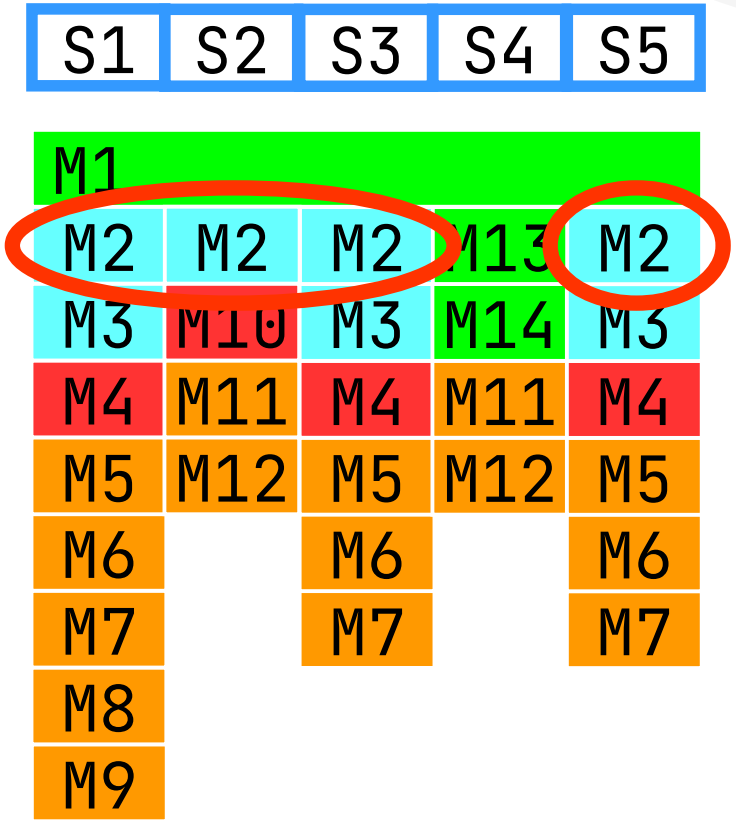
Введение


S1
S2
S3
S4
S5
M1
M1
M1
M1
M1
M2
M2
M2
M13
M2
M3
M10
M3
M14
M3
M4
M11
M4
M11
M4
M5
M12
M5
M12
M5
M6
M6
M6
M7
M7
M7
M8
M9

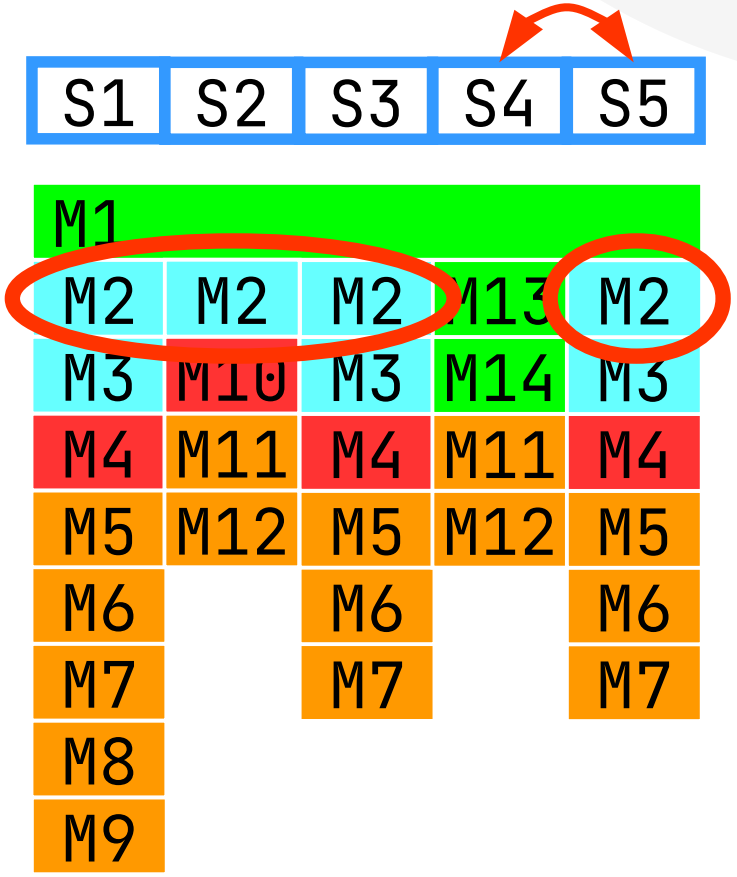
Введение


S1
S2
S3
S4
S5
M1
M2
M2
M2
M13
M2
M3
M10
M3
M14
M3
M4
M11
M4
M11
M4
M5
M12
M5
M12
M5
M6
M6
M6
M7
M7
M7
M8
M9

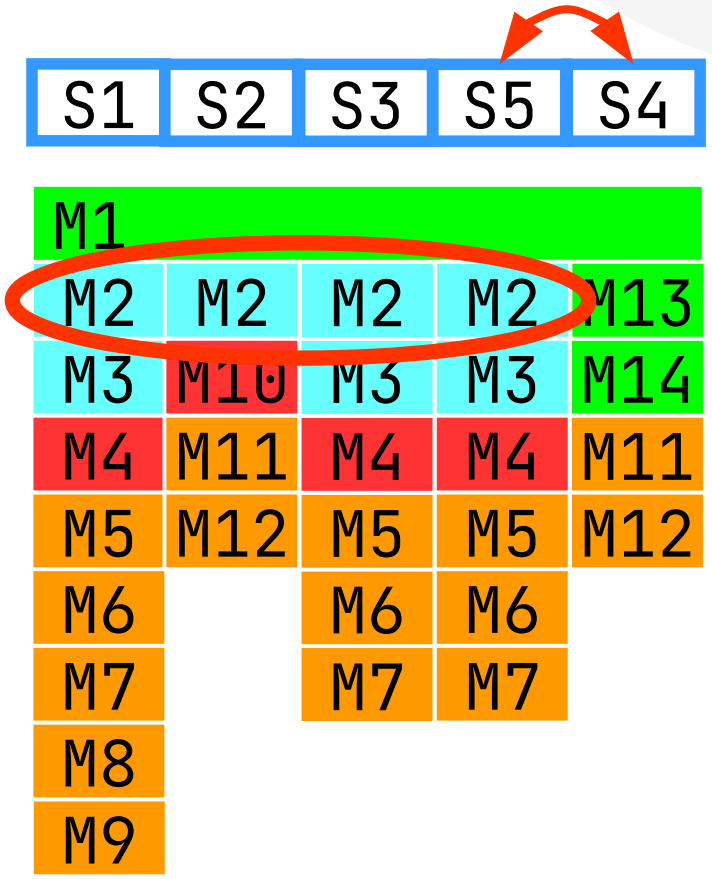
Введение


S1
S2
S3
S4
S5
M1
M2
M2
M2
M13
M2
M3
M10
M3
M14
M3
M4
M11
M4
M11
M4
M5
M12
M5
M12
M5
M6
M6
M6
M7
M7
M7
M8
M9

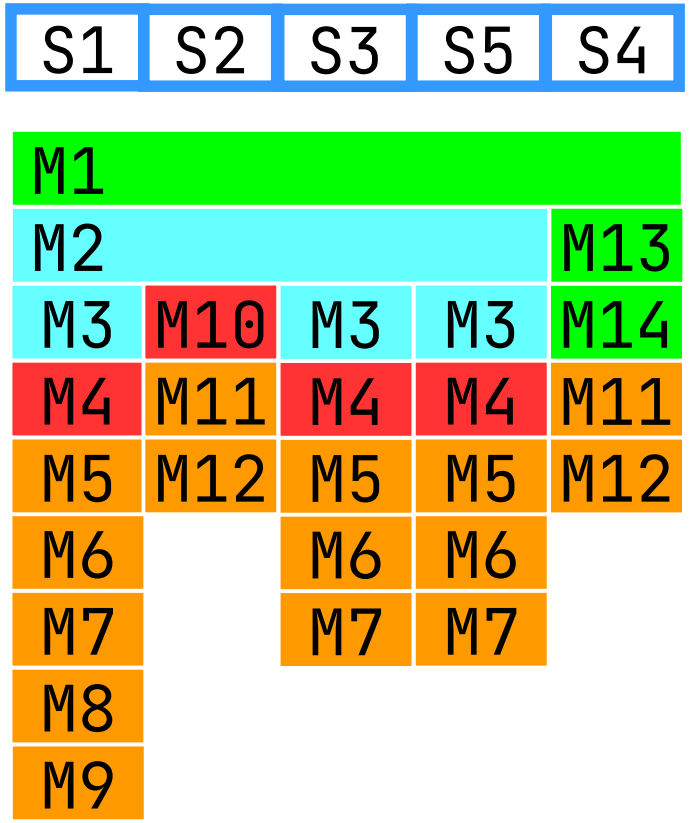
Введение


S1
S2
S3
S4
S5
M1
M2
M2
M2
M13
M2
M3
M10
M3
M14
M3
M4
M11
M4
M11
M4
M5
M12
M5
M12
M5
M6
M6
M6
M7
M7
M7
M8
M9

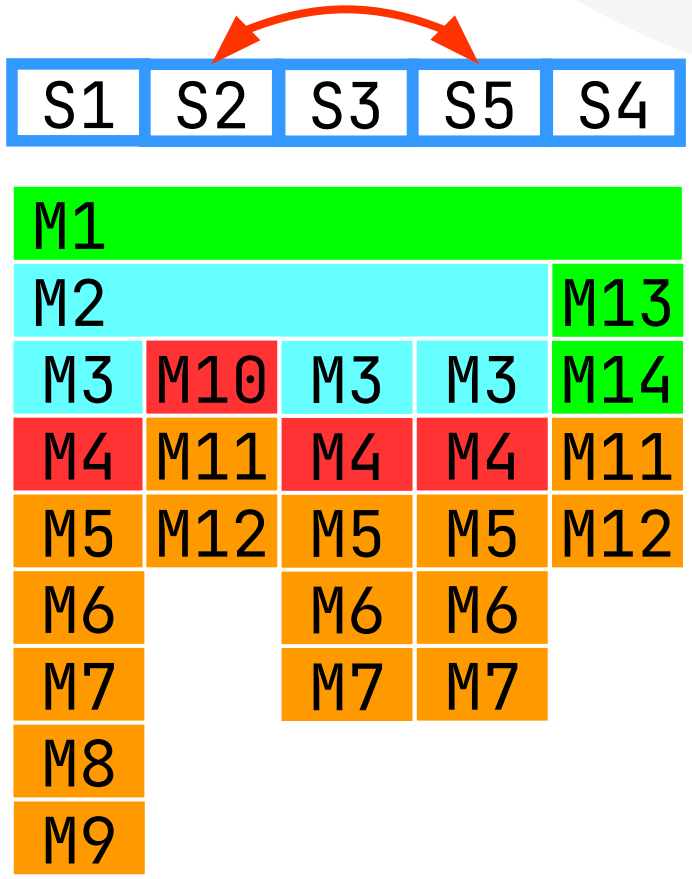
Введение


S1
S2
S3
S5
S4
M1
M2
M2
M2
M2
M13
M3
M10
M3
M3
M14
M4
M11
M4
M4
M11
M5
M12
M5
M5
M12
M6
M6
M6
M7
M7
M7
M8
M9

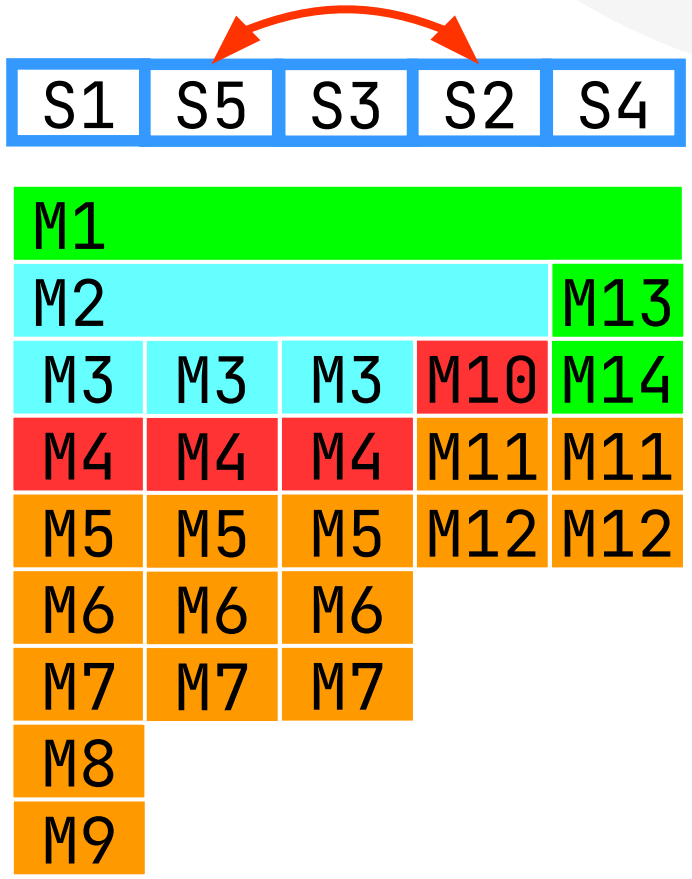
Введение


S1
S2
S3
S5
S4
M1
M2
M13
M3
M10
M3
M3
M14
M4
M11
M4
M4
M11
M5
M12
M5
M5
M12
M6
M6
M6
M7
M7
M7
M8
M9

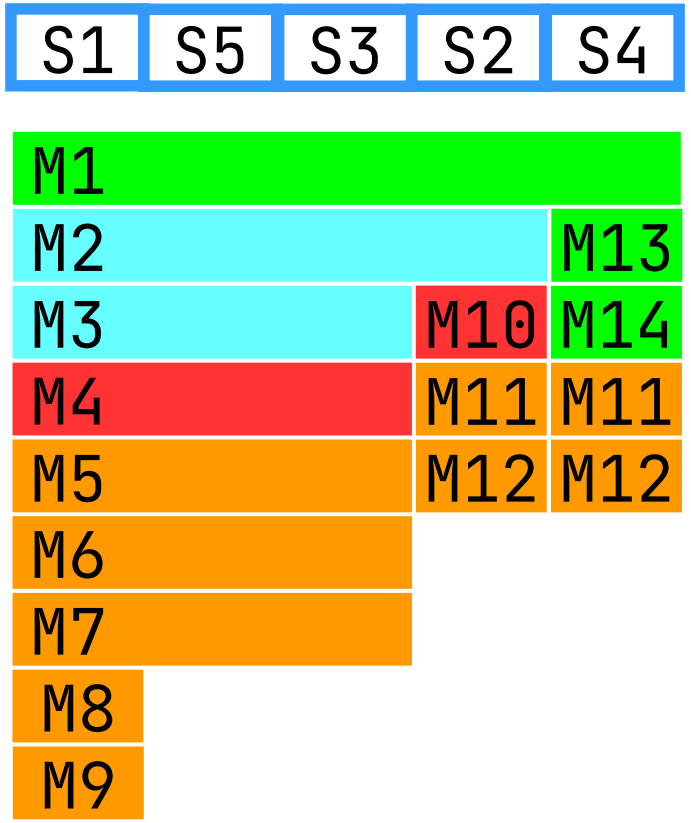
Введение


S1
S2
S3
S5
S4
M1
M2
M13
M3
M10
M3
M3
M14
M4
M11
M4
M4
M11
M5
M12
M5
M5
M12
M6
M6
M6
M7
M7
M7
M8
M9

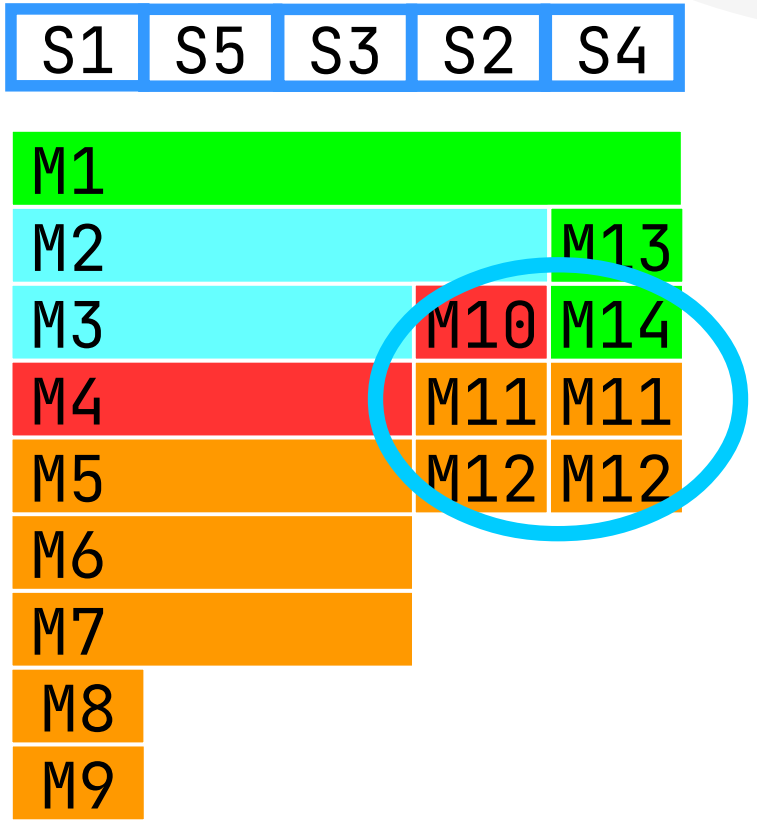
Введение


S1
S5
S3
S2
S4
M1
M2
M13
M3
M3
M3
M10
M14
M4
M4
M4
M11
M11
M5
M5
M5
M12
M12
M6
M6
M6
M7
M7
M7
M8
M9

Введение



Введение

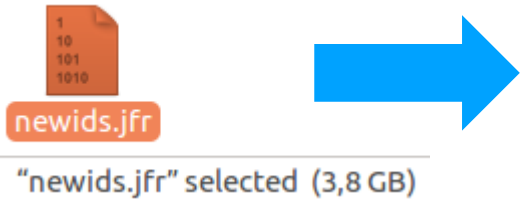


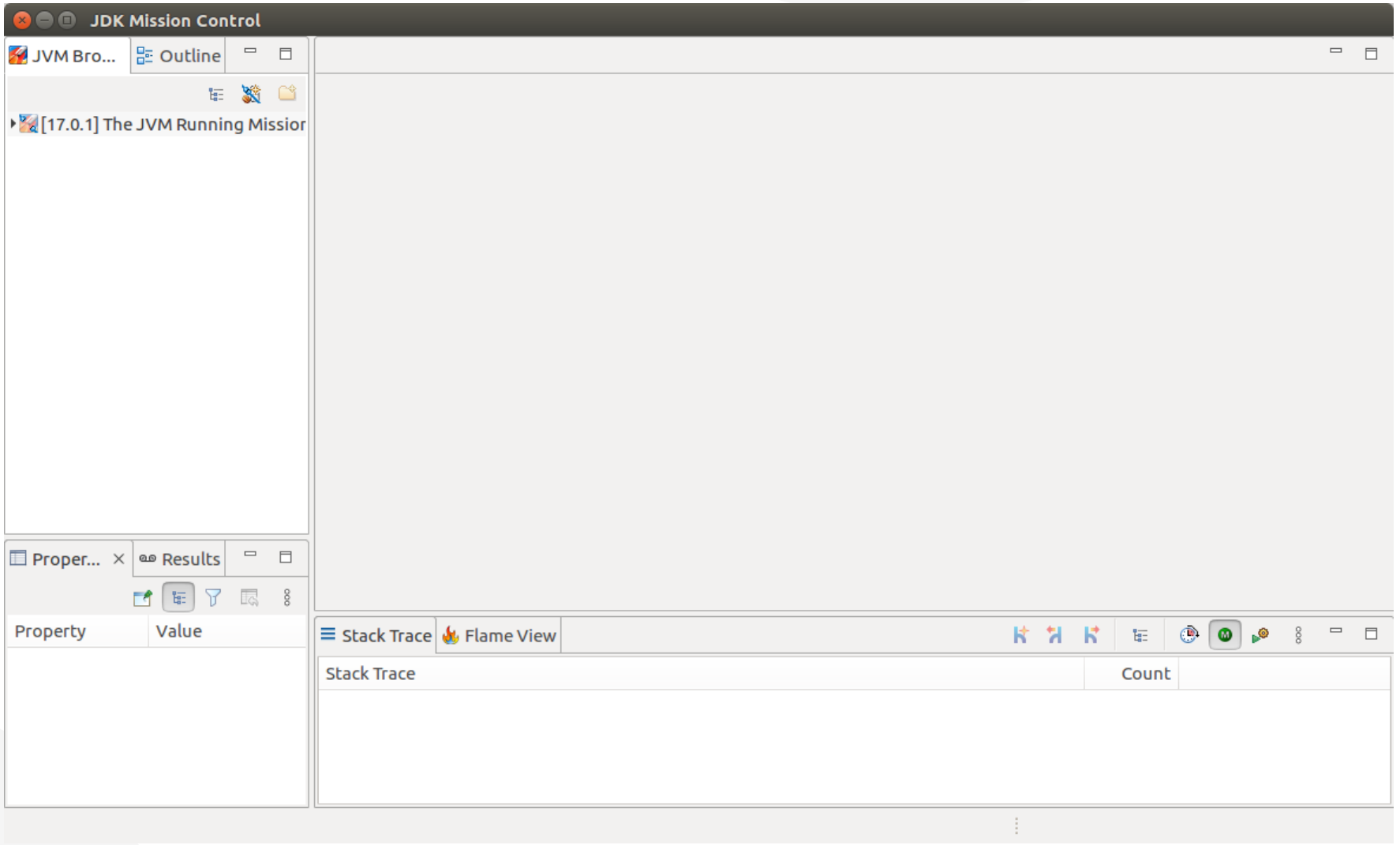
Сутки профиля

По версии JMC

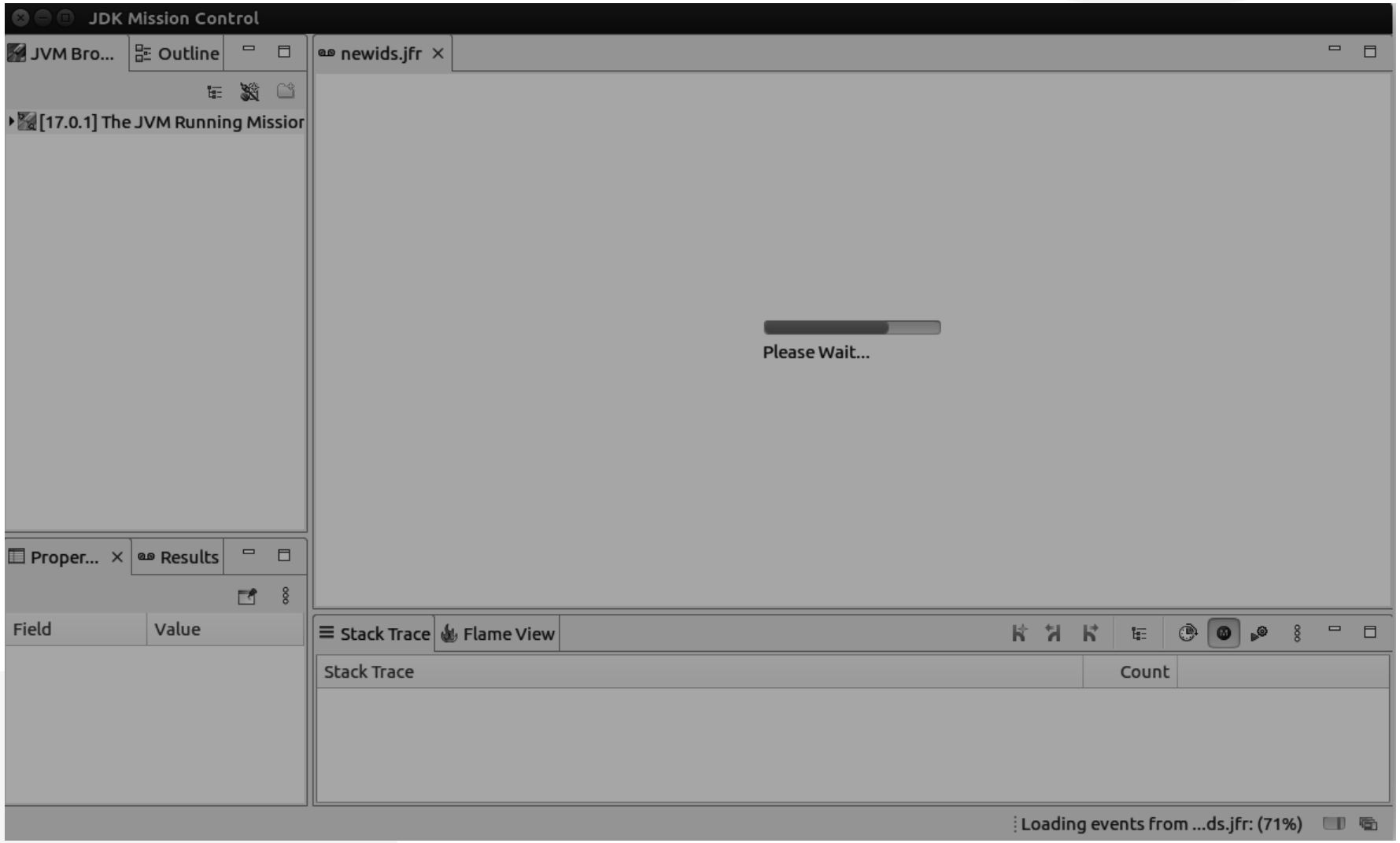
Mission Control

Mission Control

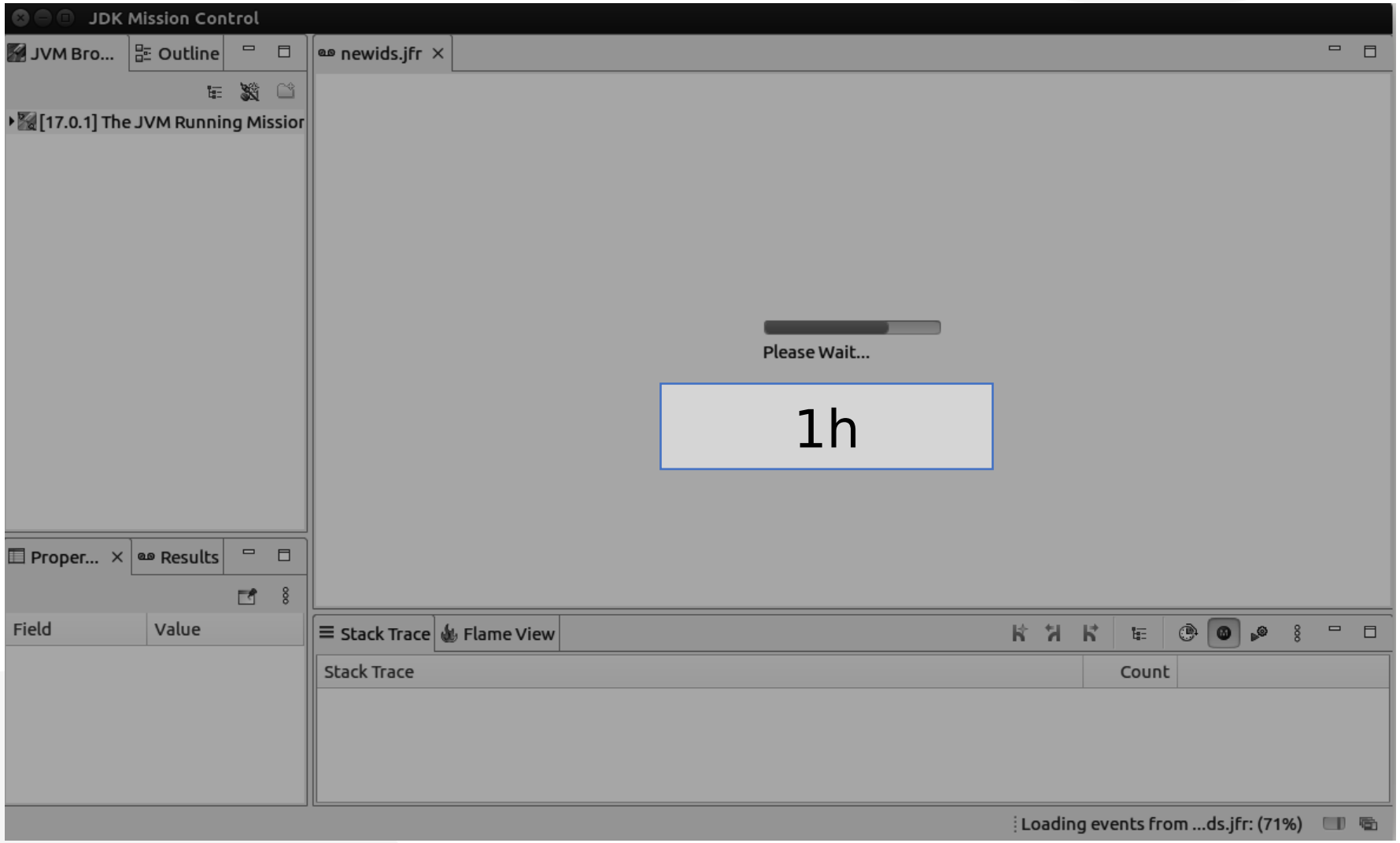




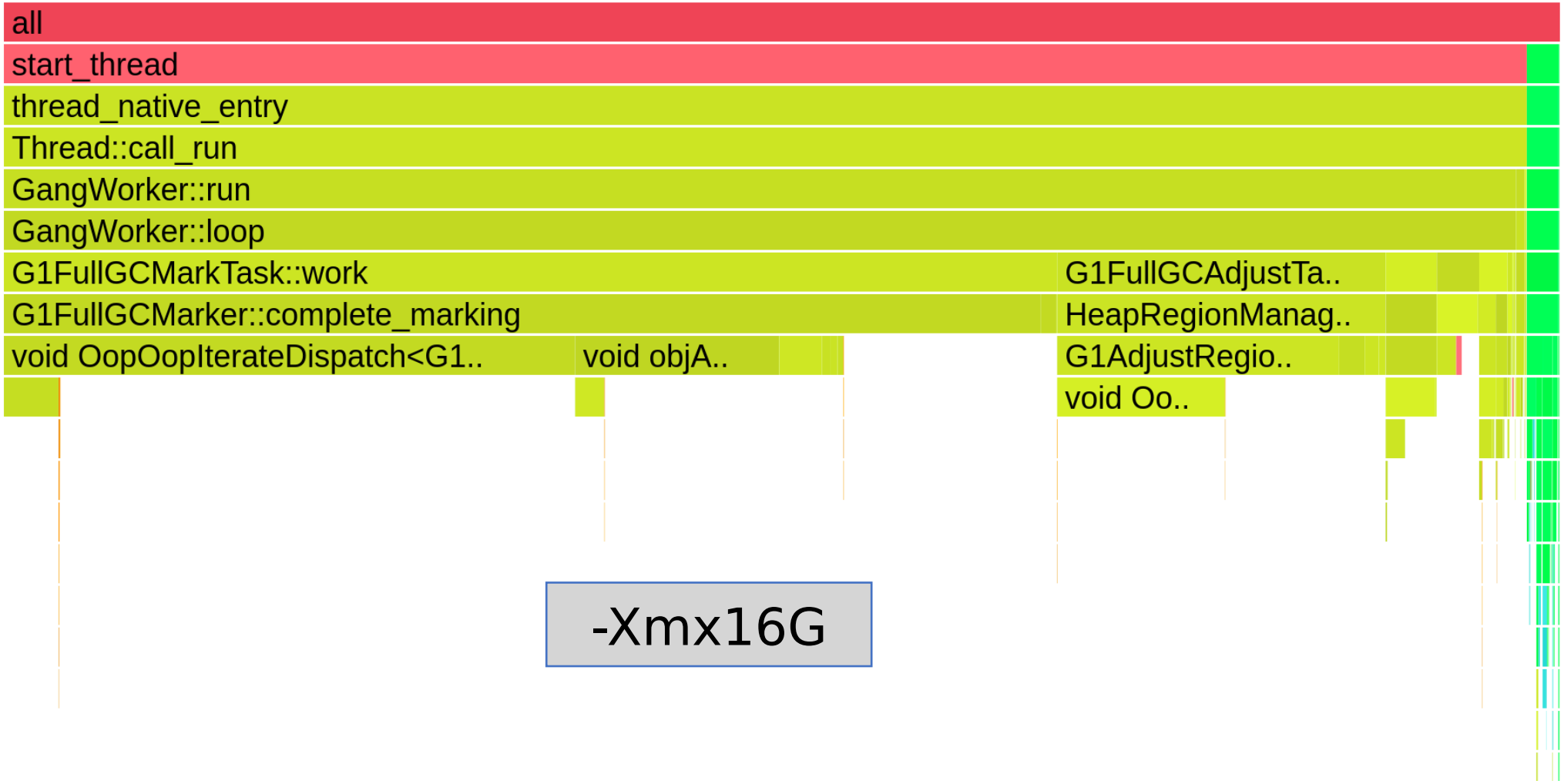
Mission Control



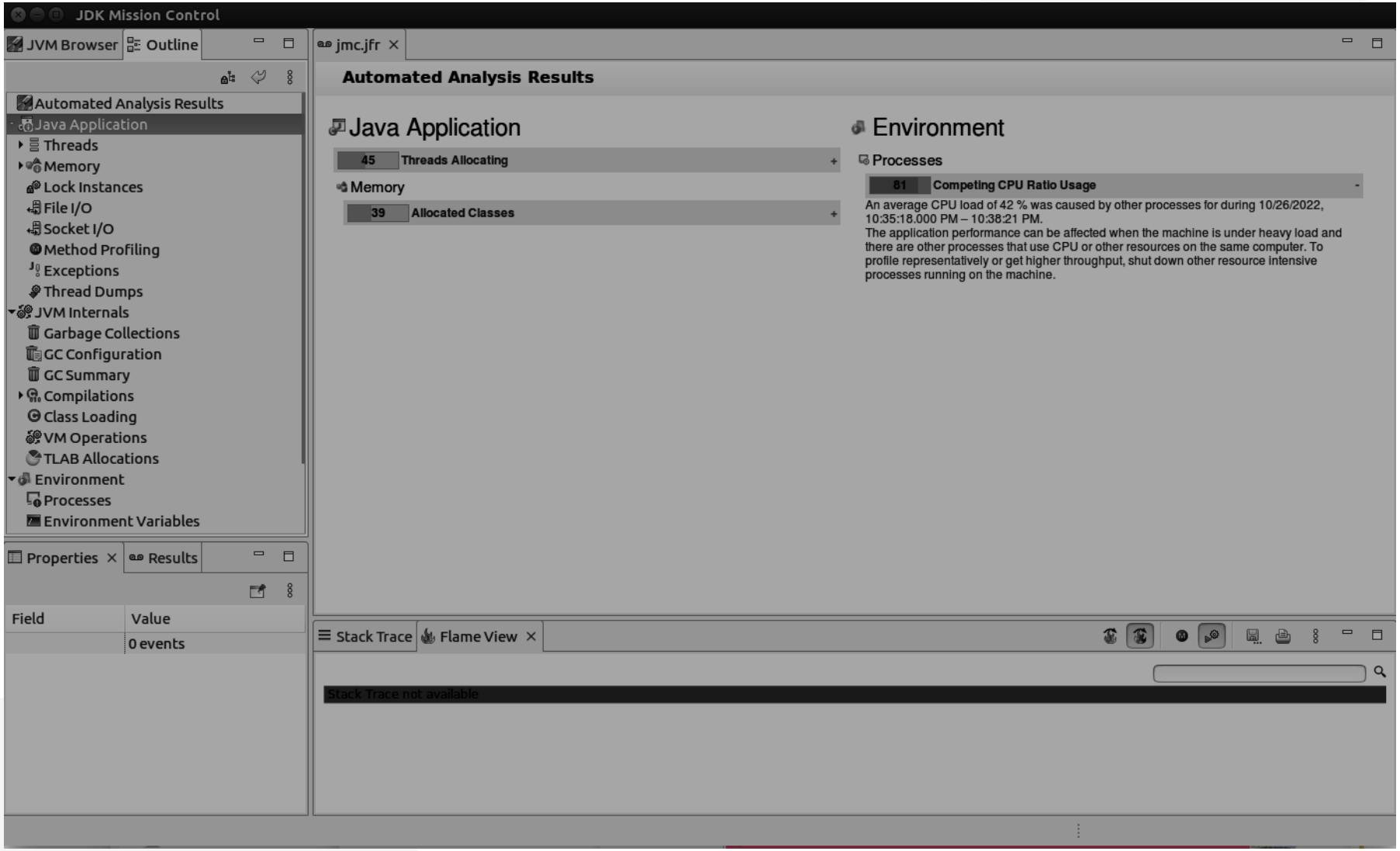
Mission Control



Mission Control



JMC смотрит сам себя



Что предстоит сделать?

- Загрузить

Что предстоит сделать?

- Загрузить
- Отобразить

Что предстоит сделать?

- Подготовить
- Загрузить
- Отобразить

Требования и ограничения

Требования и ограничения

- HeatMap за сутки

Требования и ограничения

- HeatMap за сутки
- FlameGraph за любой период

Требования и ограничения

- HeatMap за сутки
- FlameGraph за любой период
- Гранулярность 20 мс

Требования и ограничения

- HeatMap за сутки
- FlameGraph за любой период
- Гранулярность 20 мс
- Профили до 4 ГБ

Требования и ограничения

- HeatMap за сутки
- FlameGraph за любой период
- Гранулярность 20 мс
- Профили до 4 ГБ
- CPU: 6 млн сэмплов

Требования и ограничения

- HeatMap за сутки
- FlameGraph за любой период
- Гранулярность 20 мс
- Профили до 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов

Готовим данные

Зачем готовить данные?

Зачем готовить данные?

- JFR: 4 ГБ

Зачем готовить данные?

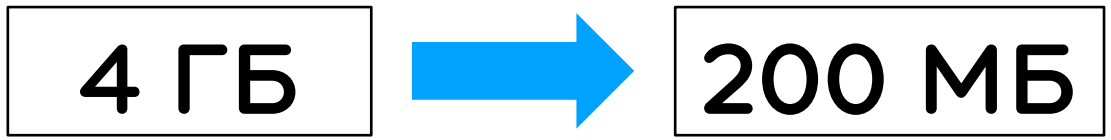
- JFR: 4 ГБ
- CPU: 6 млн сэмплов

Зачем готовить данные?

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов

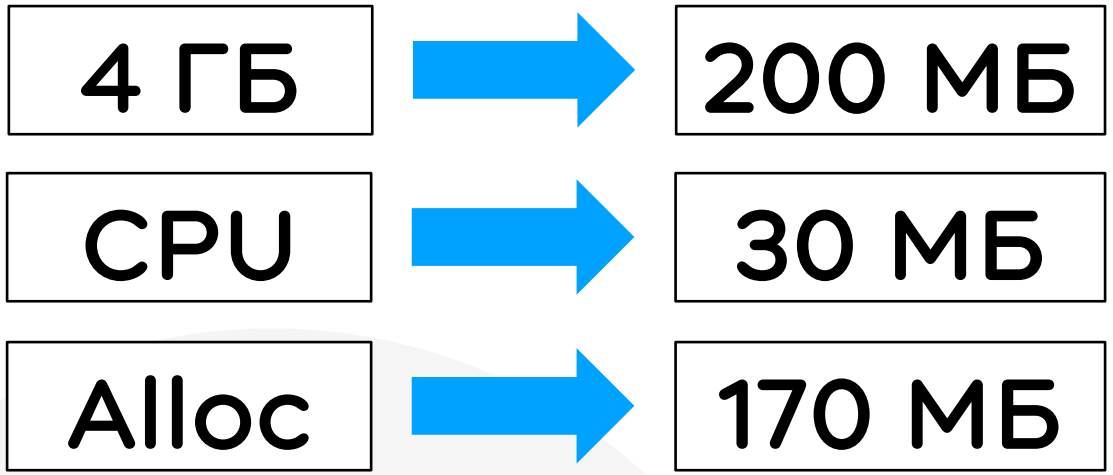
Зачем готовить данные?

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов



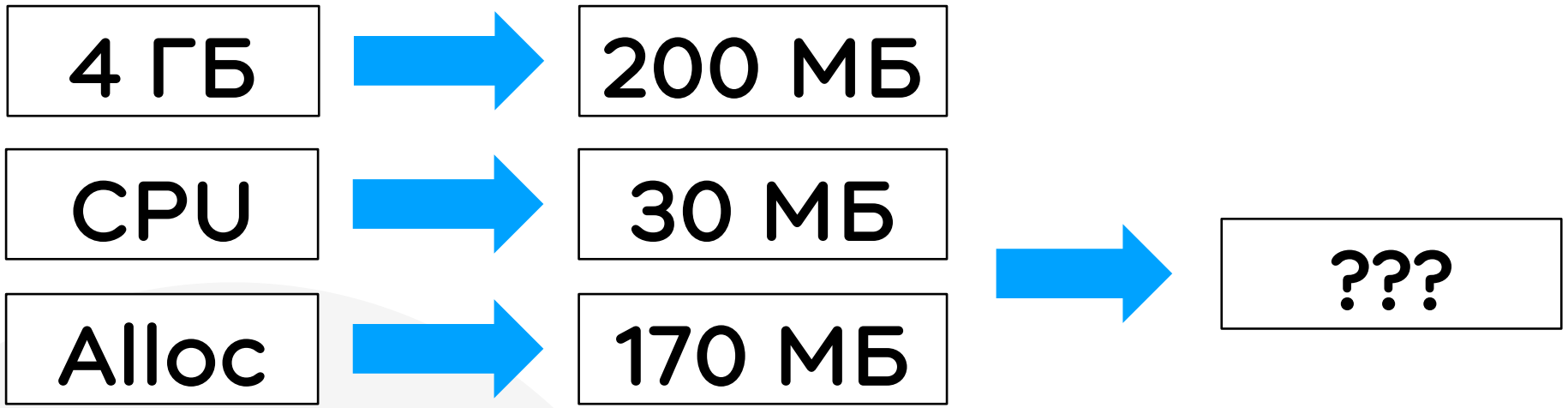
Зачем готовить данные?

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов



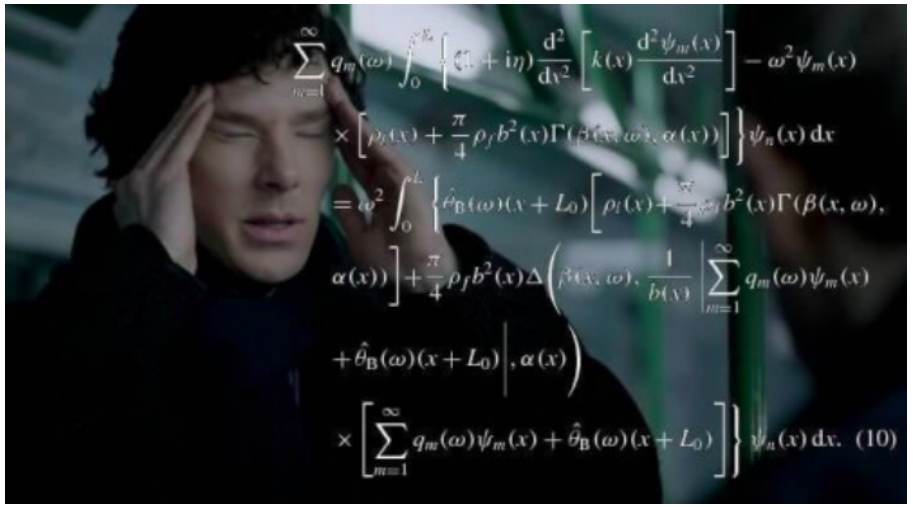
Зачем готовить данные?

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов



Зачем готовить данные?

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов

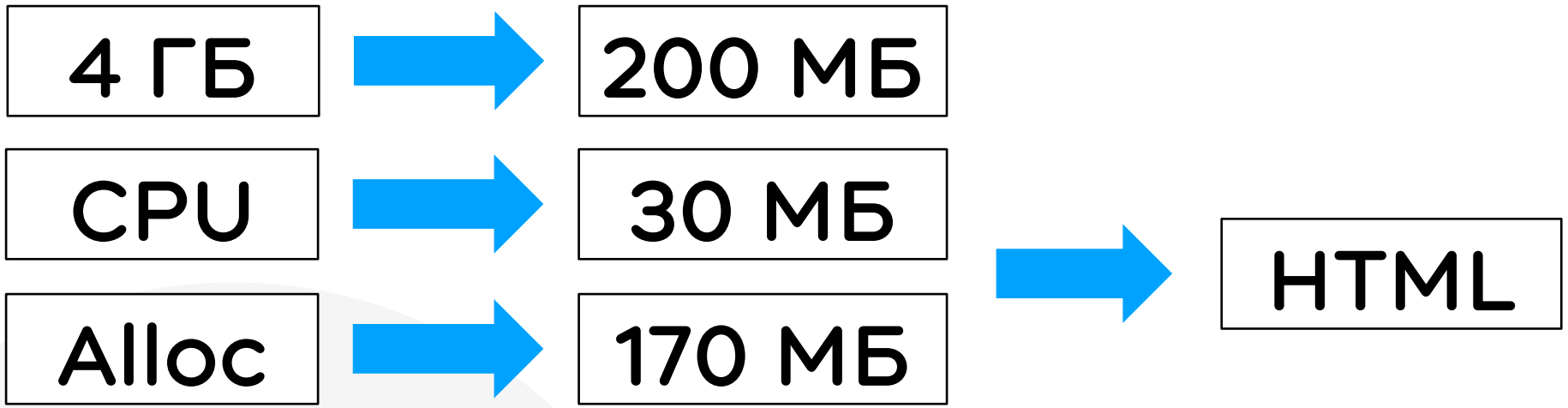


4 ГБ	→	200 МБ
CPU	→	30 МБ
Alloc	→	170 МБ

→ ???

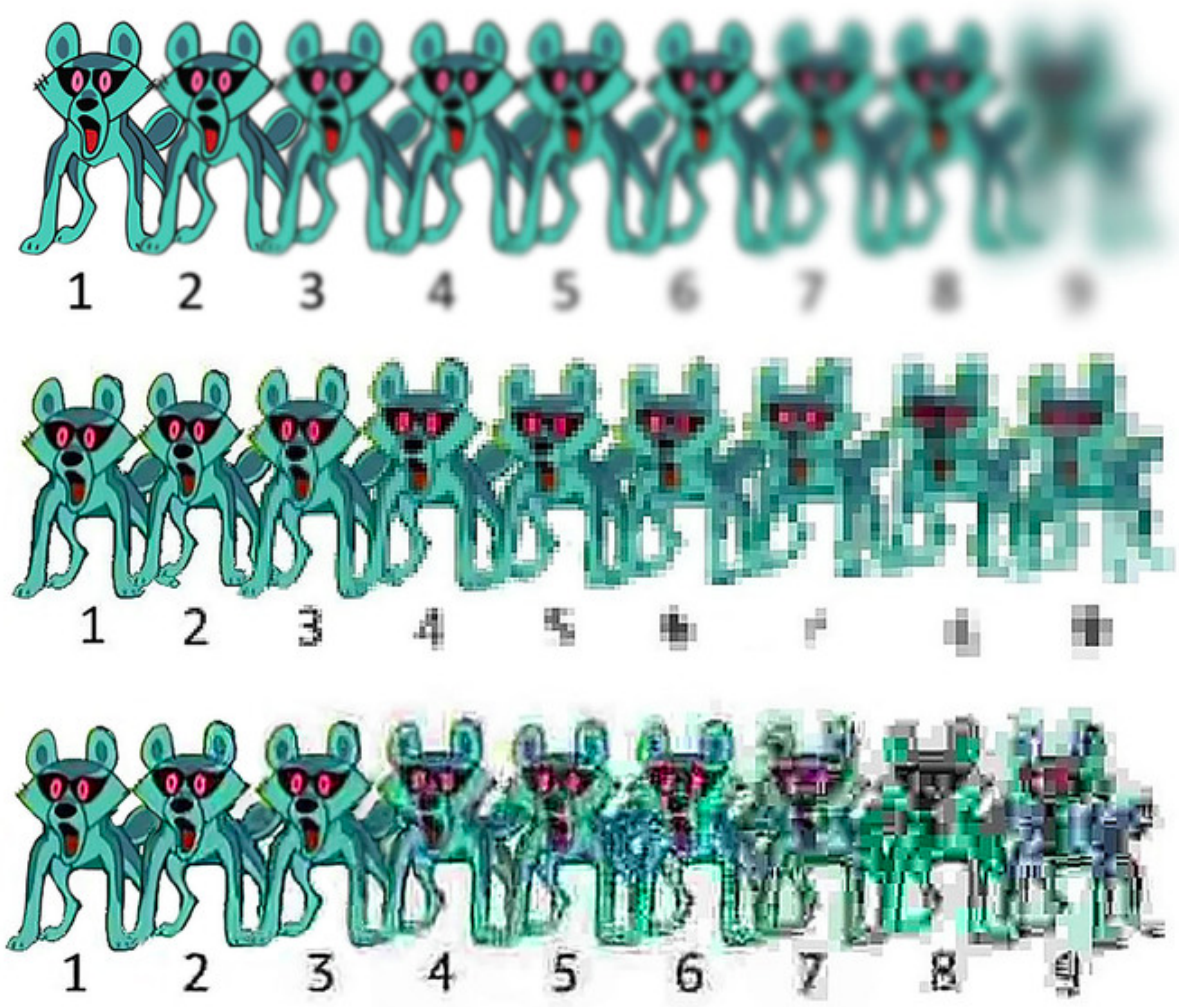
Зачем готовить данные?

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов

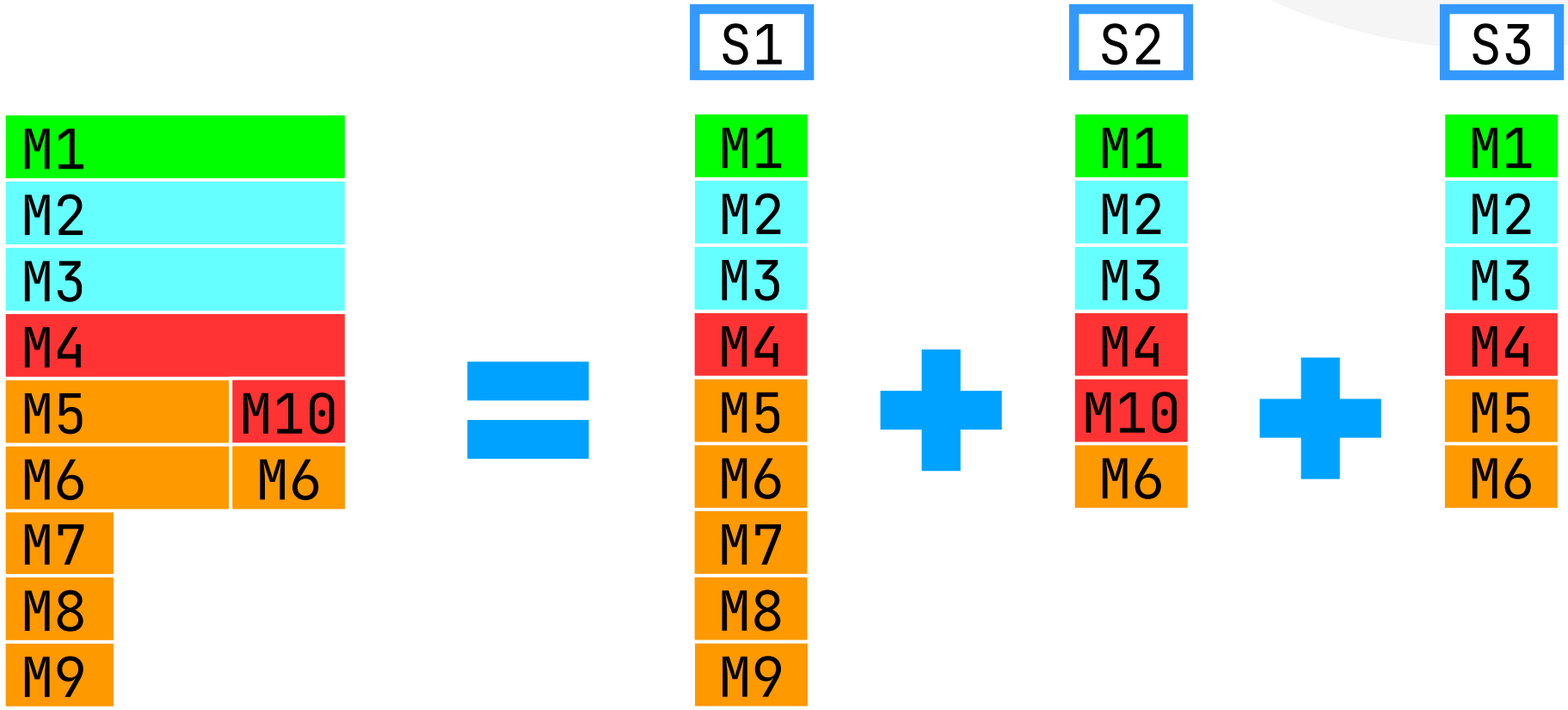


Готовим данные

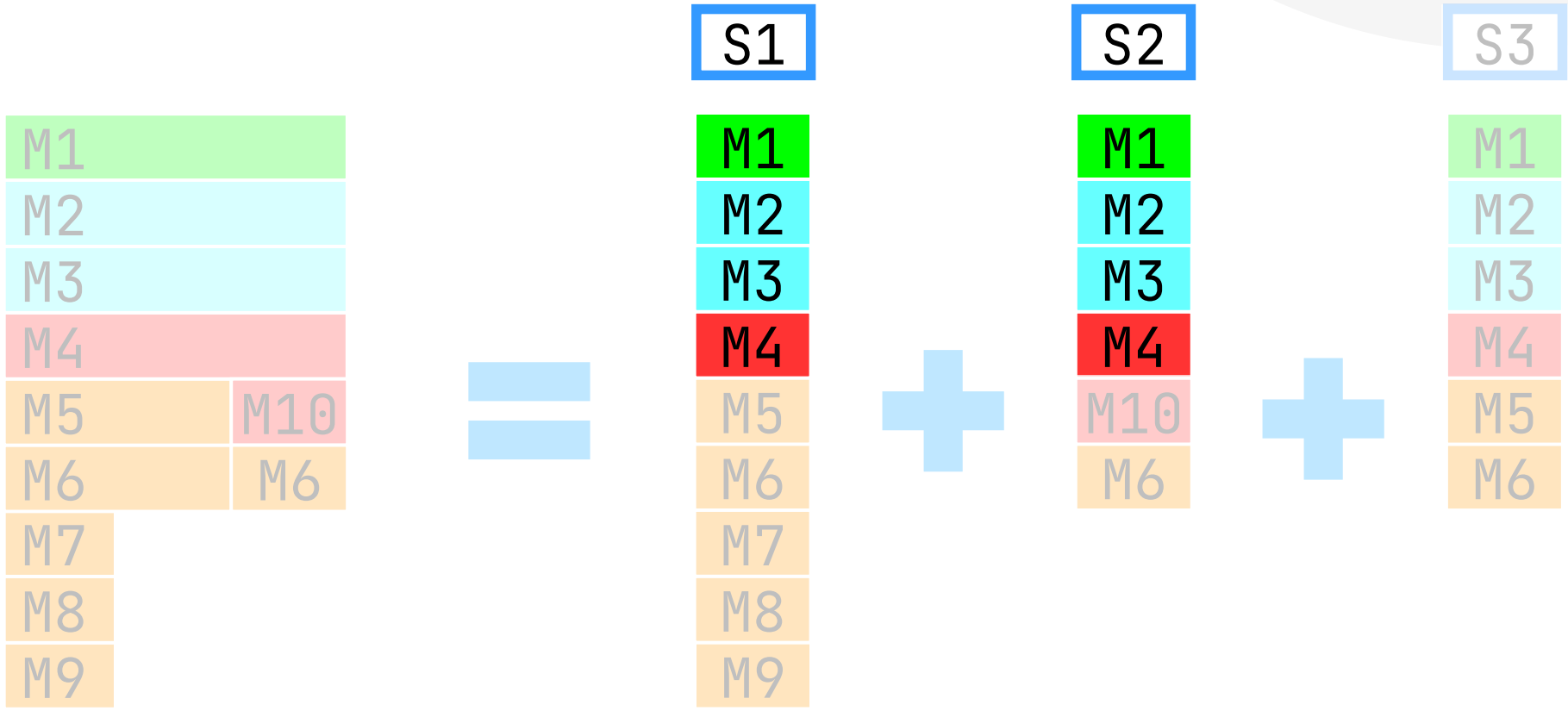
Сжатие



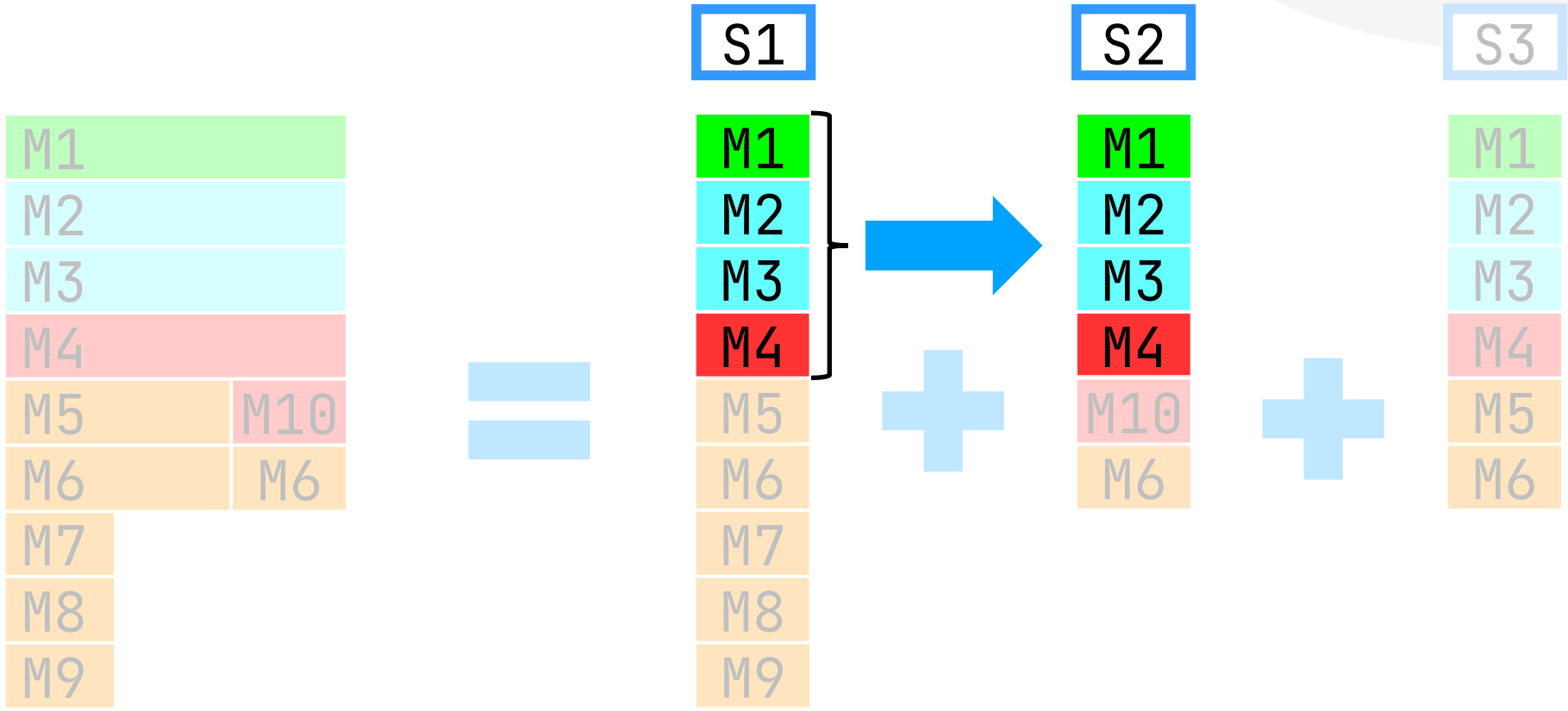
Сжатие


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
S2
M1
M2
M3
M4
M10
M6
+
S3
M1
M2
M3
M4
M5
M6

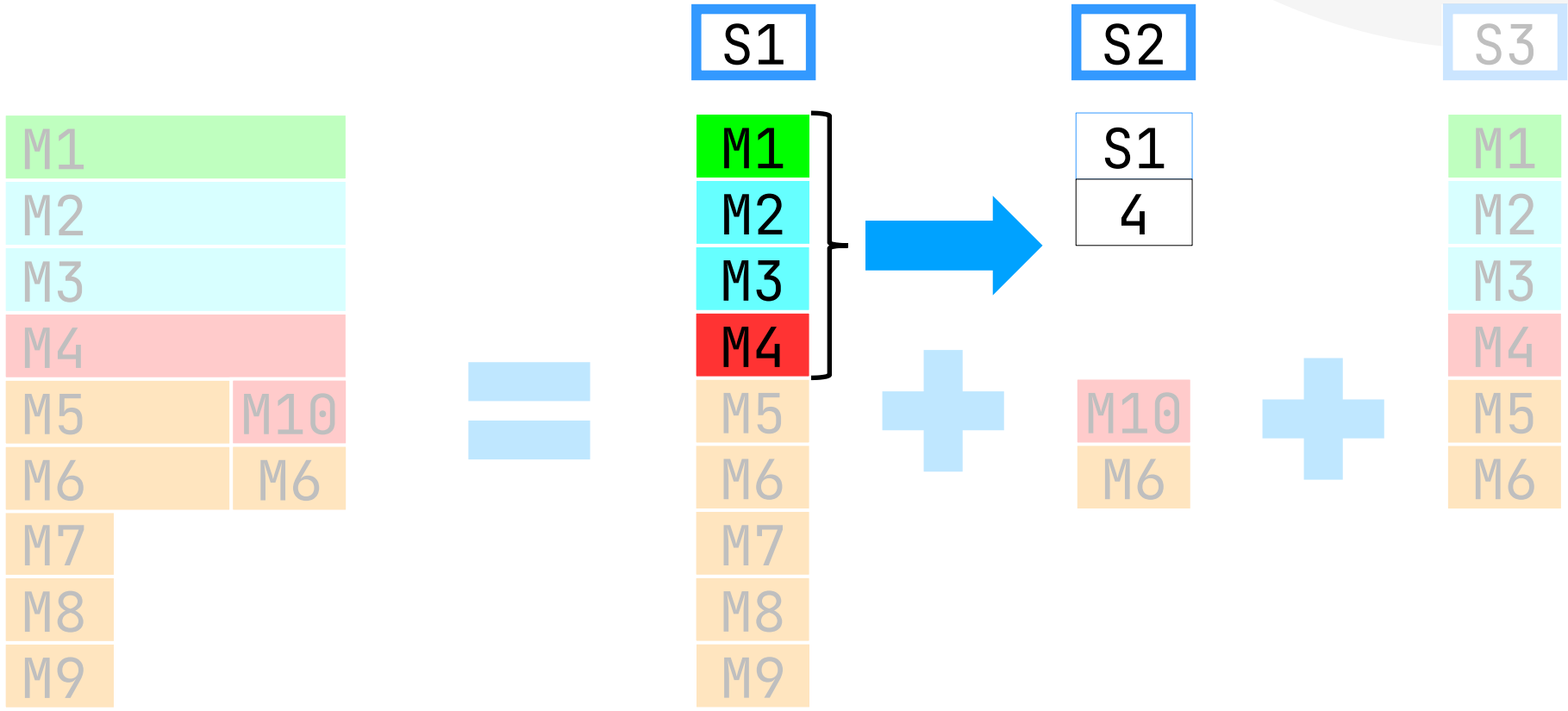
Сжатие


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
S2
M1
M2
M3
M4
M10
M6
+
S3
M1
M2
M3
M4
M5
M6

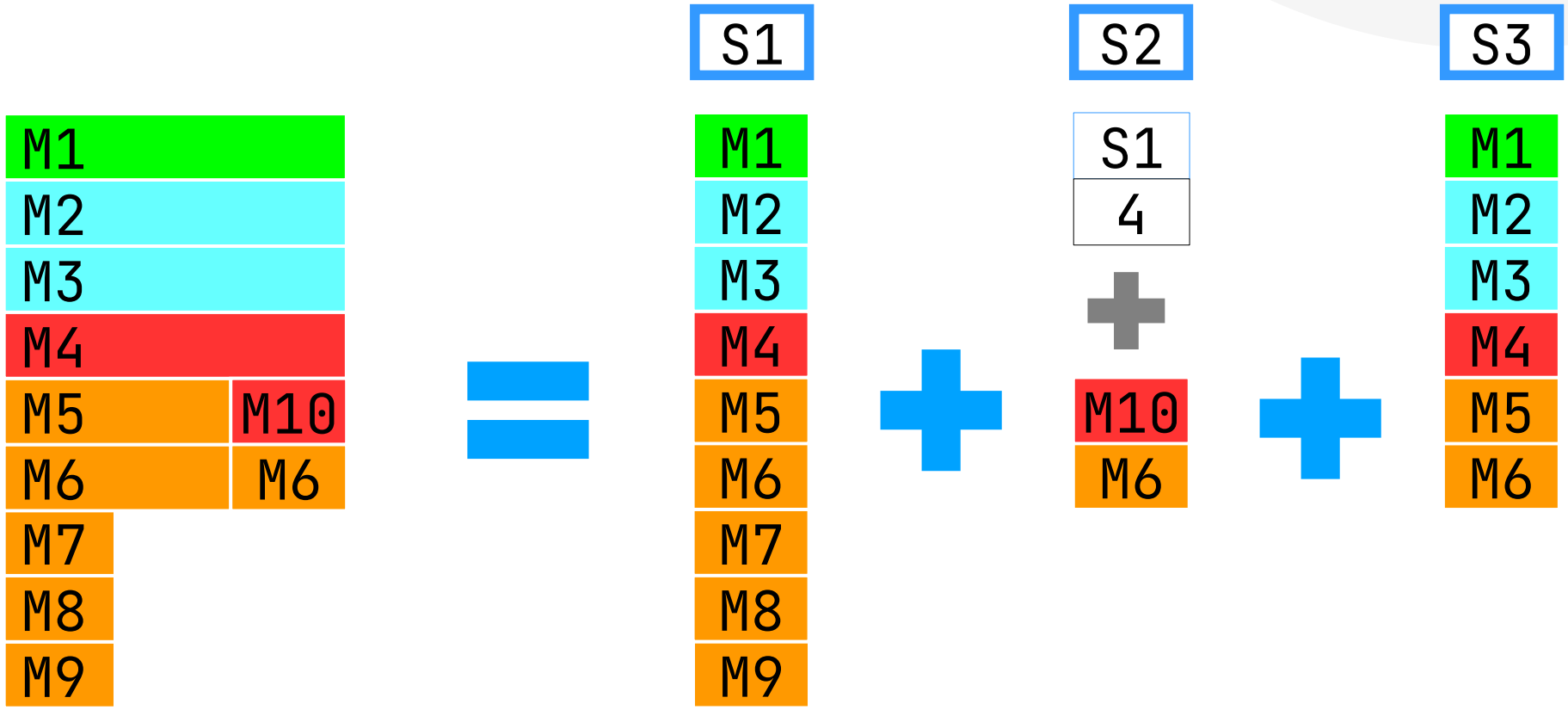
Сжатие


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M1
M2
M3
M4
M10
M6
S3
M1
M2
M3
M4
M5
M6

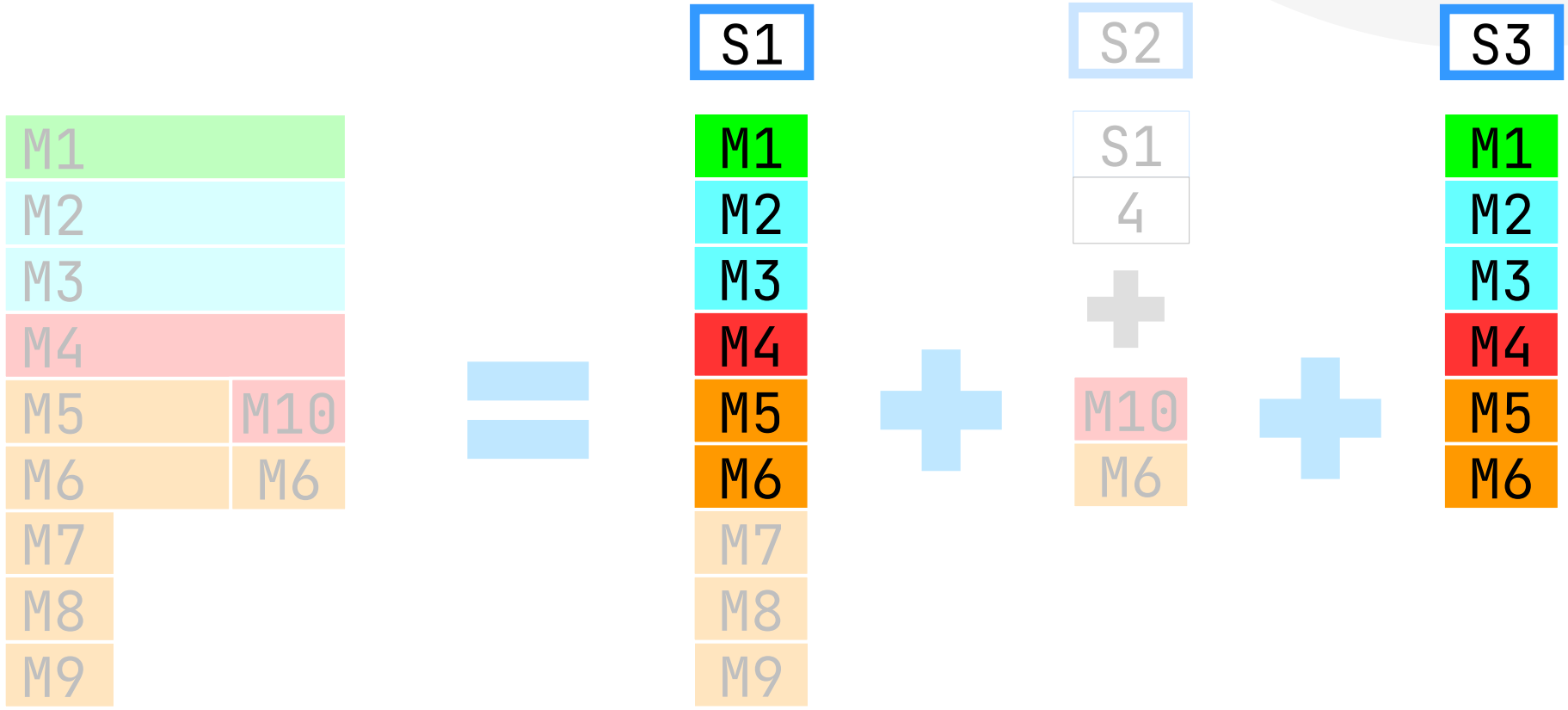
Сжатие


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
S1
4
M10
M6
S3
M1
M2
M3
M4
M5
M6

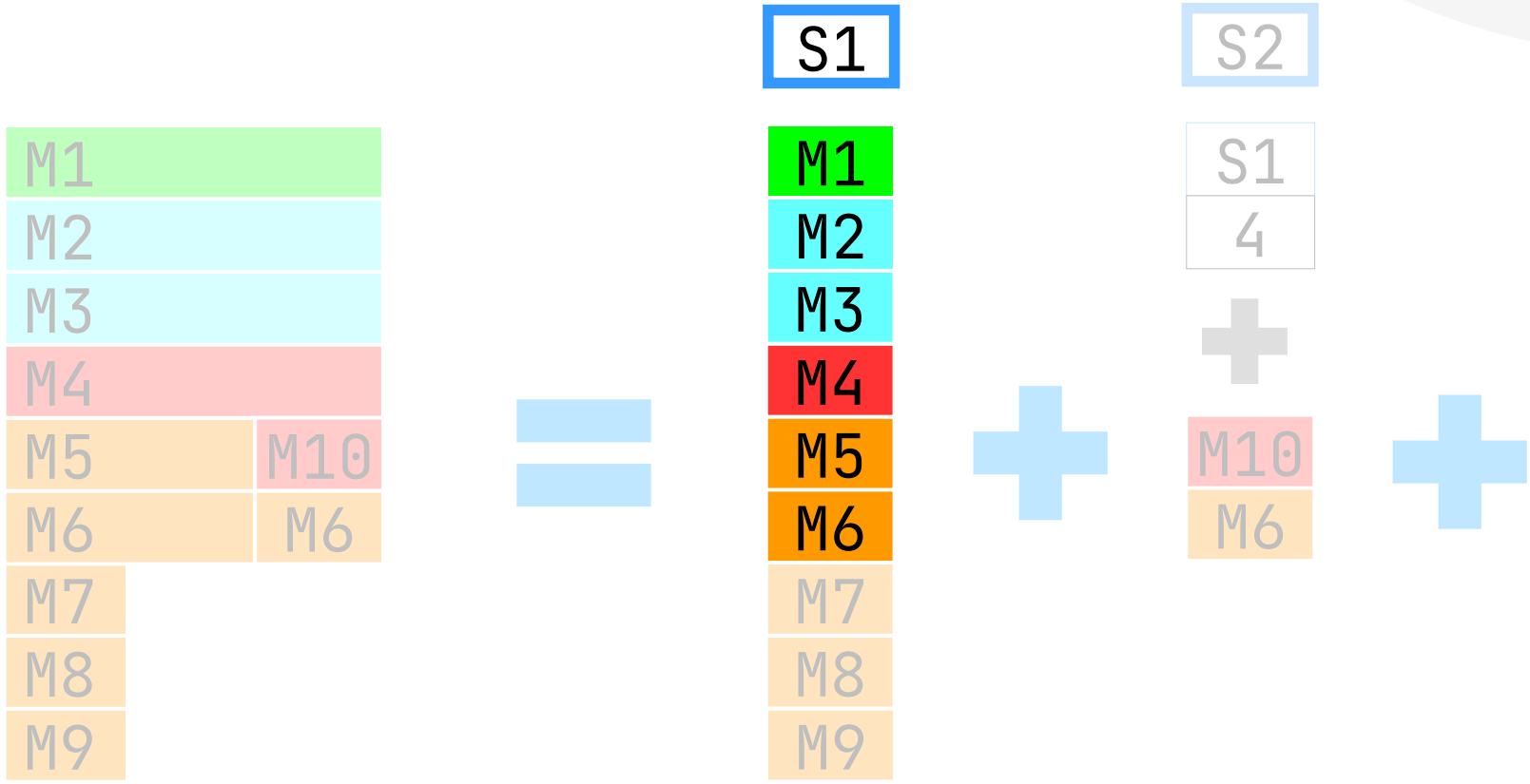
Сжатие


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
S2
S1
4
+
M10
M6
+
S3
M1
M2
M3
M4
M5
M6

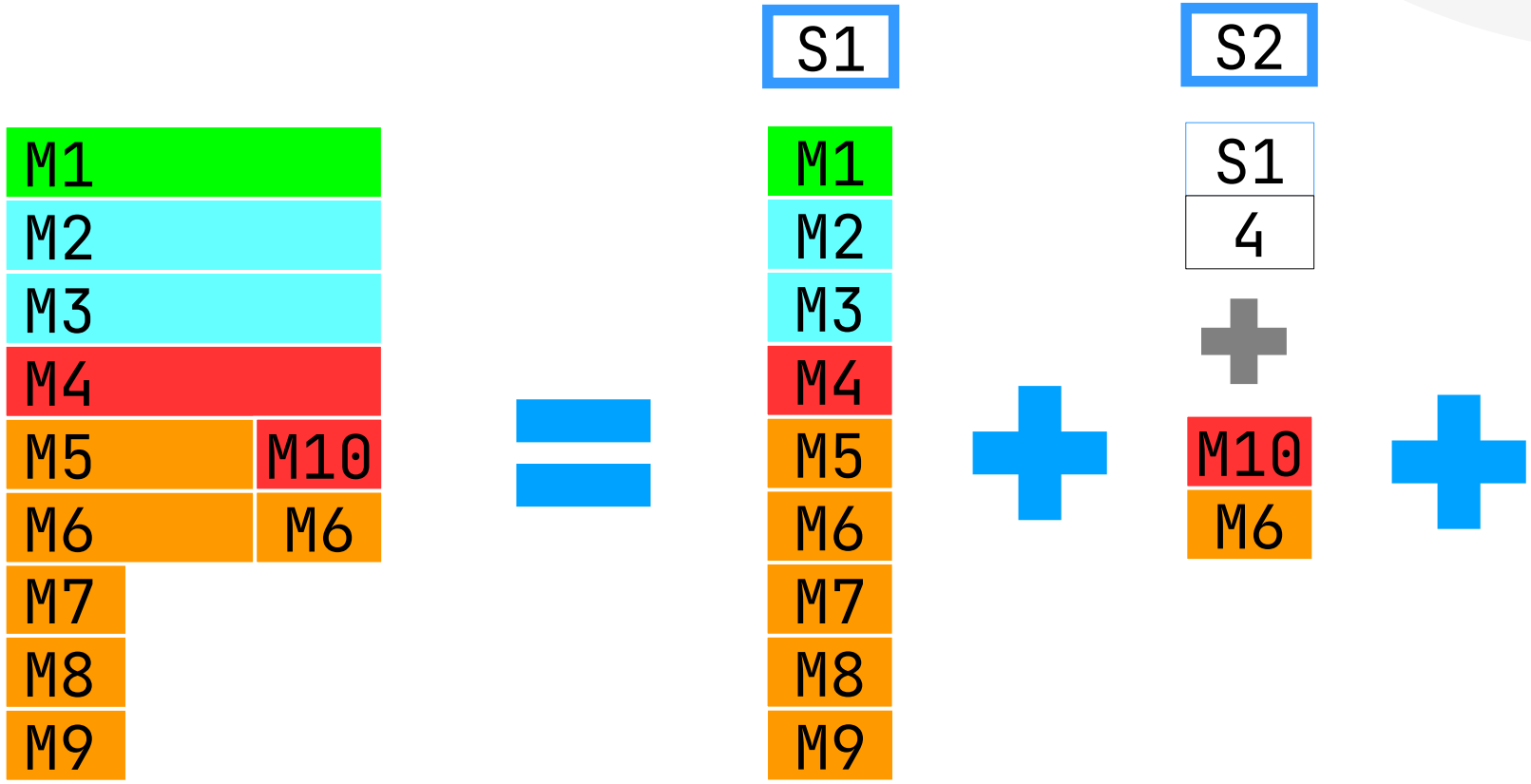
Сжатие


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
S2
S1
4
+
M10
M6
+
S3
M1
M2
M3
M4
M5
M6

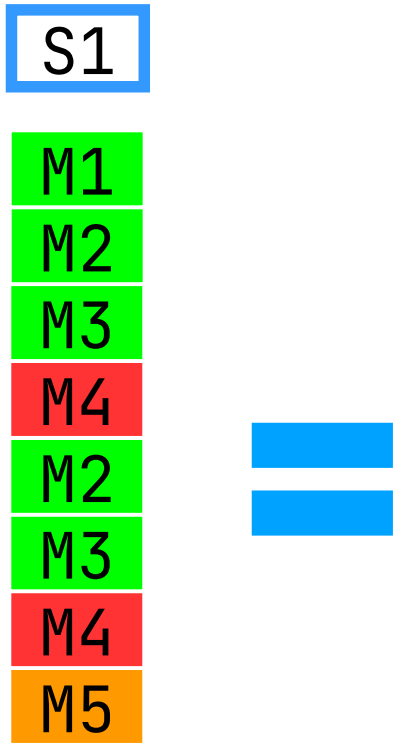
Сжатие


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
=
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
+
S2
S1
4
+
M10
M6
+
S3
S1
6

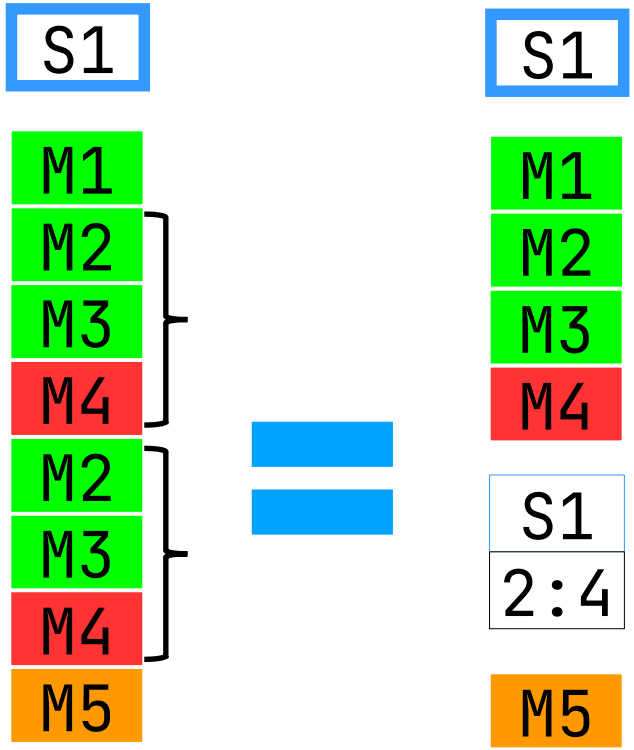
Сжатие


M1
M2
M3
M4
M5
M10
M6
M6
M7
M8
M9
S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
S1
4
M10
M6
S3
S1
6

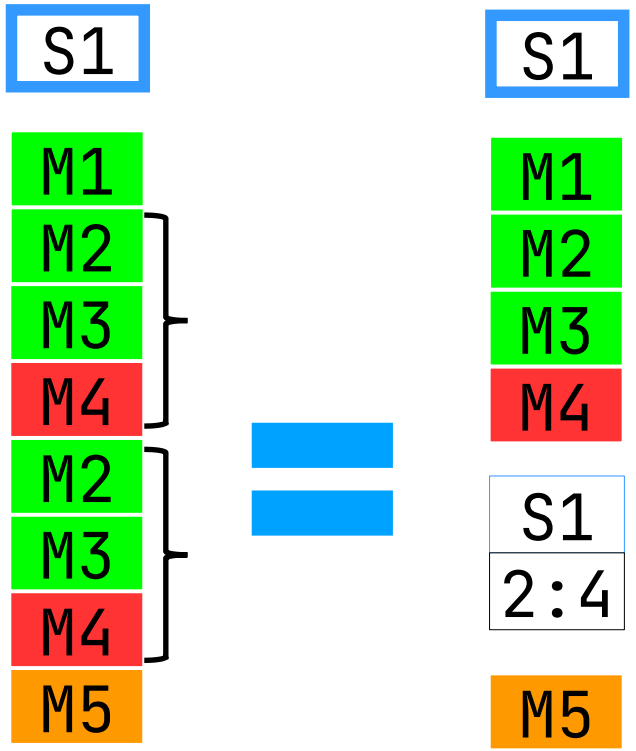
Сжатие



Сжатие



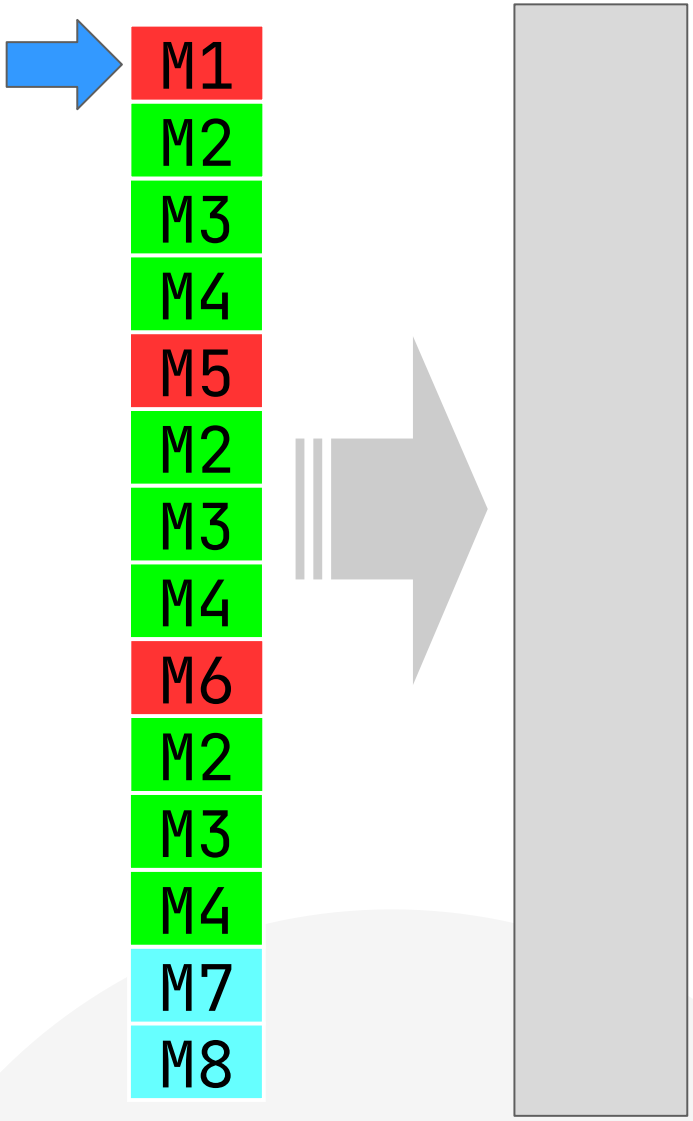
Сжатие



Что если

Помнить то что вижу часто?

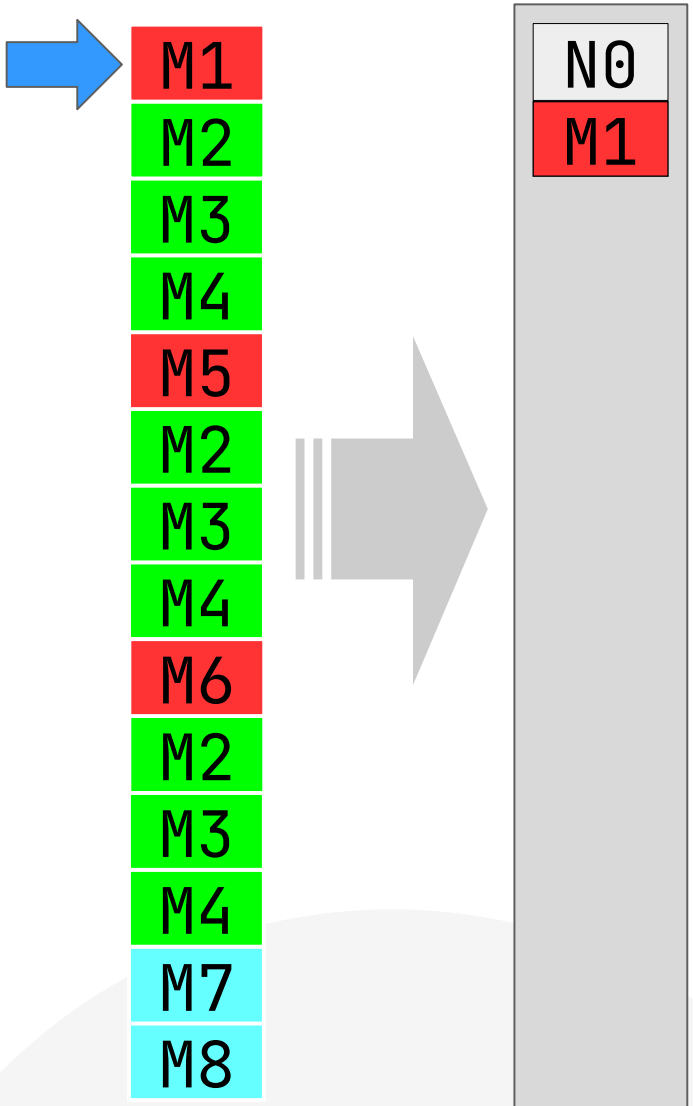
Сжимаем



Словарь

```
0: ""
```

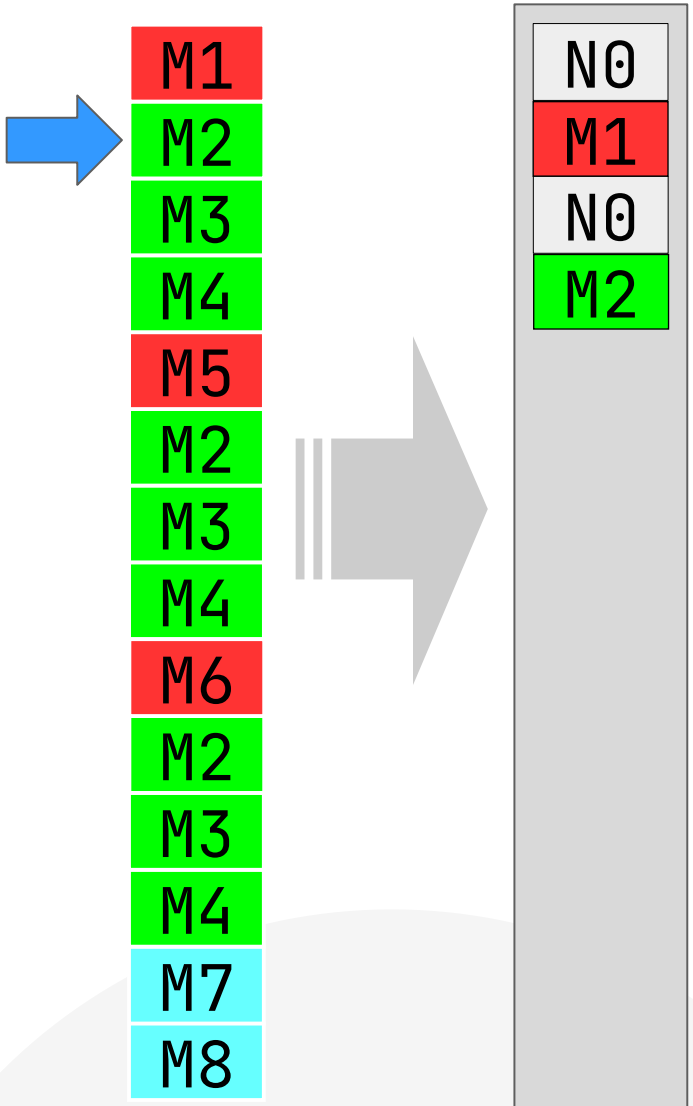
Сжимаем



Словарь

```
0: ""
1: "M1"
```

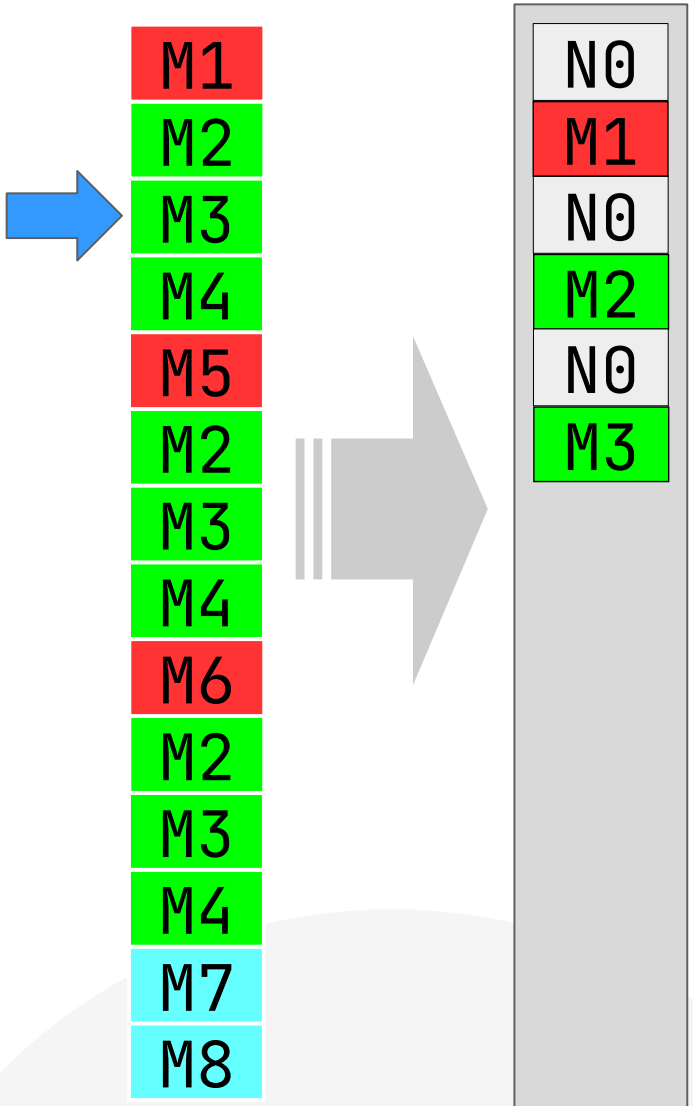
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
```

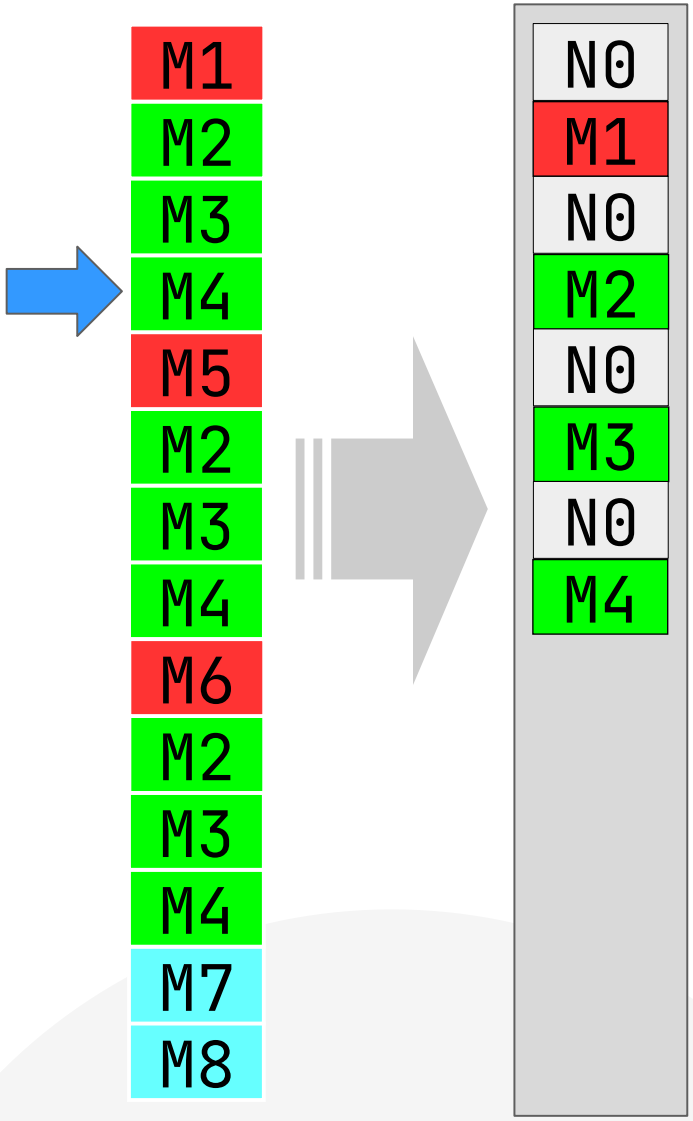
Сжимаем



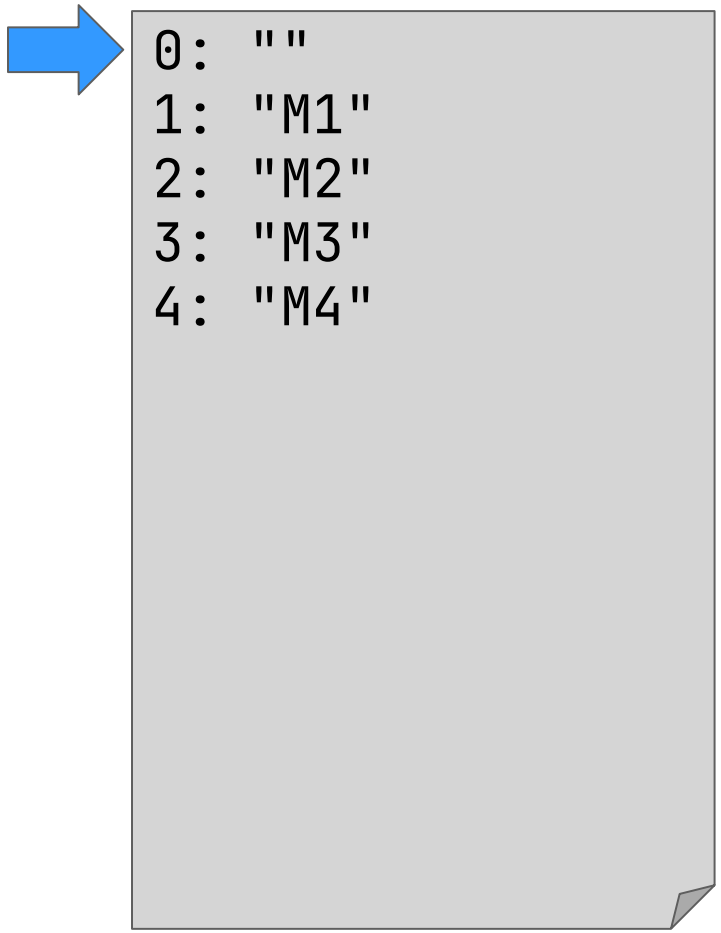
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
```

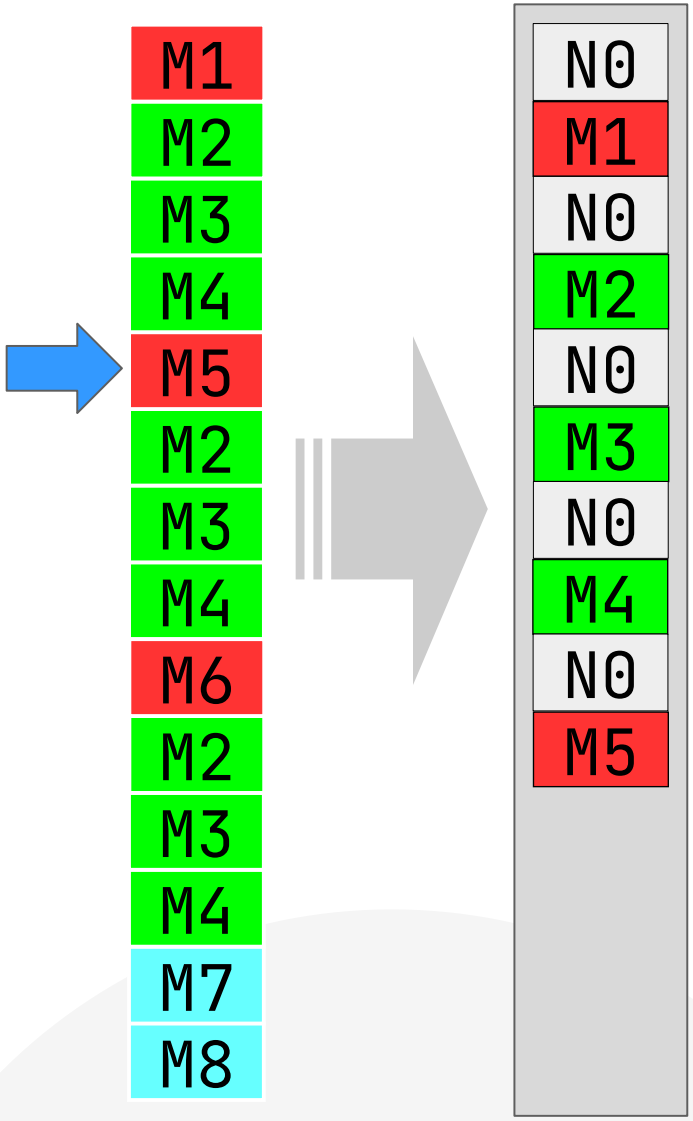
Сжимаем



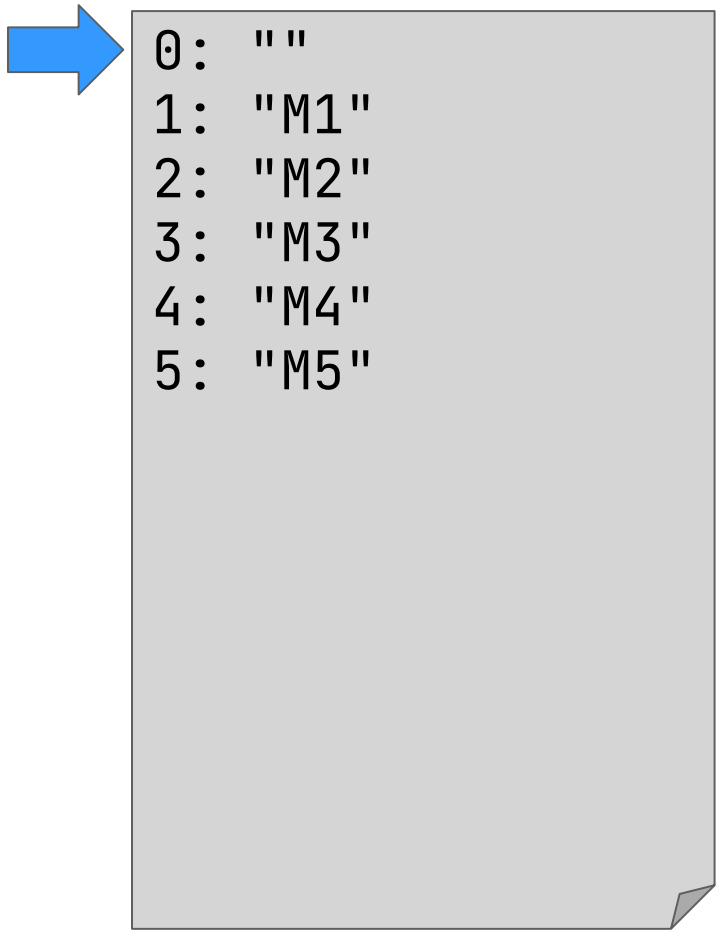
Словарь



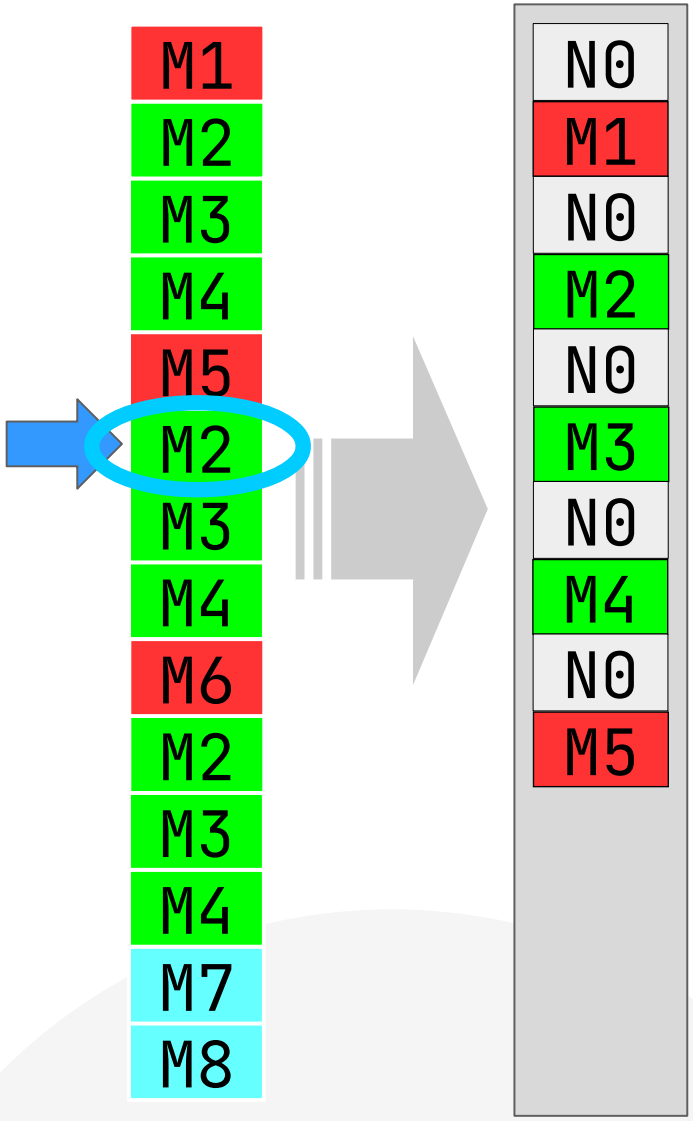
Сжимаем



Словарь



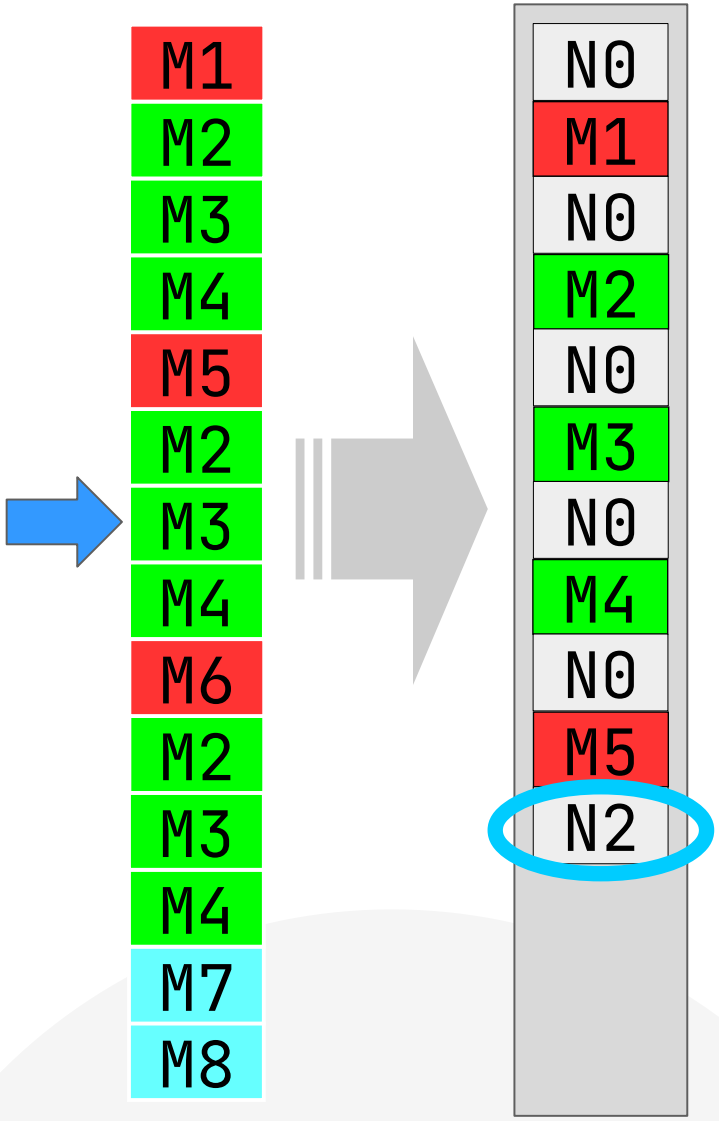
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
```

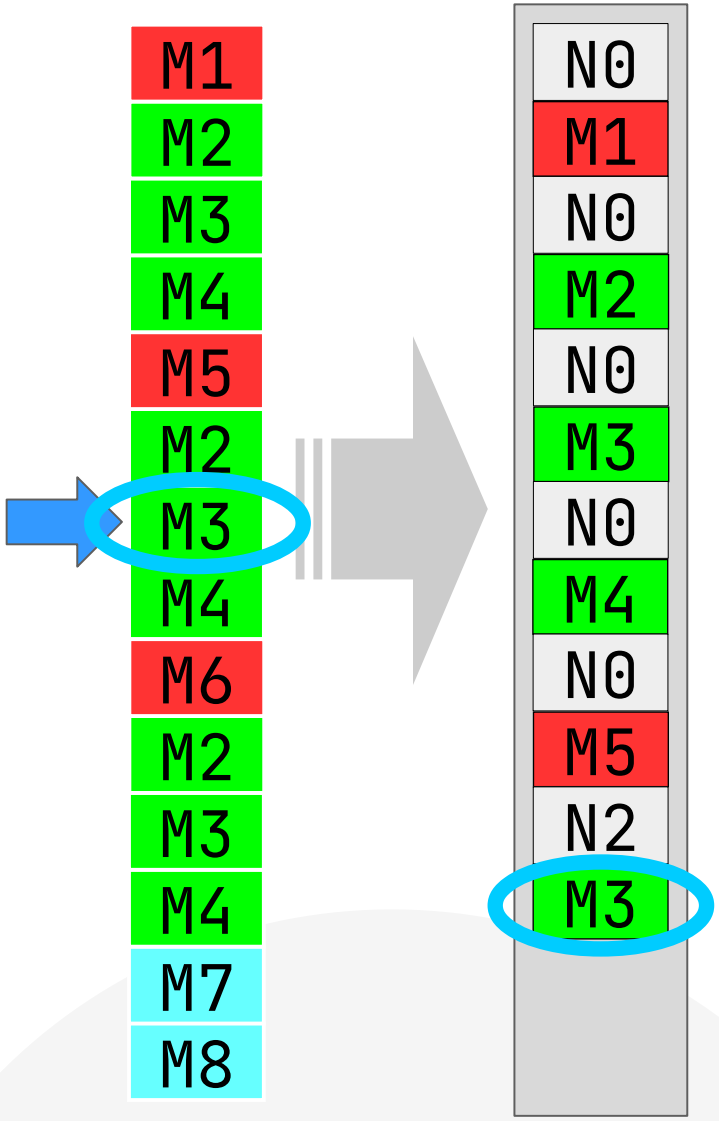
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
```

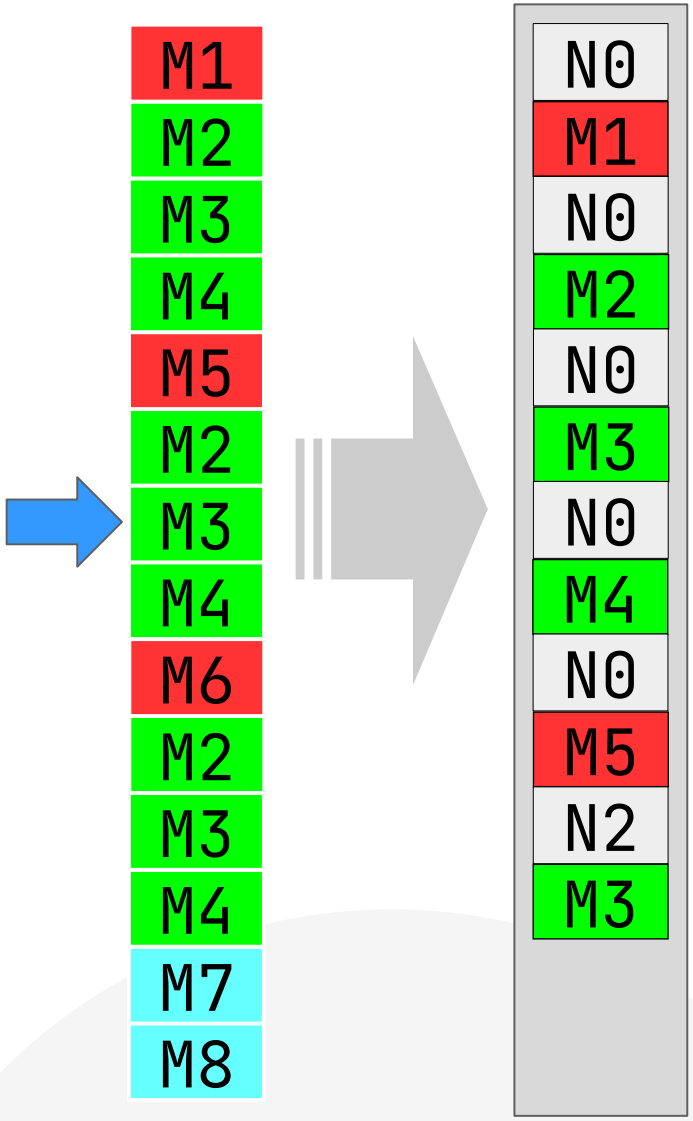
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
```

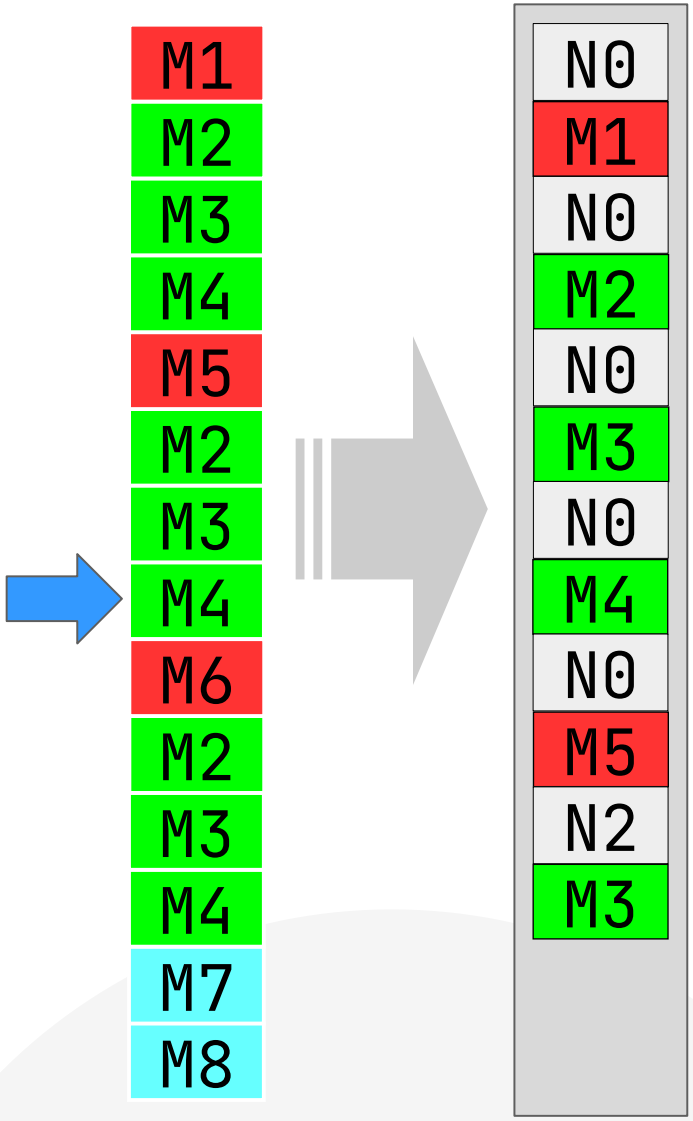
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
```

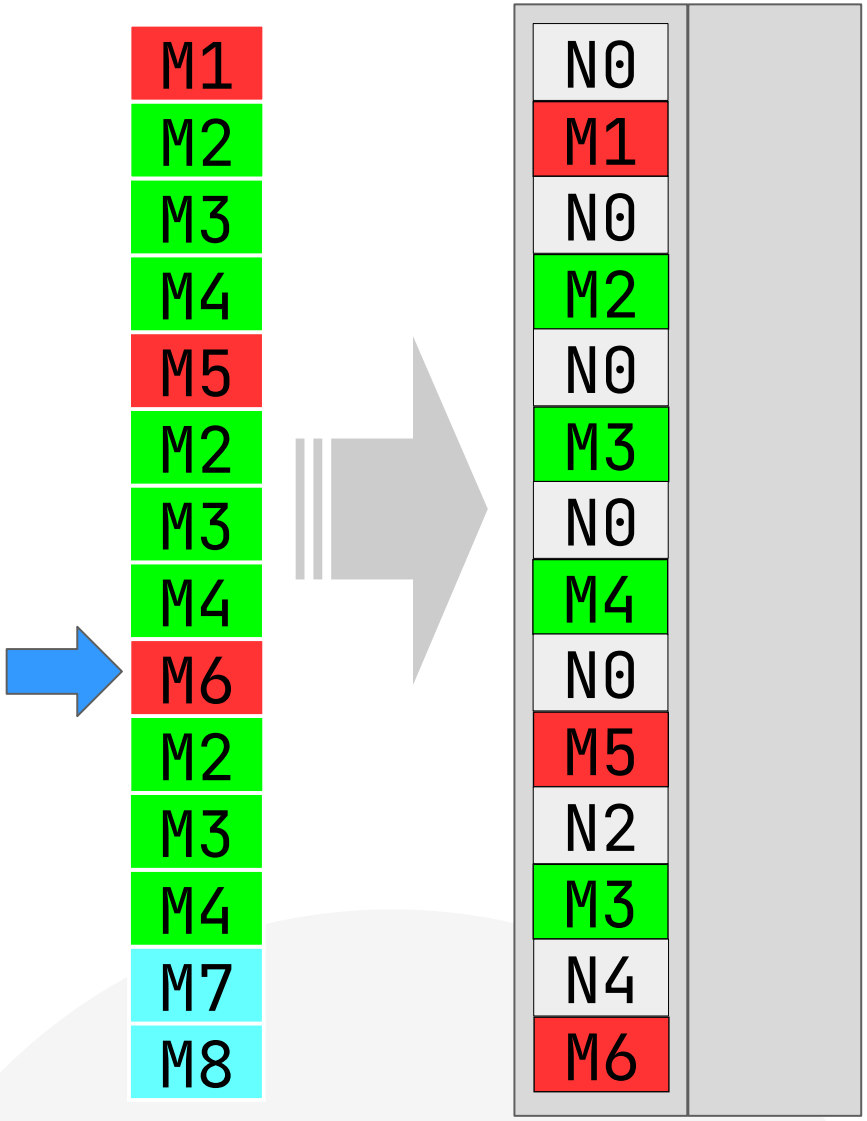
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
```

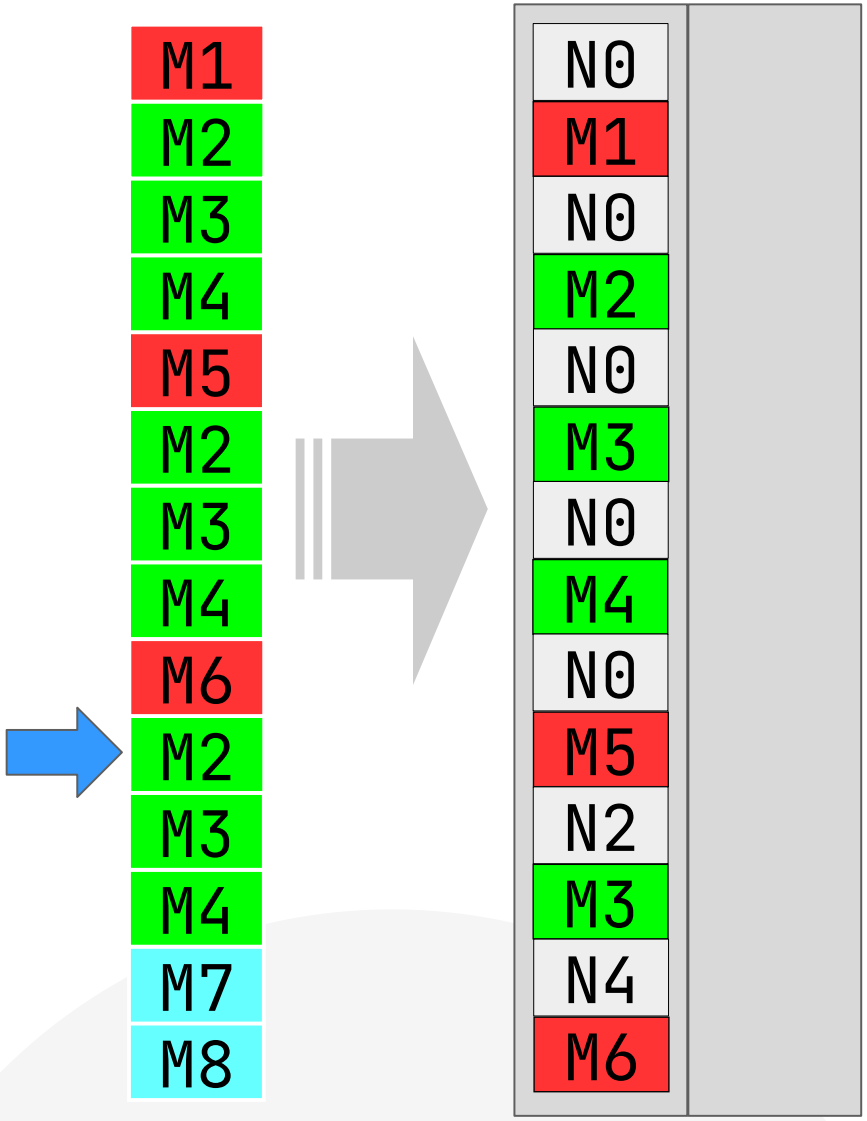
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

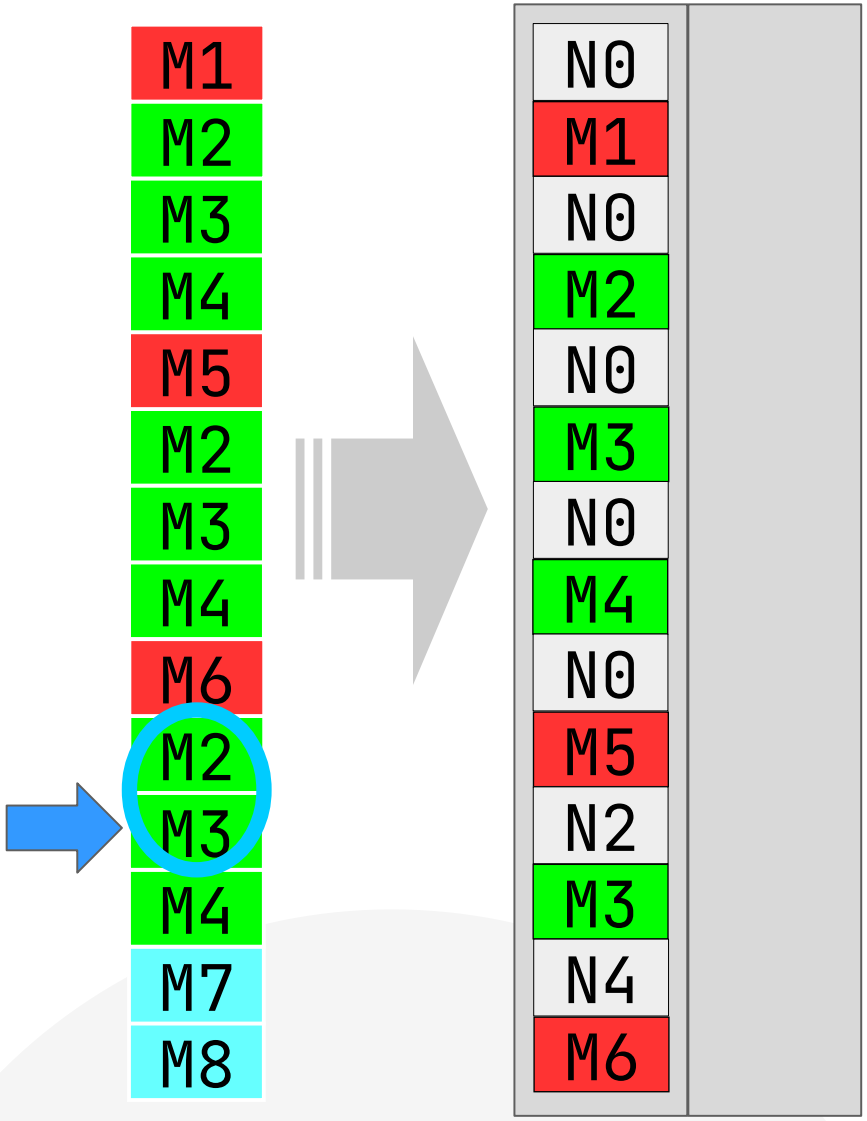
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

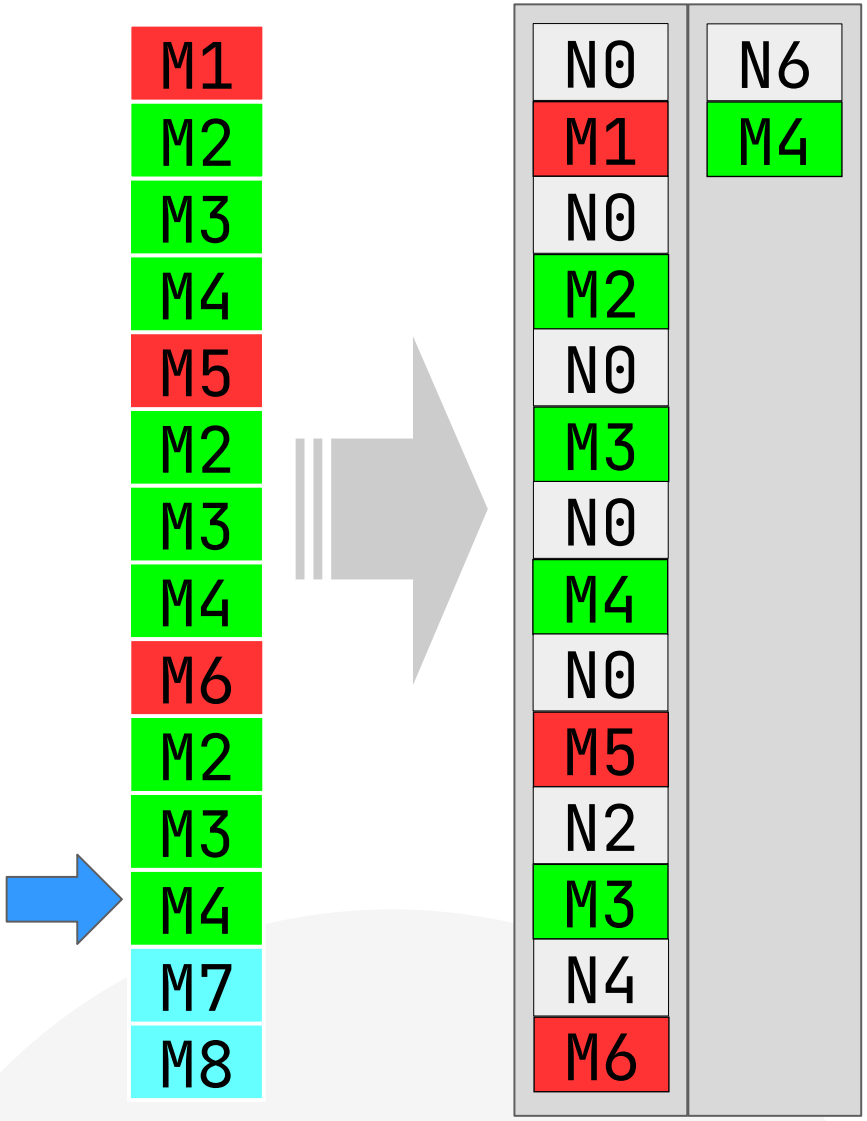
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

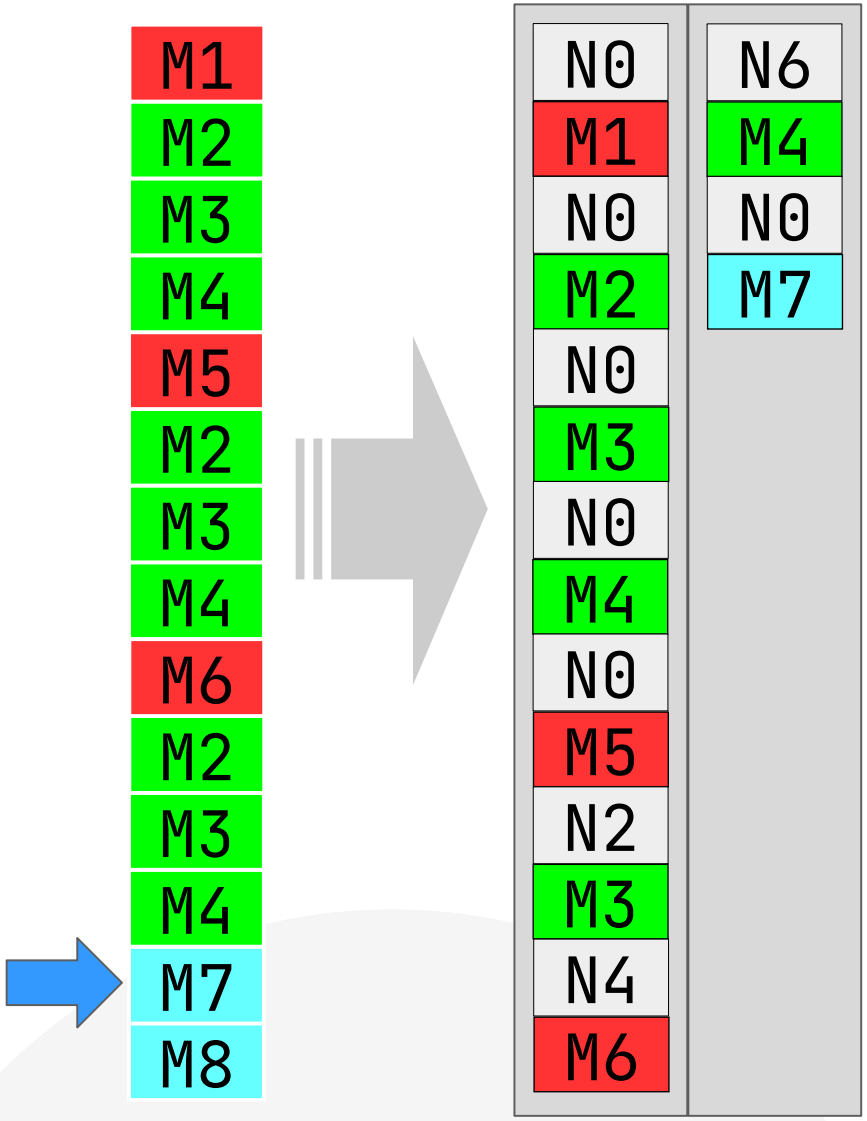
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
```

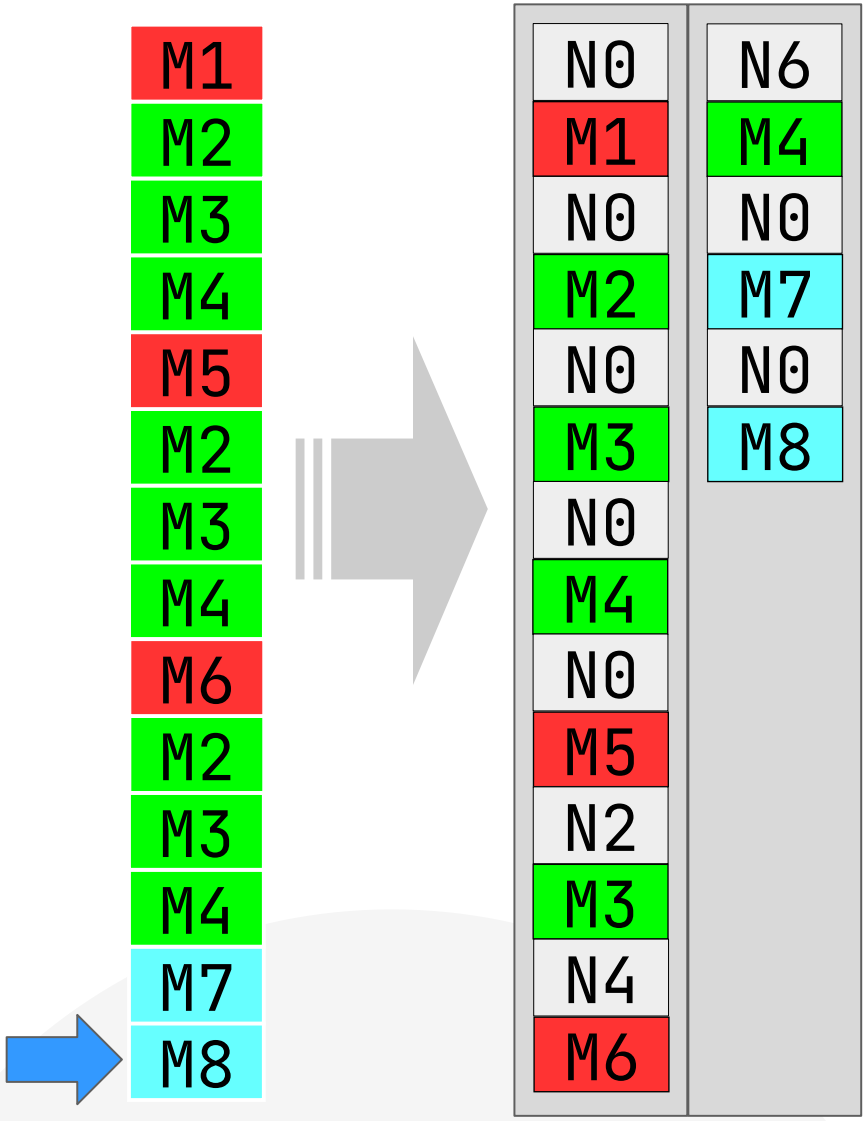
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
```

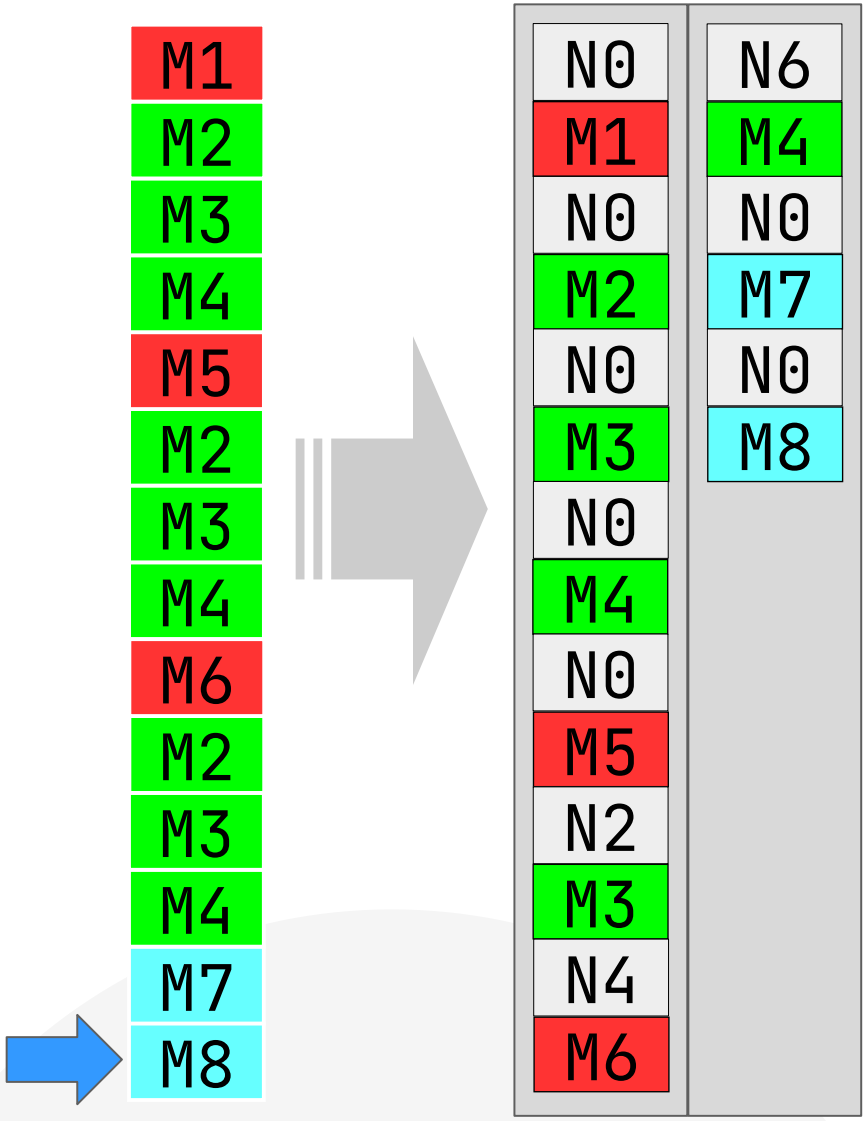
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
```

Сжимаем

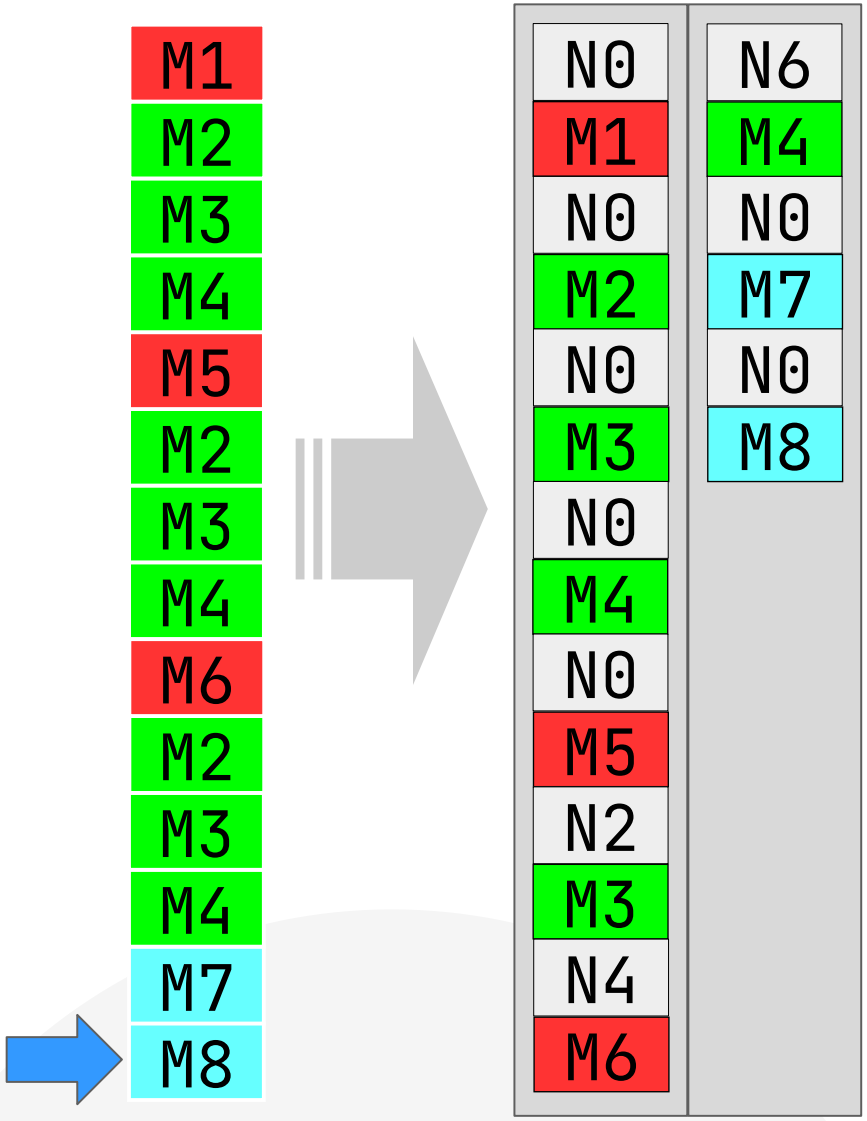


Сжал почти в полтора раза!

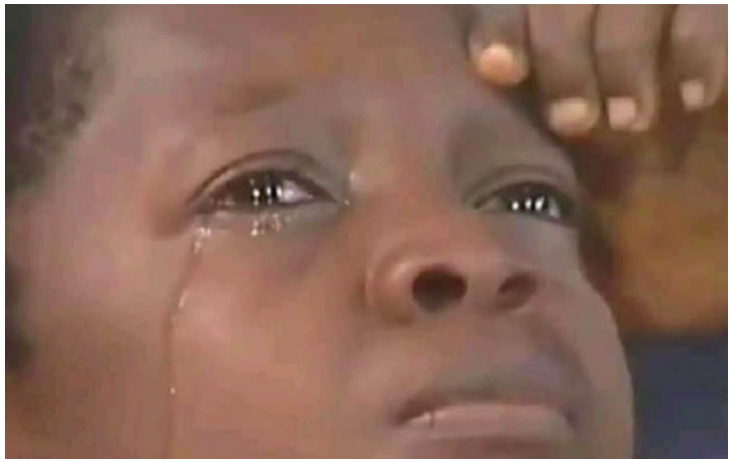
Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
```

Сжимаем



Сжал почти в полтора раза!

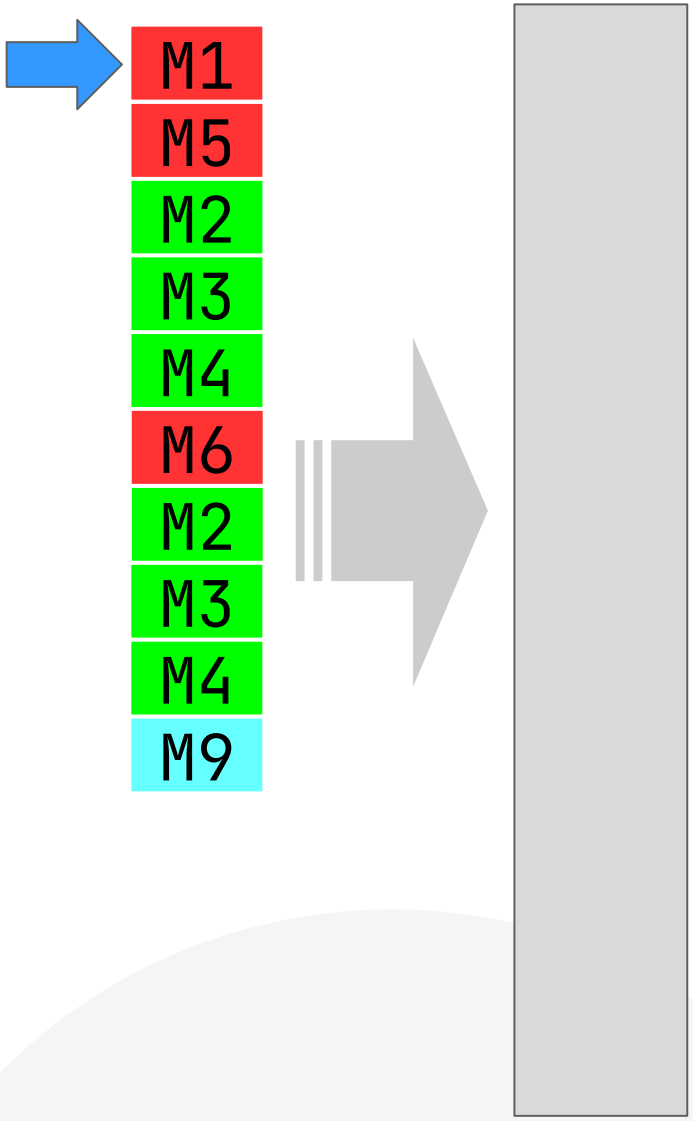


Но не в ту сторону...

Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
```

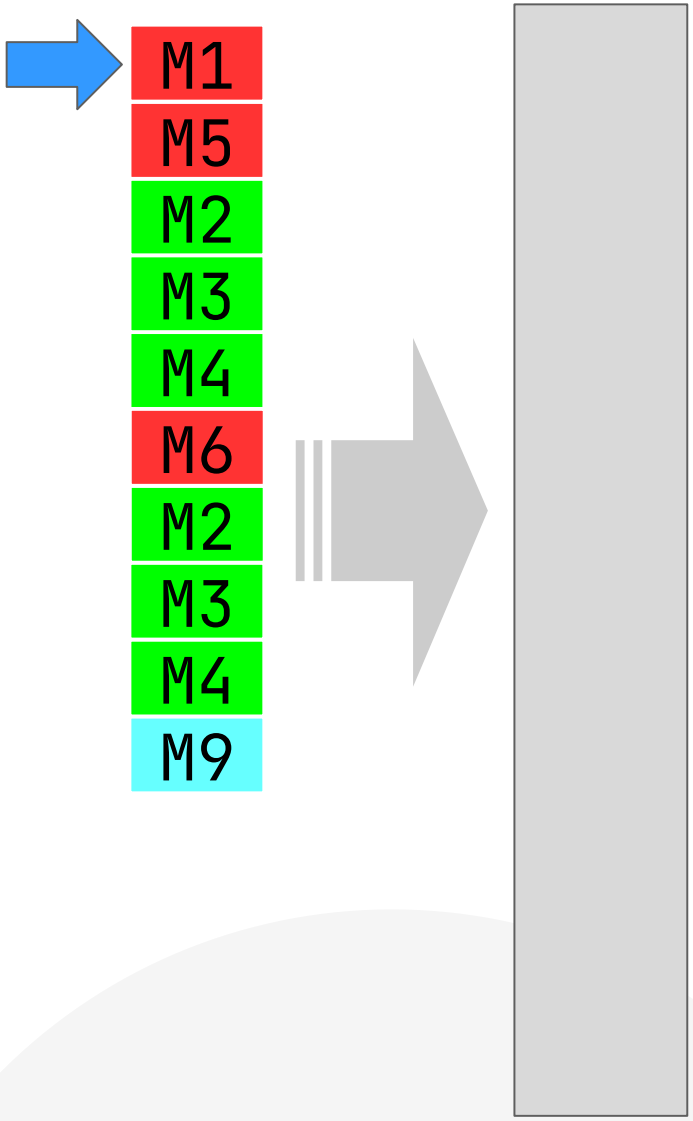
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
```

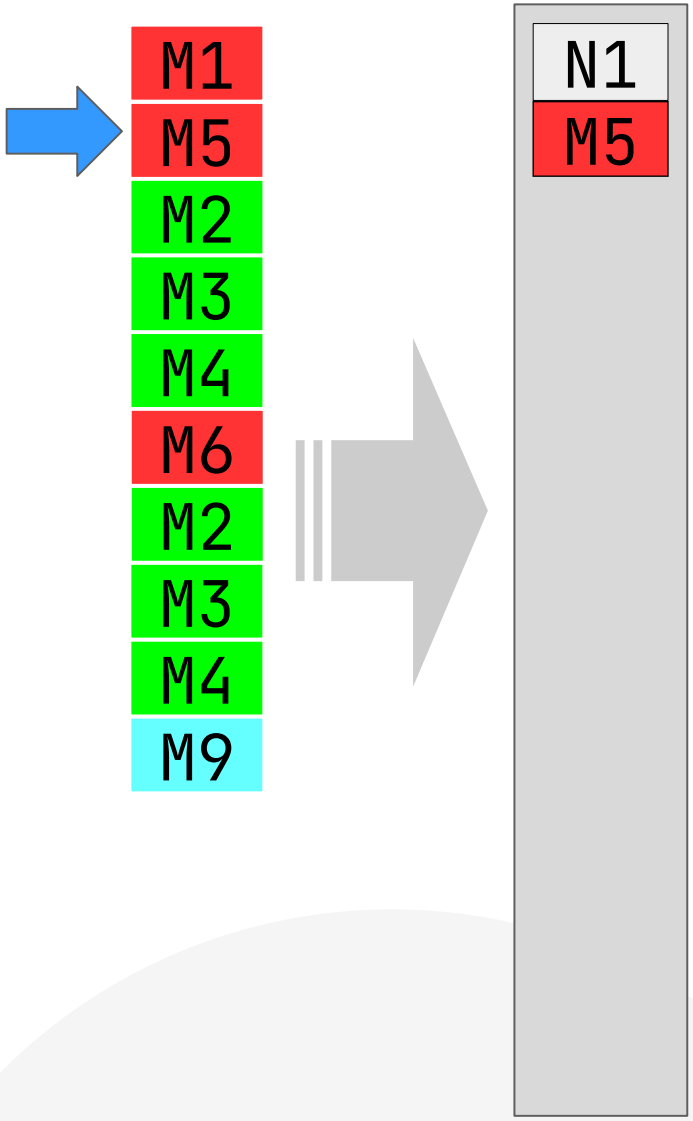
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
```

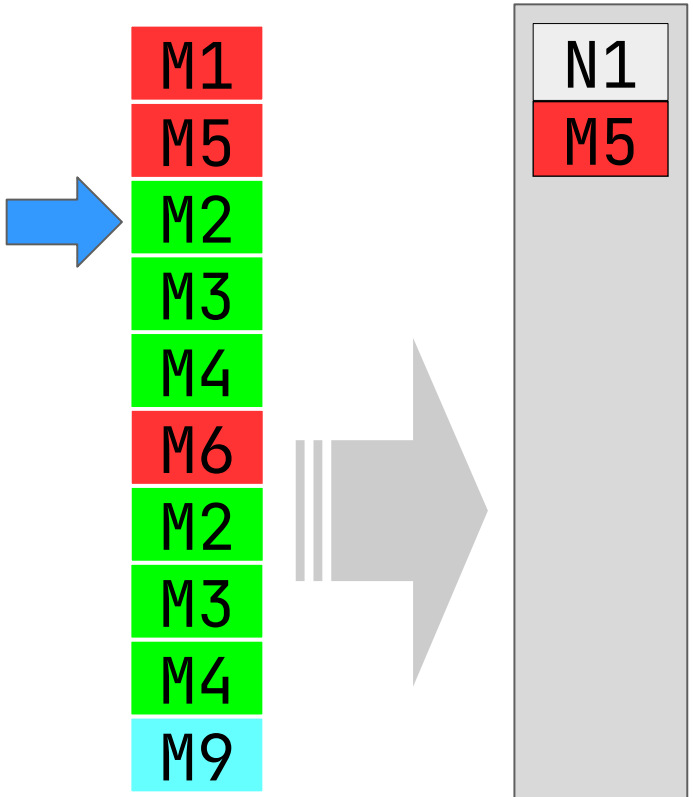
Сжимаем



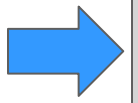
Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
```

Сжимаем

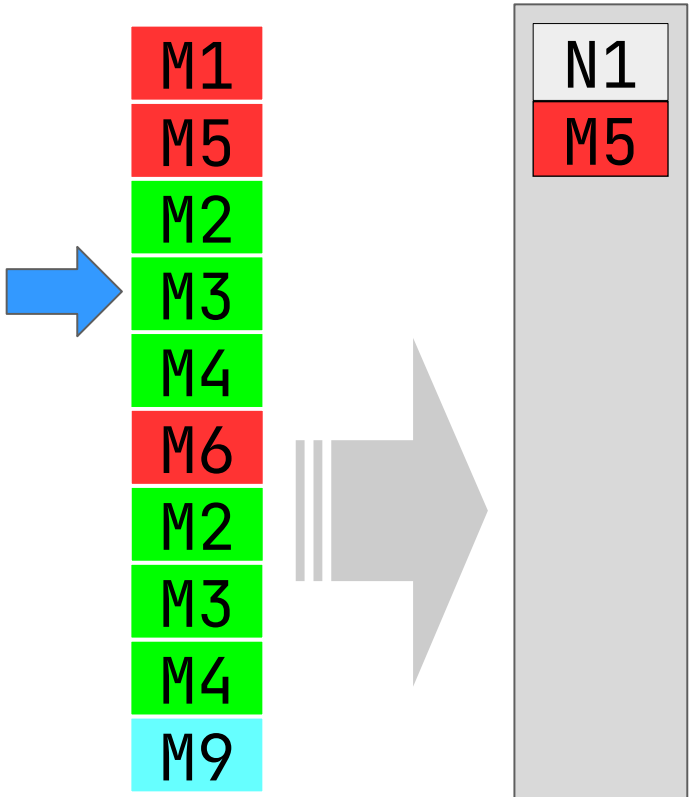


Словарь

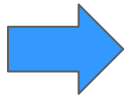


```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
```

Сжимаем

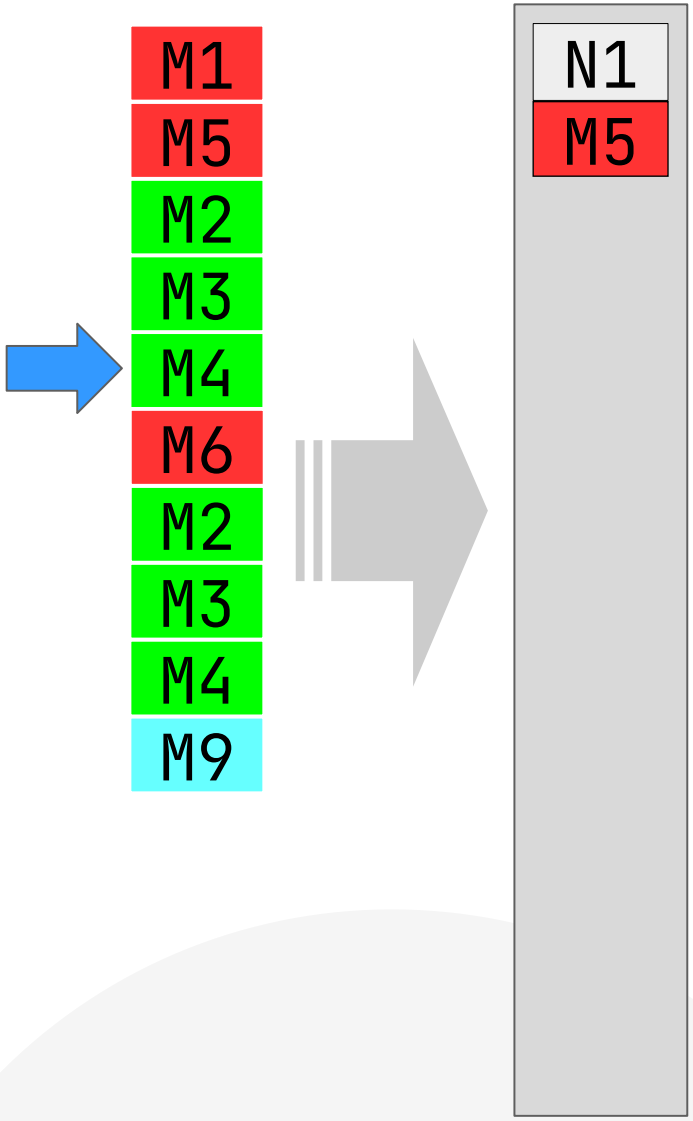


Словарь



```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
```

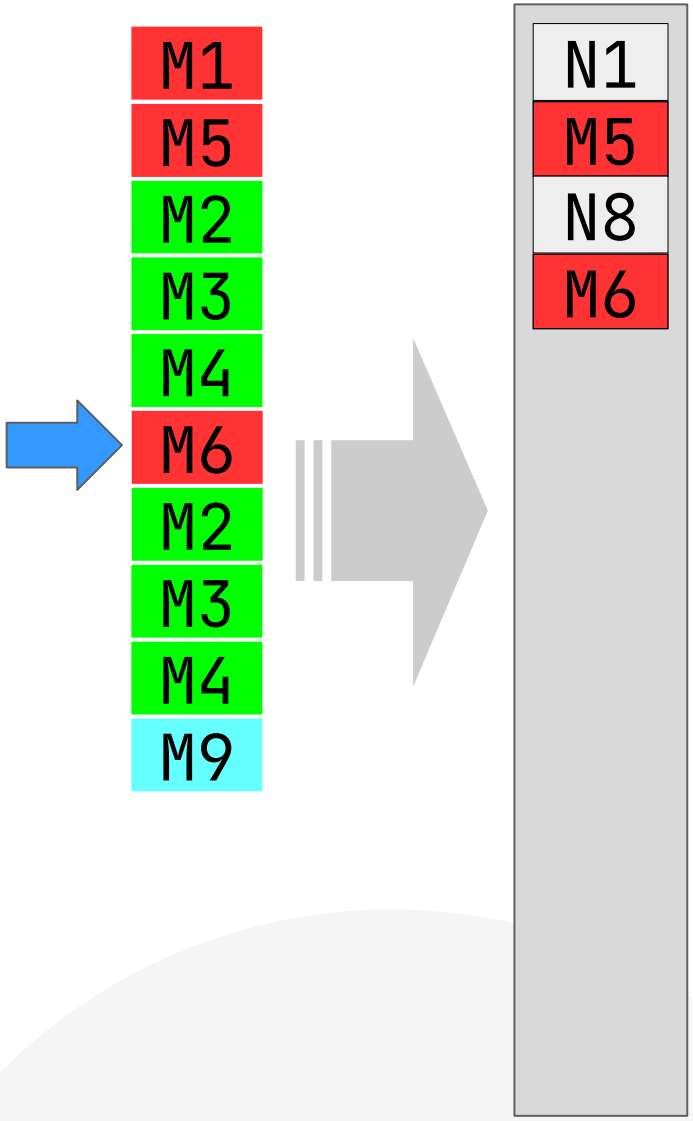
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
```

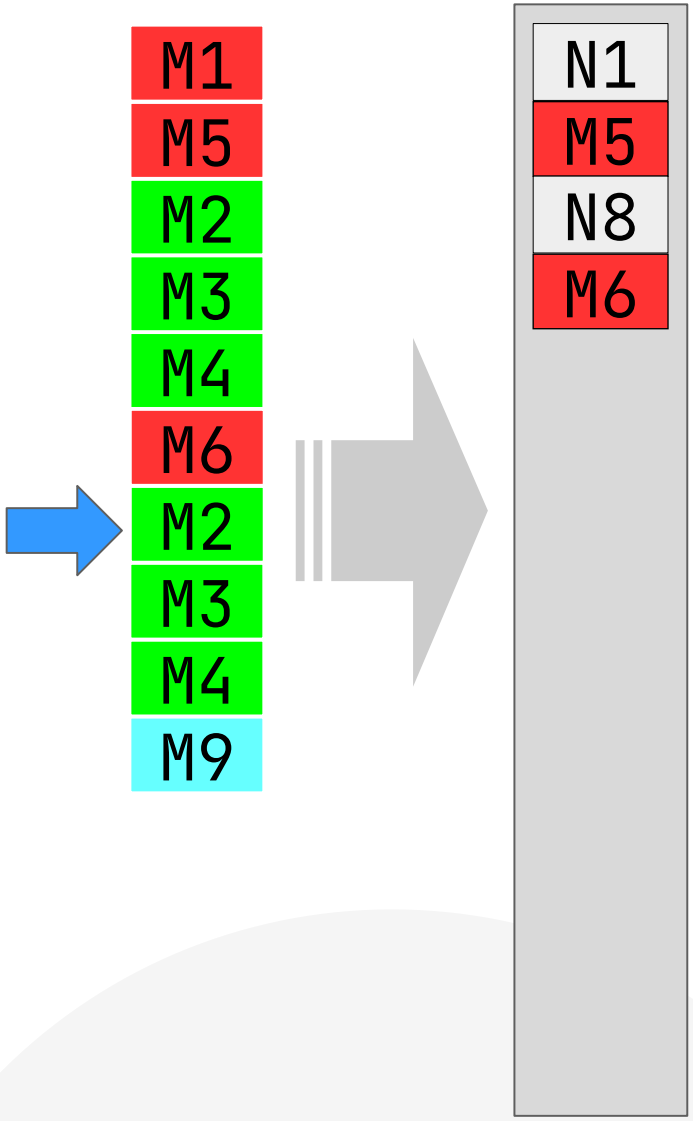
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
```

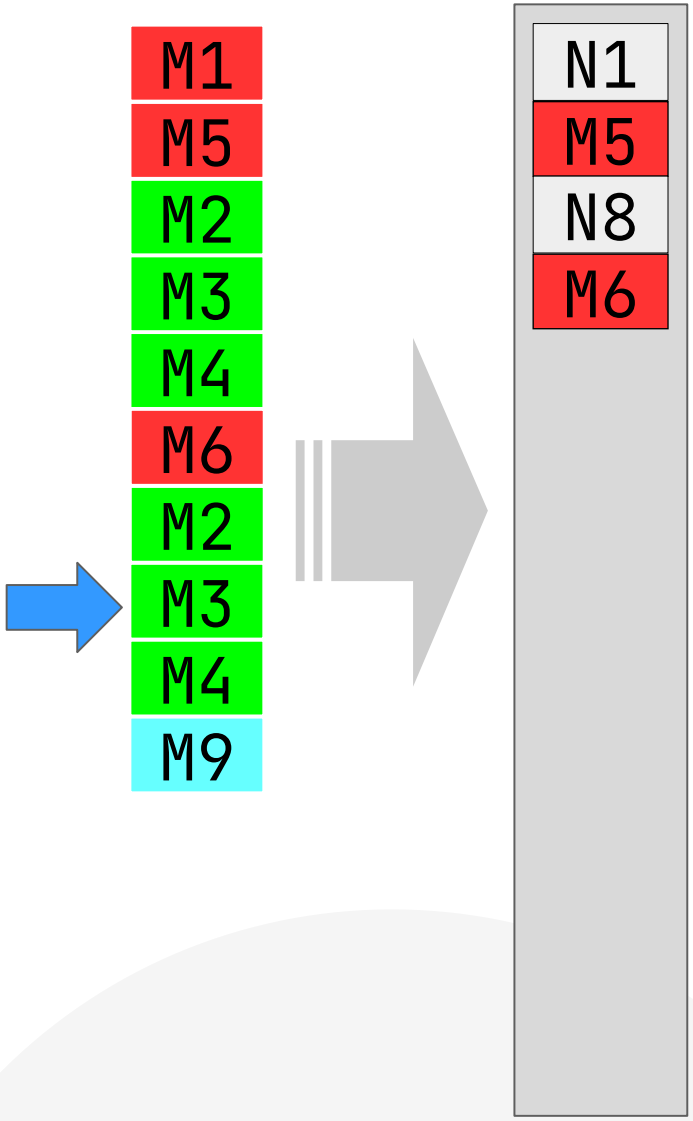
Сжимаем



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
```

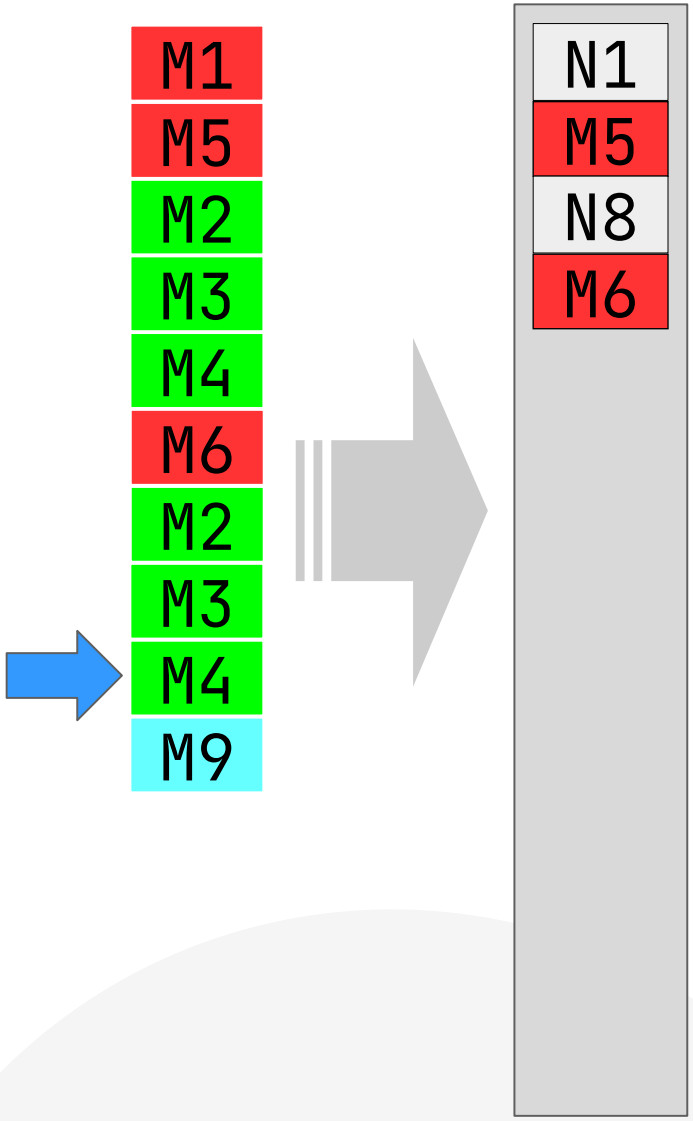
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
```

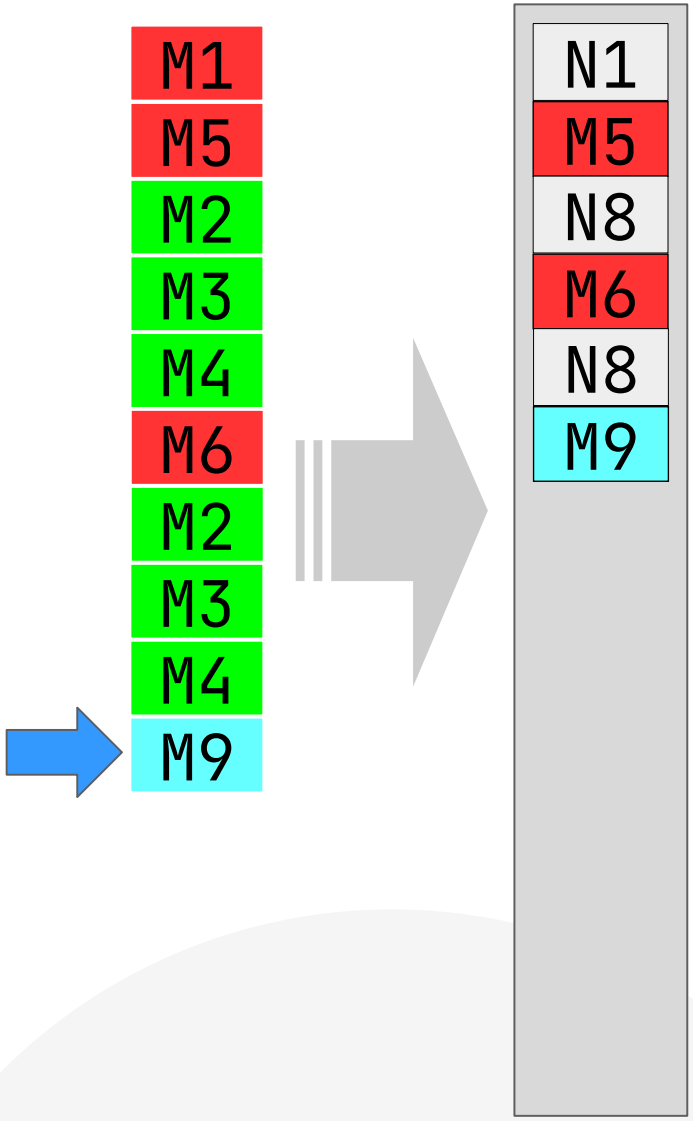
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
```

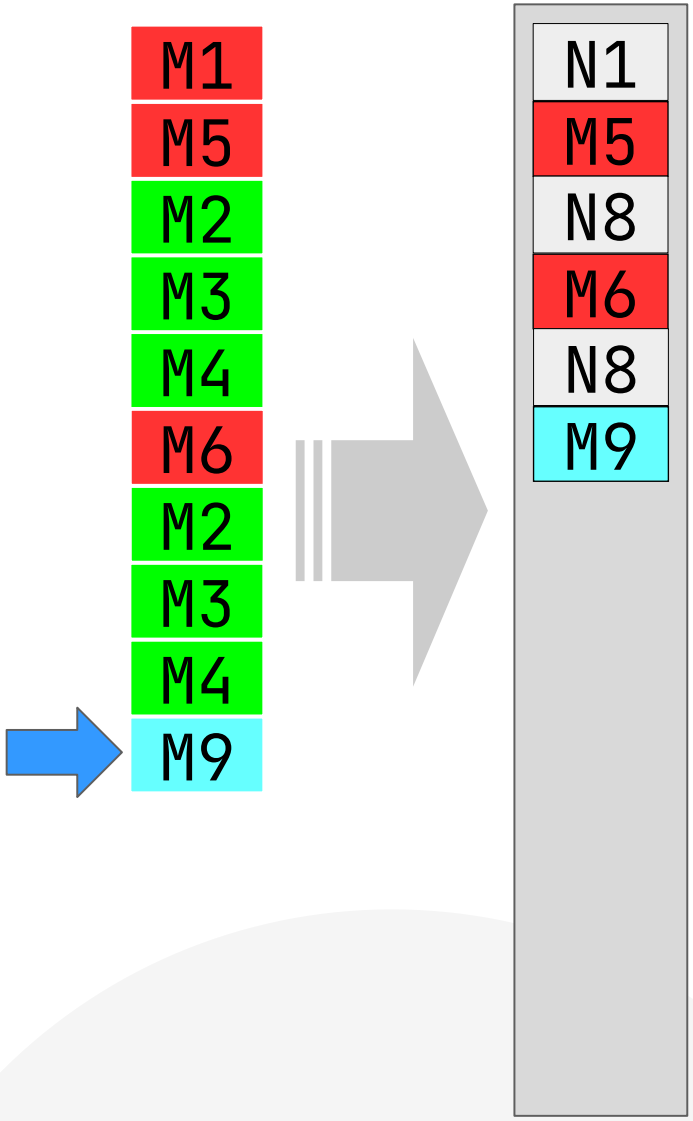
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

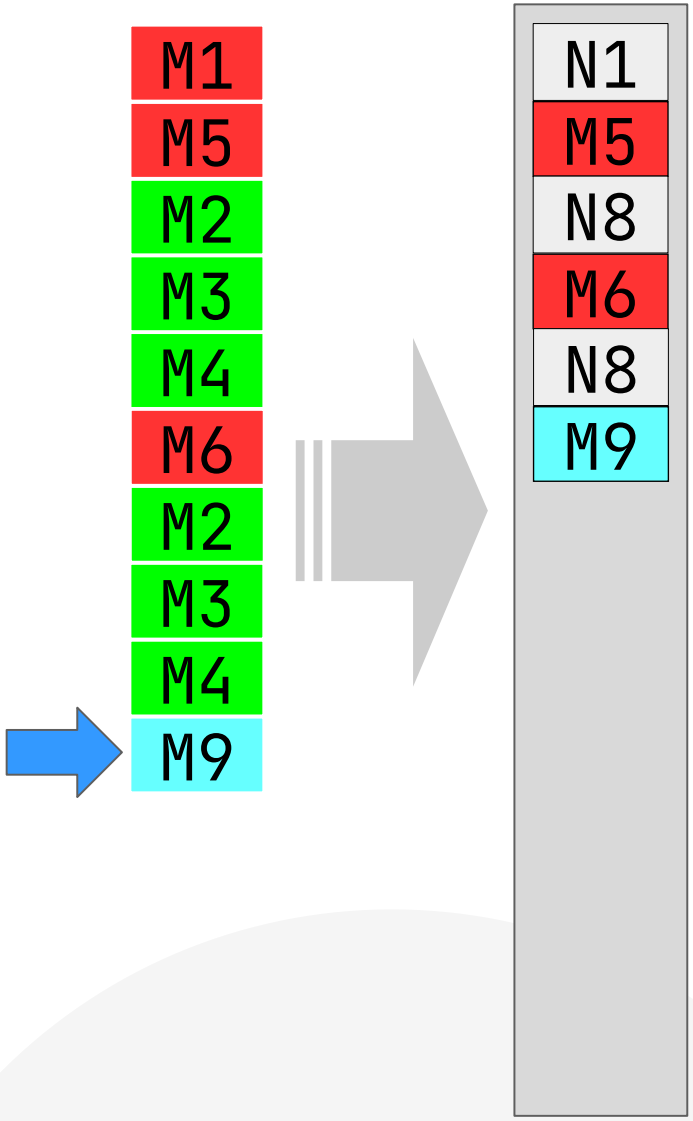
Сжимаем



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Сжимаем

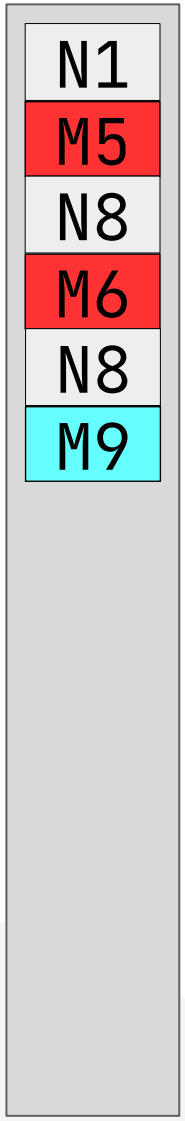


LZ78

Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

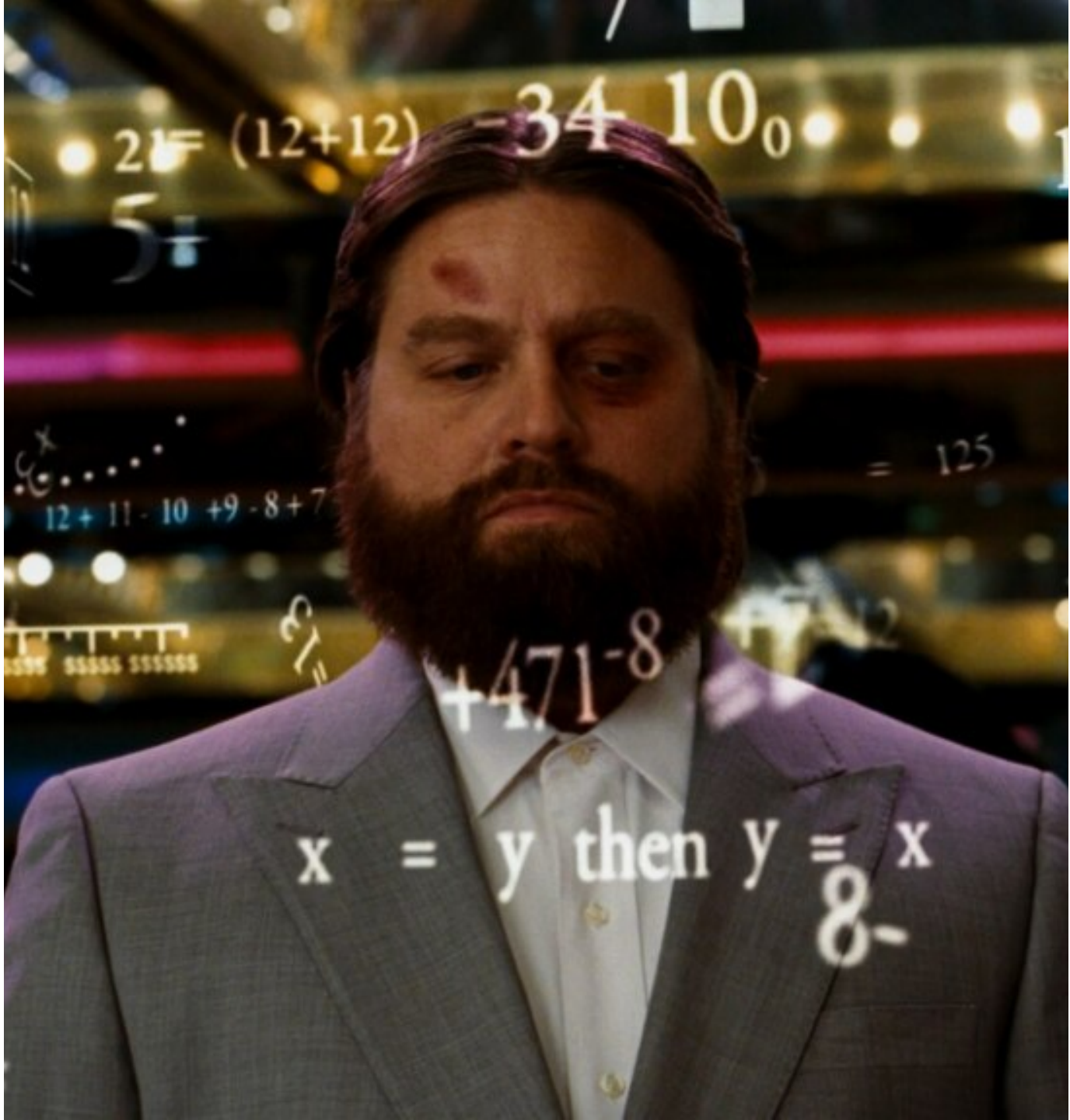
Сжимаем



HTML

Готовим данные

Кодируем числа



Кодируем числа

N102
M504
N898
M602
N835
M991

HTML

Кодируем числа

N102
M504
N898
M602
N835
M991


```
<pre>
102\n
504\n
898\n
602\n
835\n
991\n
</pre>
```

Кодируем числа

N102
M504
N898
M602
N835
M991


```
<pre>
102\n
504\n
898\n
602\n
835\n
991\n
</pre>
```

4 байта

Кодируем числа: HEX

N102
M504
N898
M602
N835
M991


```
<pre>
66\n
1F8\n
382\n
25A\n
343\n
3DF\n
</pre>
```

3 байта

4 байта

4 байта

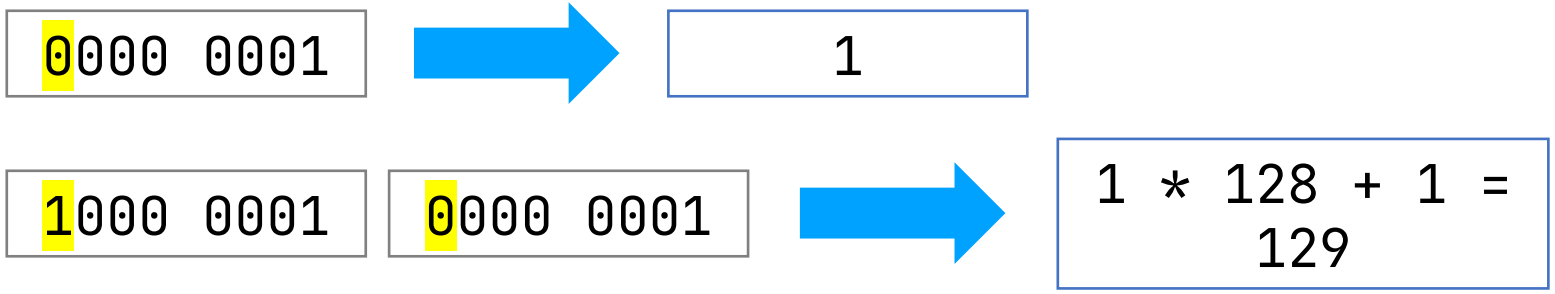
4 байта

4 байта

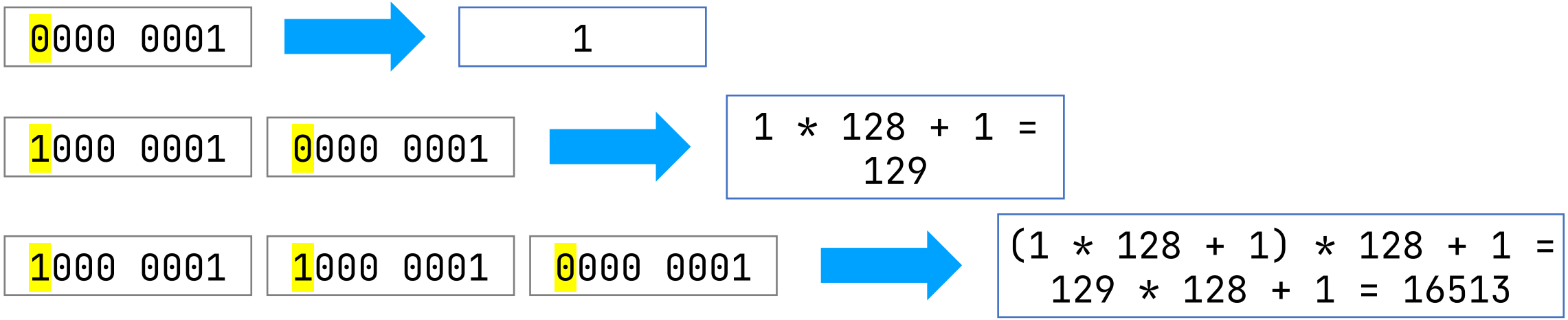
4 байта

23 байта

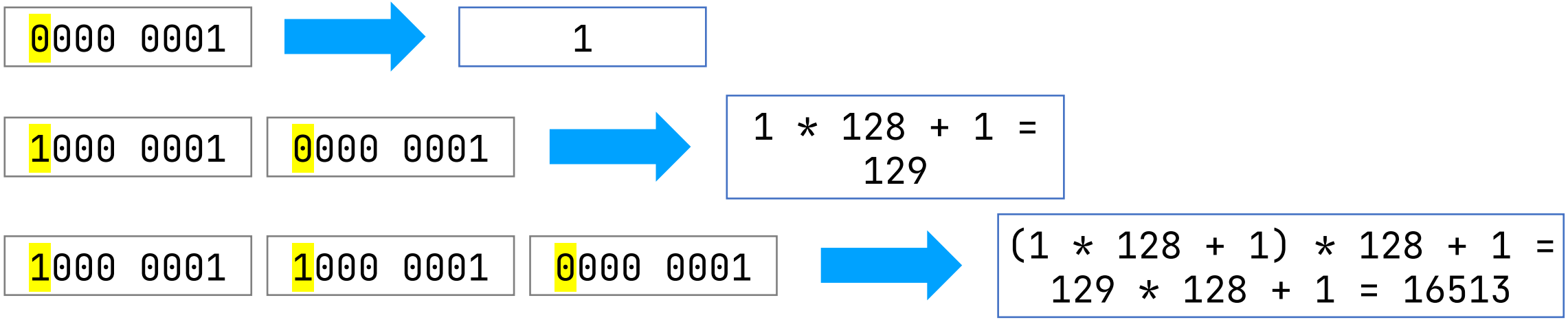
Кодируем числа: VarInt



Кодируем числа: VarInt

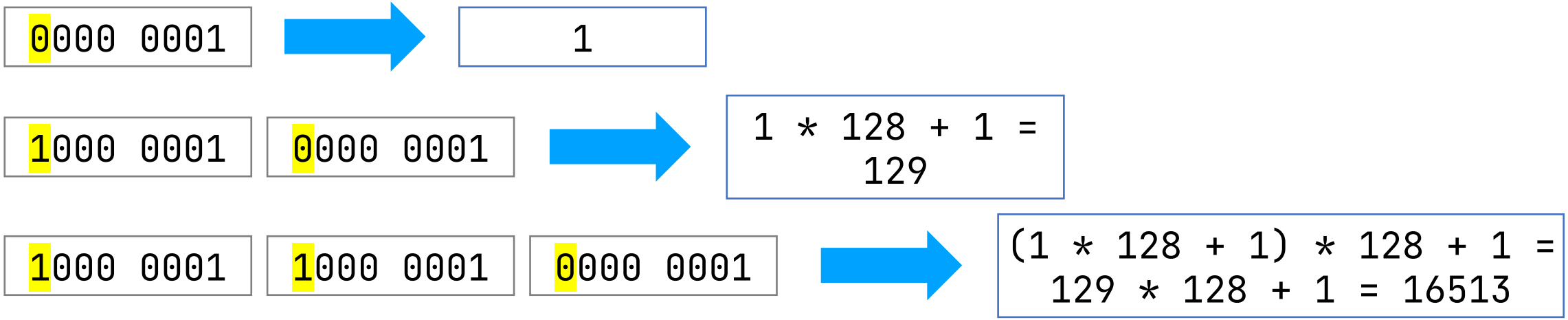


Кодируем числа: VarInt



Q: Где Хаффман?

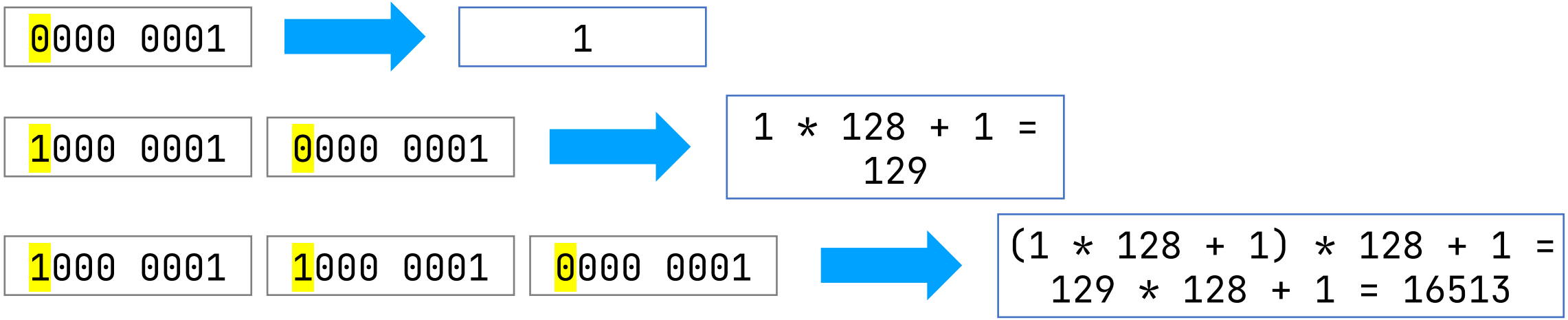
Кодируем числа: VarInt



Q: Где Хаффман?

Q: А арифметическое кодирование?

Кодируем числа: VarInt



Q: Где Хаффман?

Q: А арифметическое кодирование?

Q: Асимметрические системы счисления?

VarInt в дикой природе

```
public void writeVar(int v) {
    while (v ⩾ 128) {
        int b = 128 + v % 128;
        nextByte(b);
        v ⊨ 128;
    }
    nextByte(v);
}
```

VarInt в дикой природе

```
public void writeVar(int v) {
    while (v ⩾ 128) {
        int b = 128 + v % 128;
        nextByte(b);
        v ⊨ 128;
    }
    nextByte(v);
}
```

VarInt в HTML

```
<pre>0-127</pre>
```

1) \0
2) \r
3) &
4) <
5) >

Итого: 123 значения

VarInt в дикой природе

```
public void writeVar(int v) {
    while (v ⩾ 128) {
        int b = 128 + v % 128;
        nextByte(b);
        v ⊨ 128;
    }
    nextByte(v);
}
```

VarInt в HTML

```
public void writeVar(int v) {
    while (v >= 61) {
        int b = 61 + v % 61;
        nextByte(b);
        v /= 61;
    }
    nextByte(v);
}
```

VarInt в дикой природе

```
public void writeVar(int v) {
    while (v ⩾ 128) {
        int b = 128 + v % 128;
        nextByte(b);
        v ⫦ 128;
    }
    nextByte(v);
}
```

VarInt в HTML

```
public void writeVar(int v) {
    while (v >= 61) {
        int b = 61 + v % 61;
        nextByte(b);
        v /= 61;
    }
    nextByte(v);
}
```

VarInt

- Подходит, когда мало значащих бит

VarInt

- Подходит, когда мало значащих бит
- Гранулярность - 1 байт

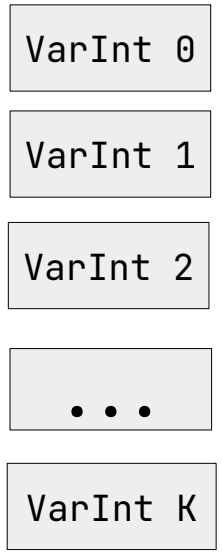
VarInt

- Подходит, когда мало значащих бит
- Гранулярность - 1 байт
- Теряем 1 бит каждого байта

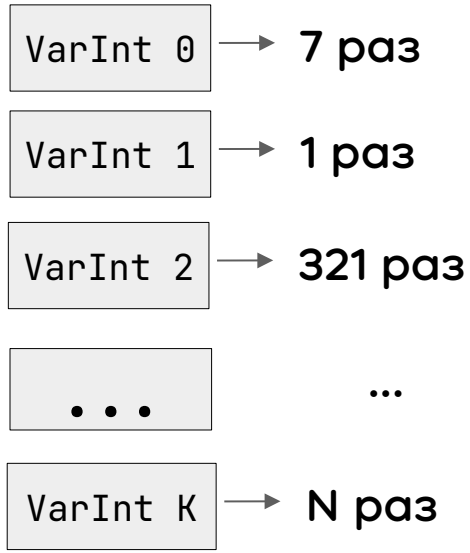
VarInt

- Подходит, когда мало значащих бит
- Гранулярность - 1 байт
- Теряем 1 бит каждого байта

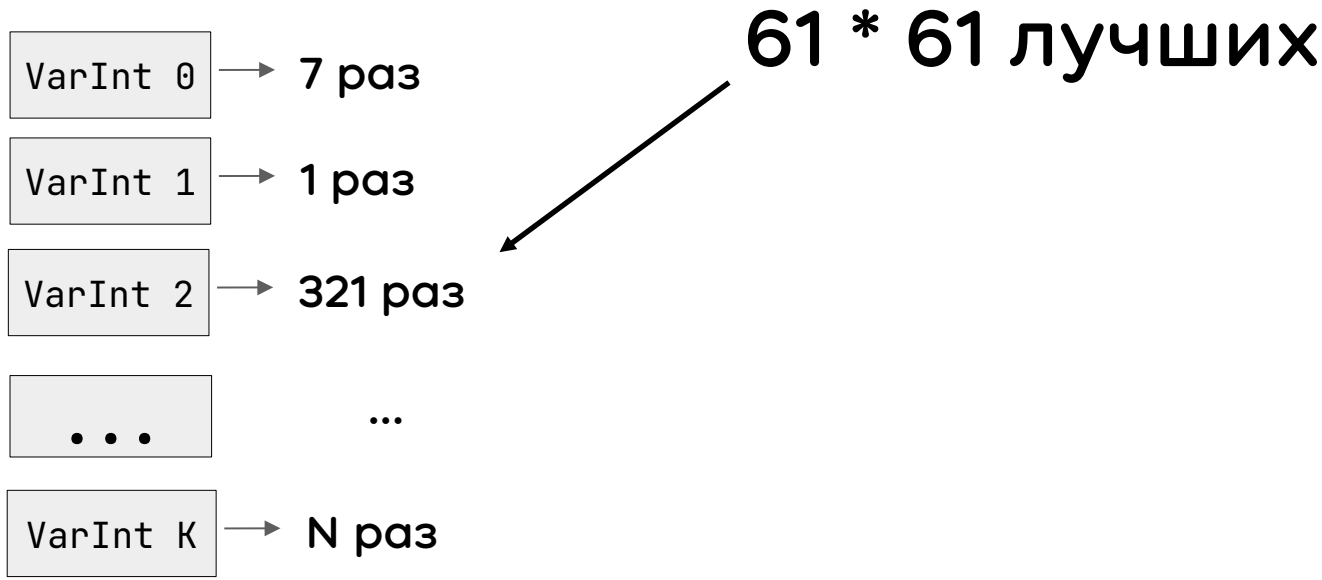
Словарь синонимов



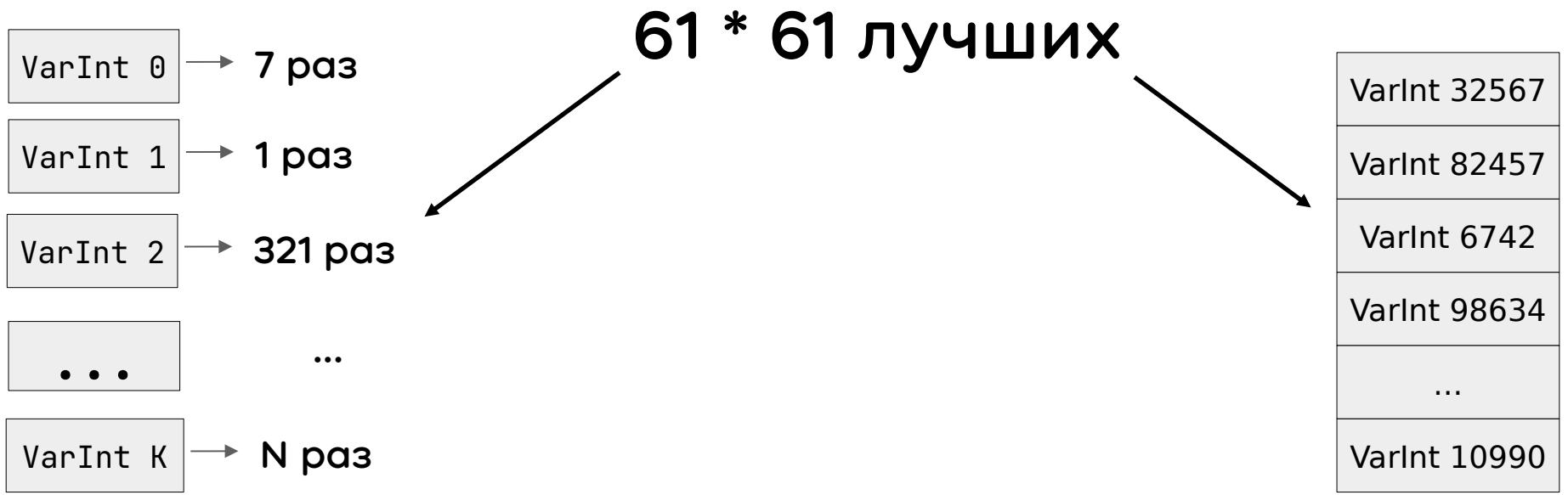
Словарь синонимов



Словарь синонимов



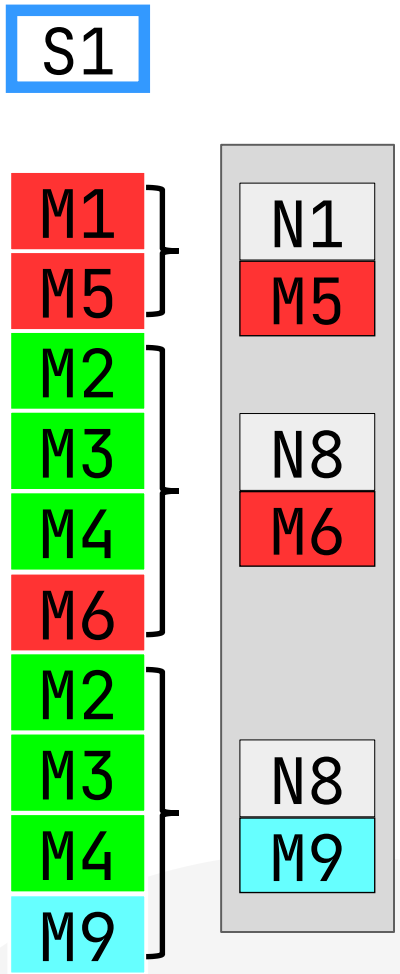
Словарь синонимов



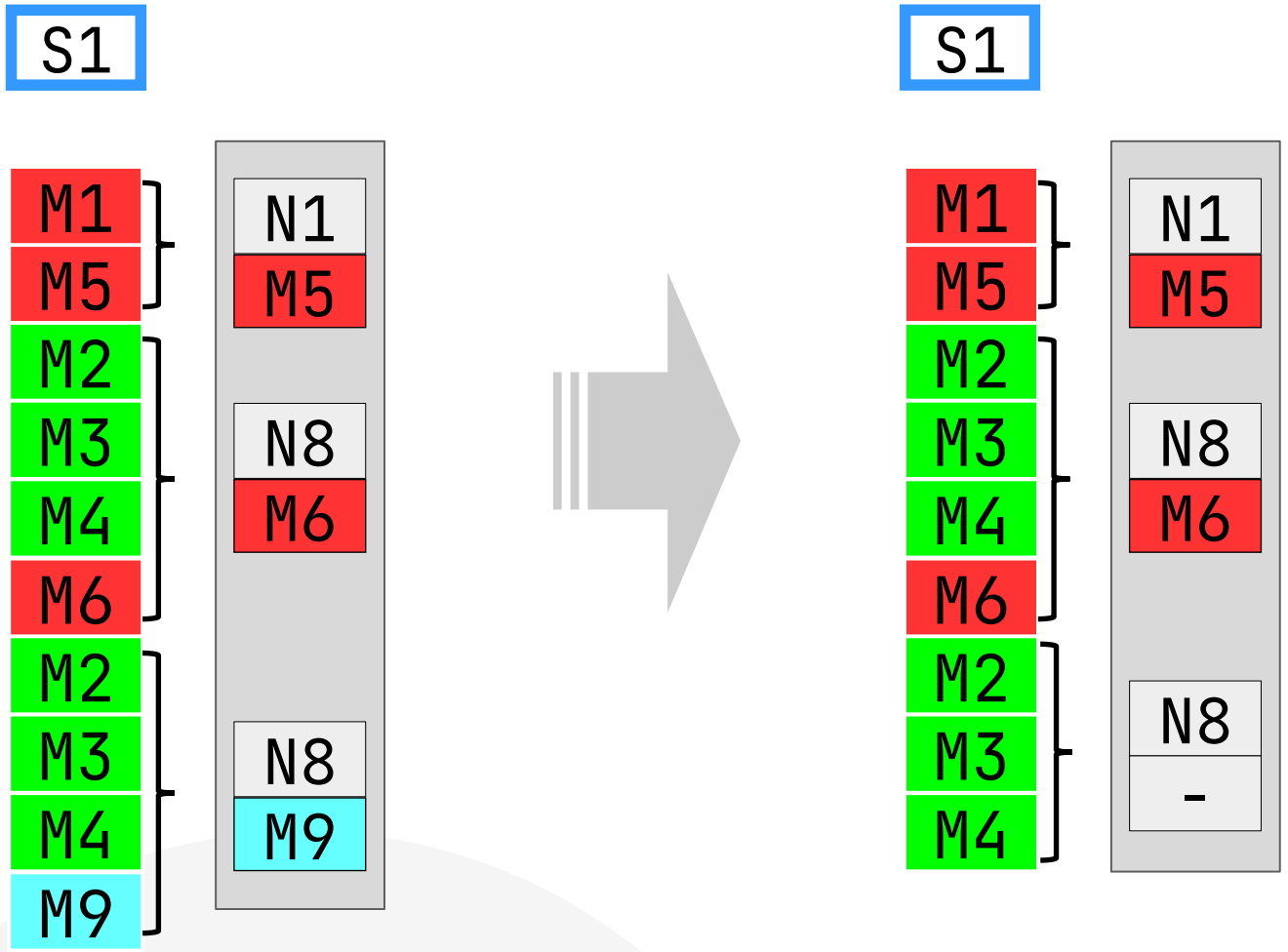
Готовим данные

Промежуточный итог

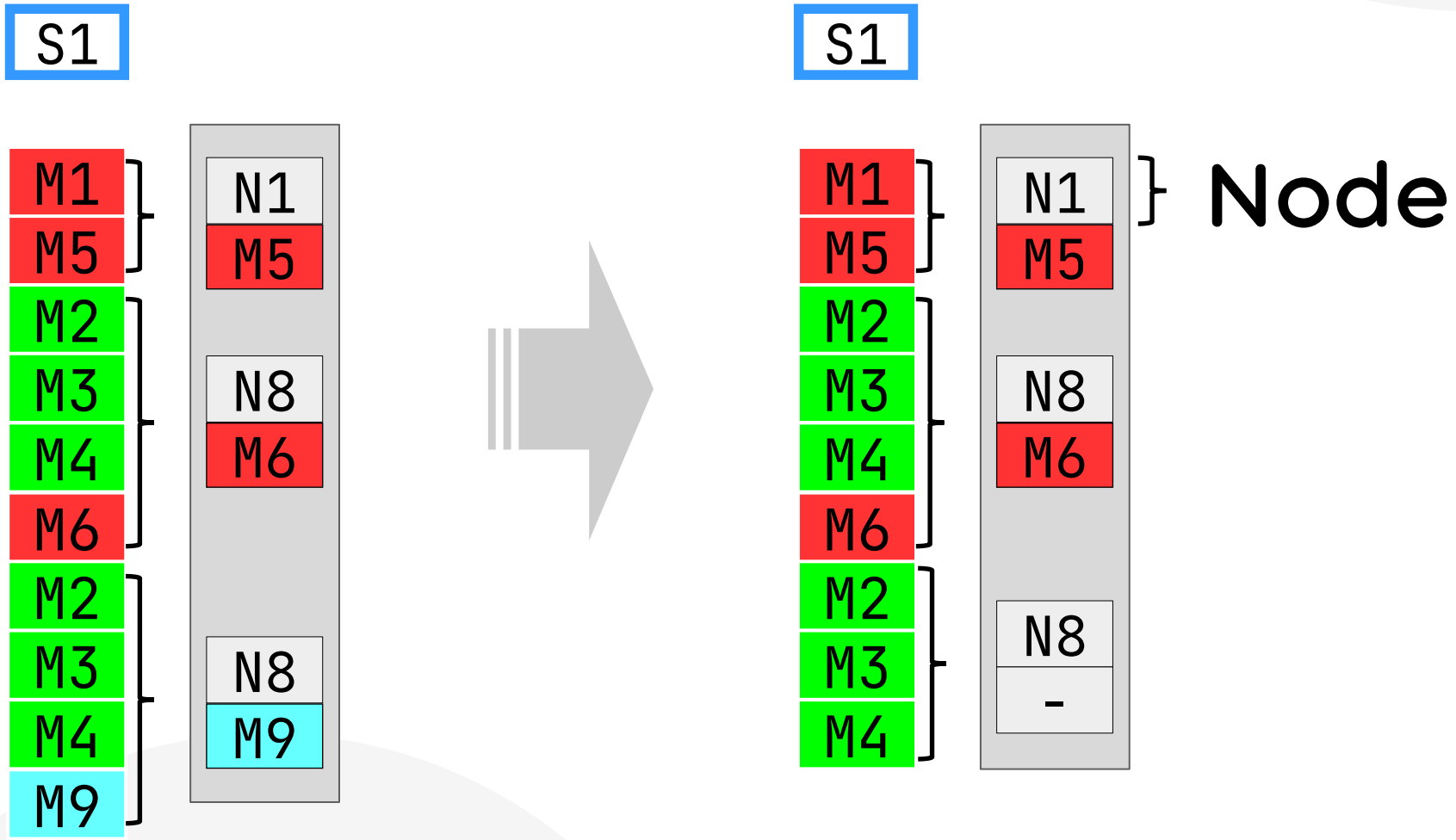
Промежуточный итог



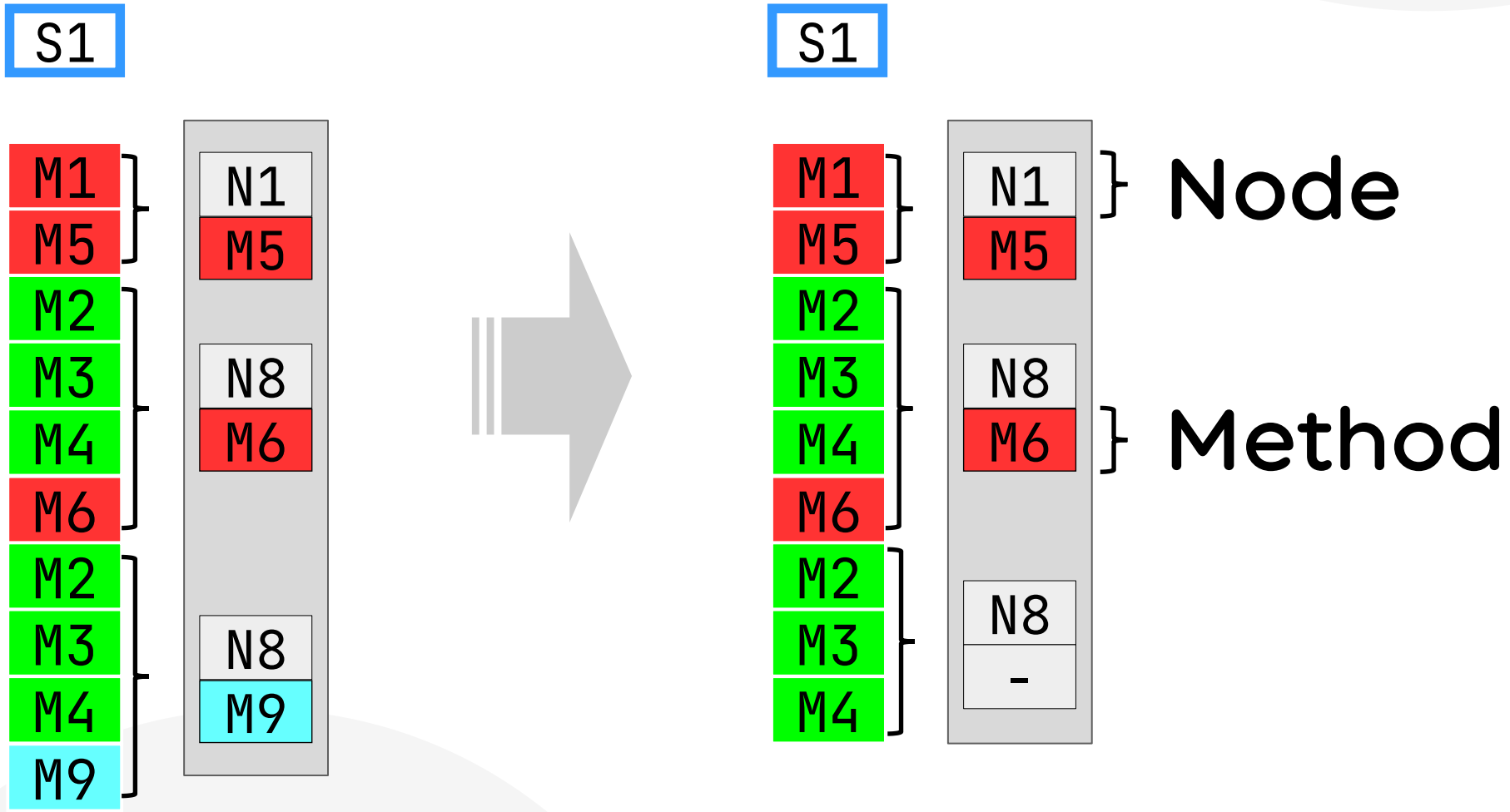
Промежуточный итог



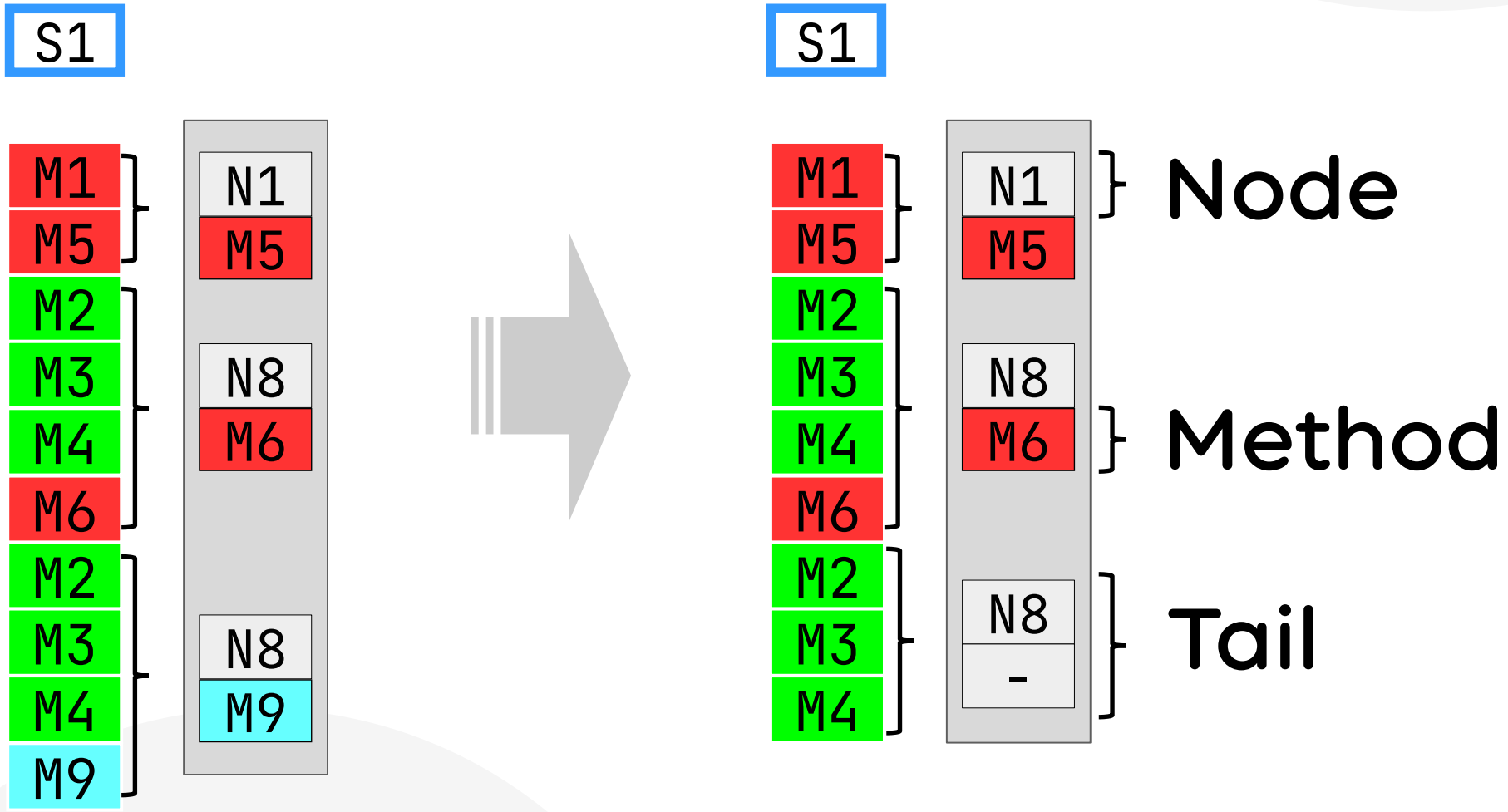
Промежуточный итог



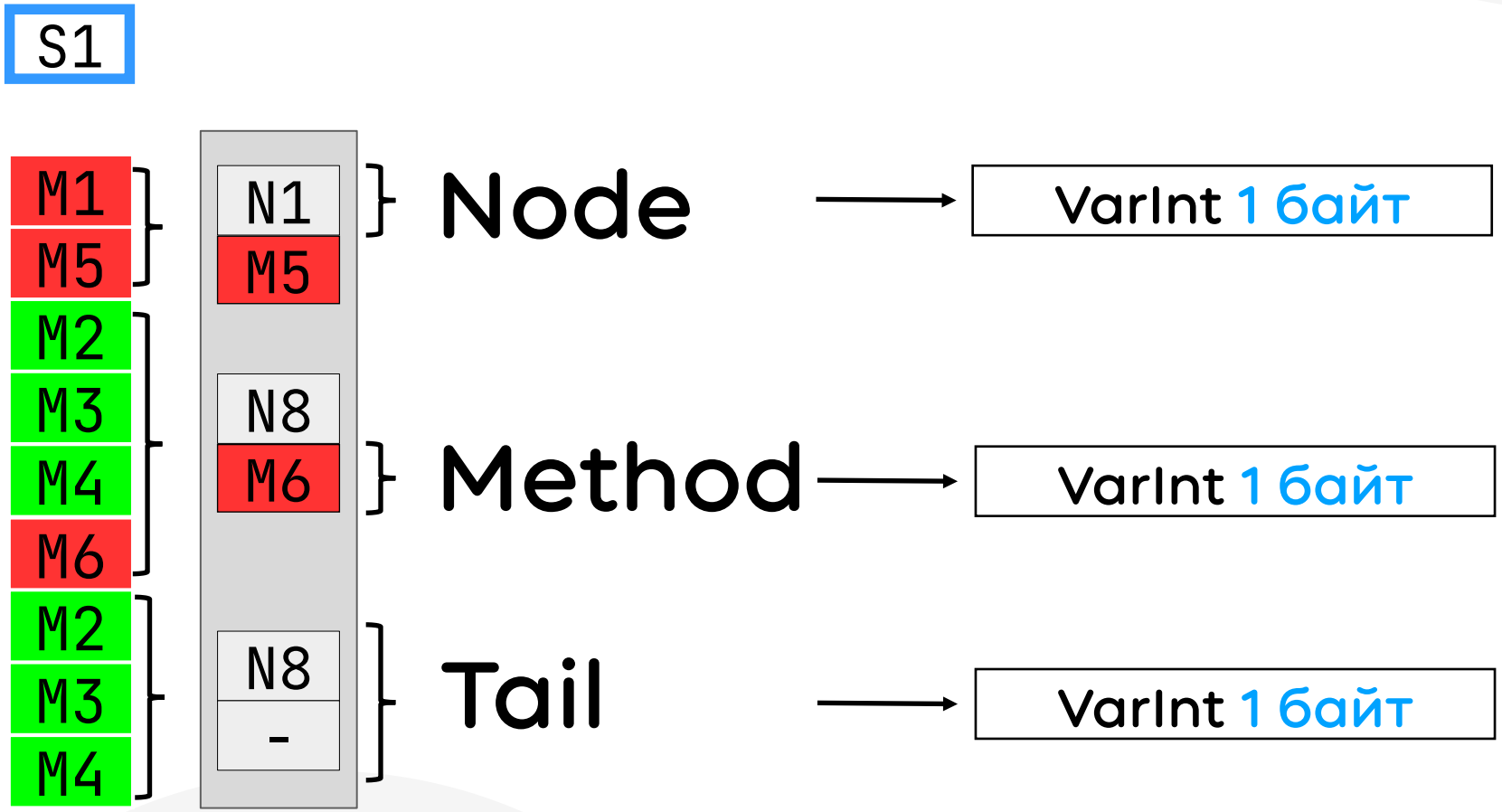
Промежуточный итог



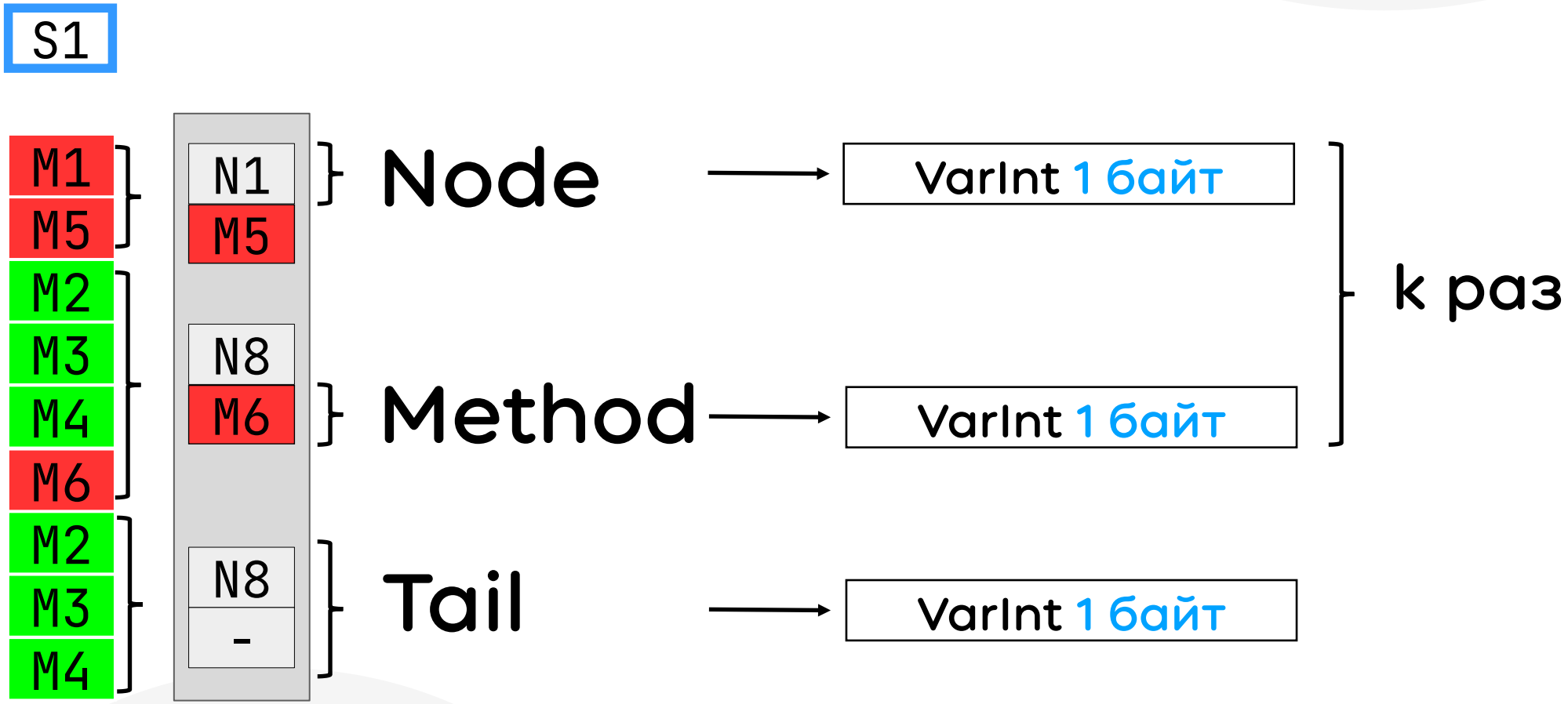
Промежуточный итог



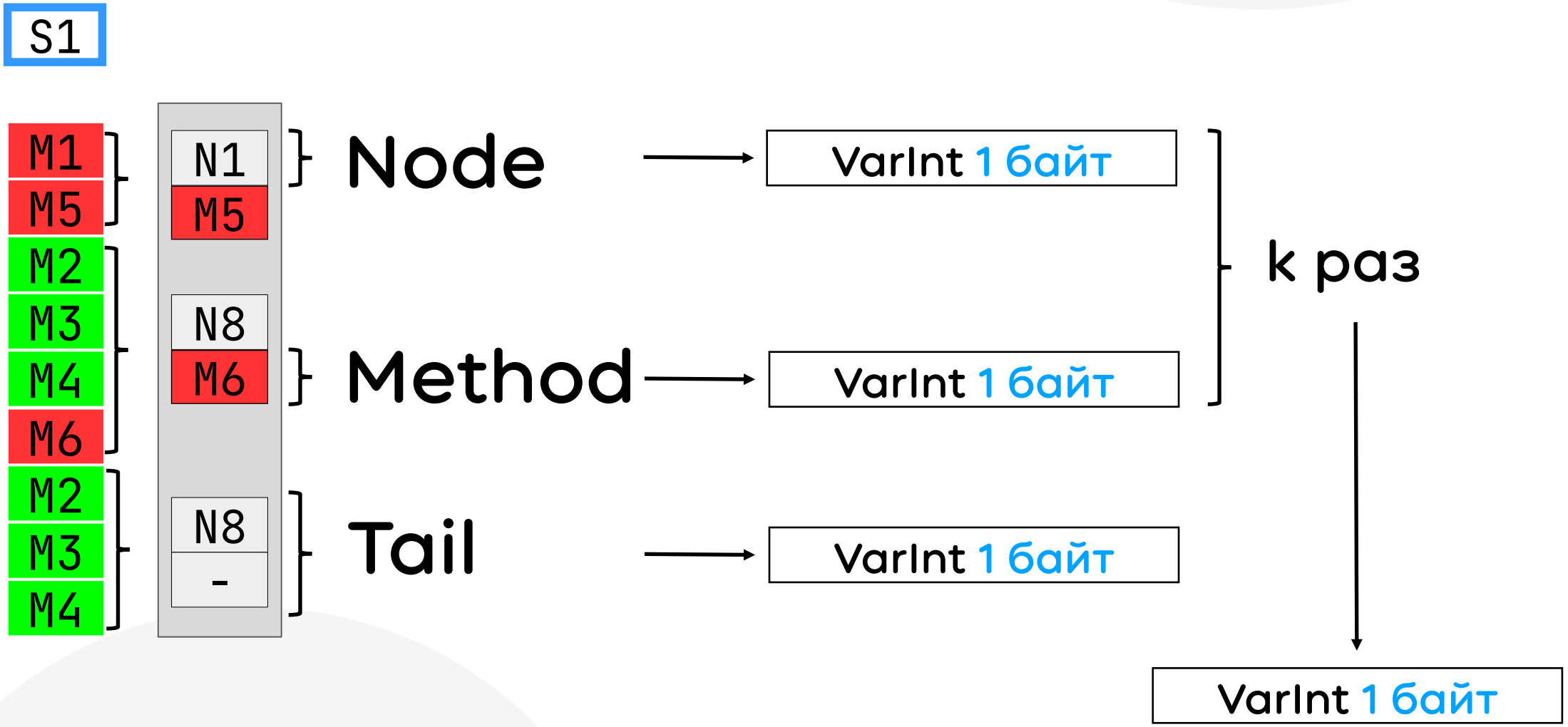
Промежуточный итог



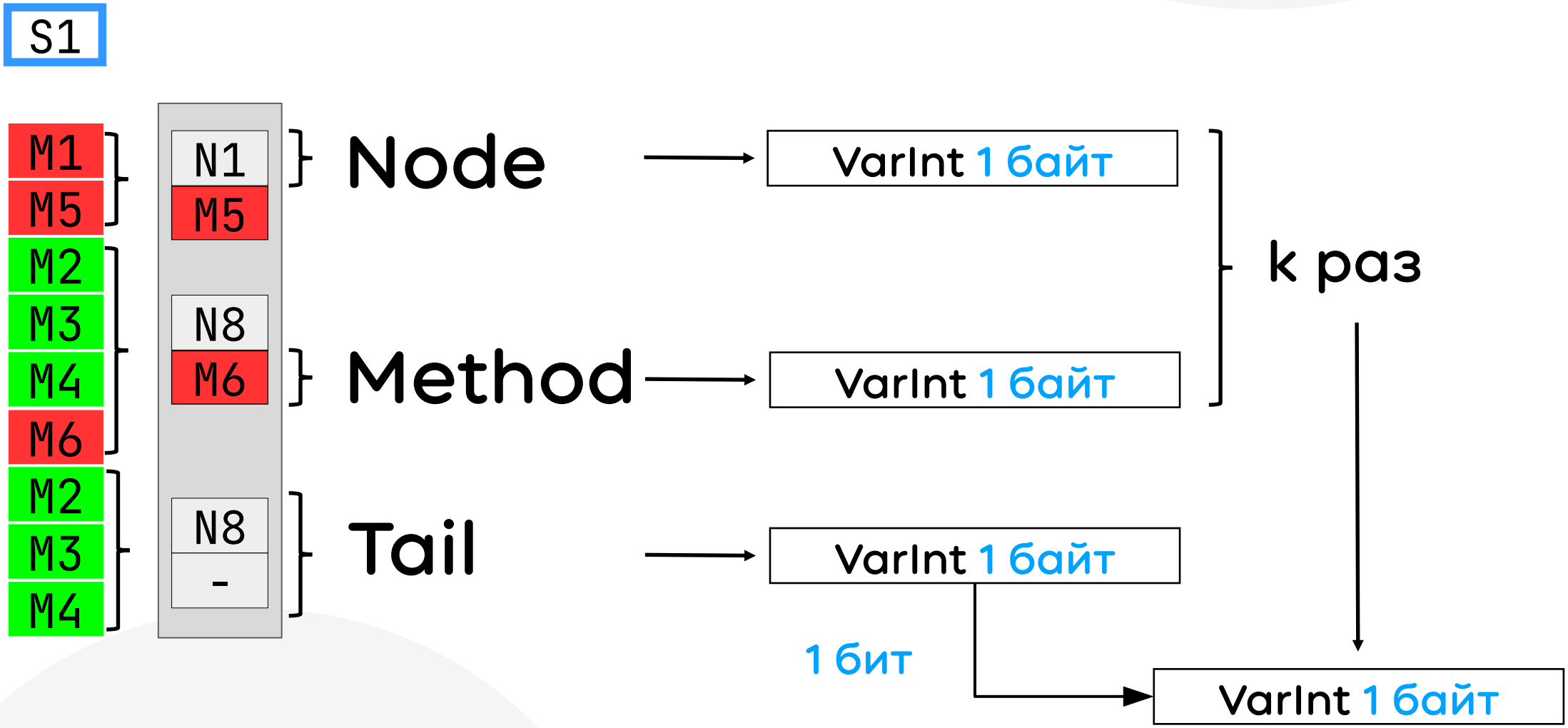
Промежуточный итог



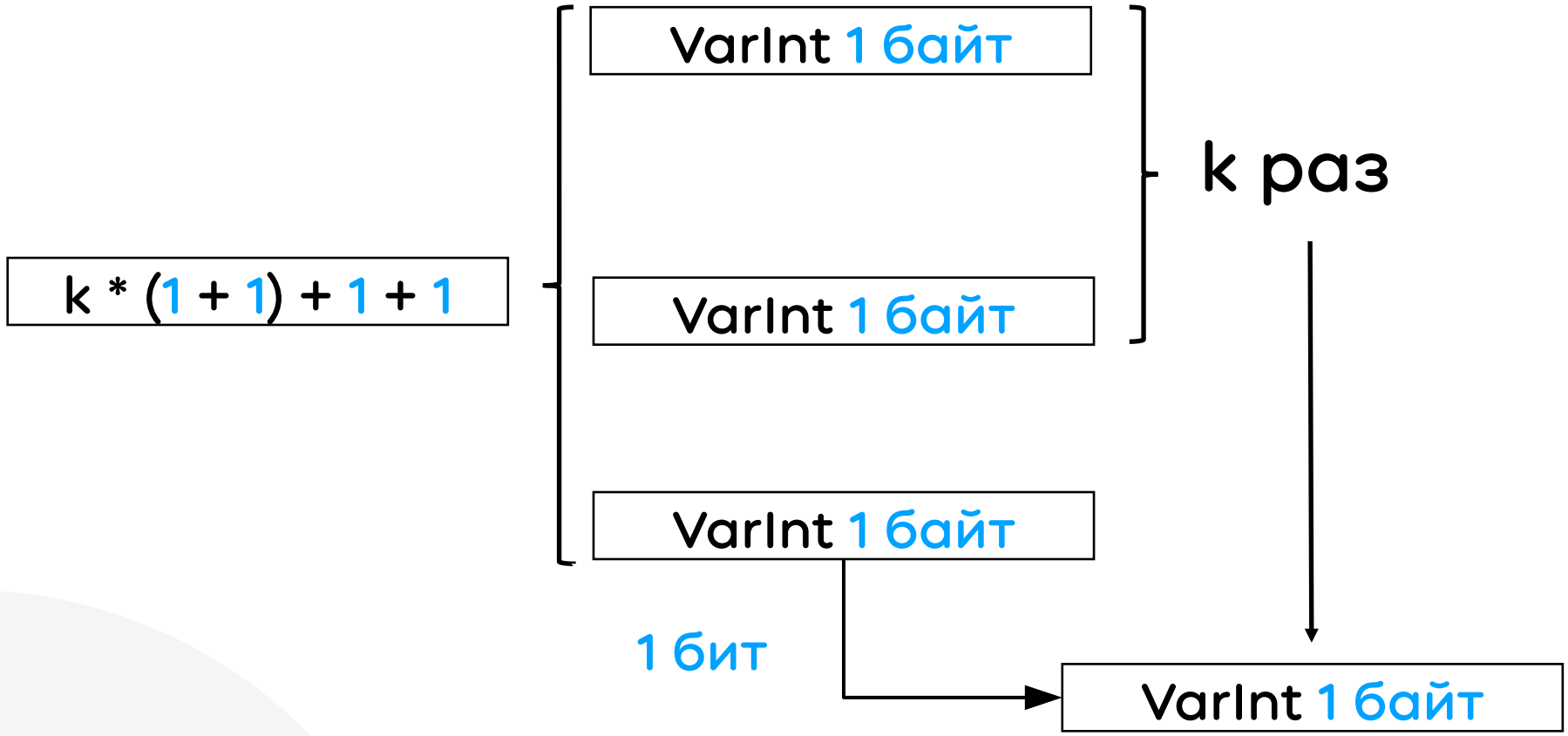
Промежуточный итог



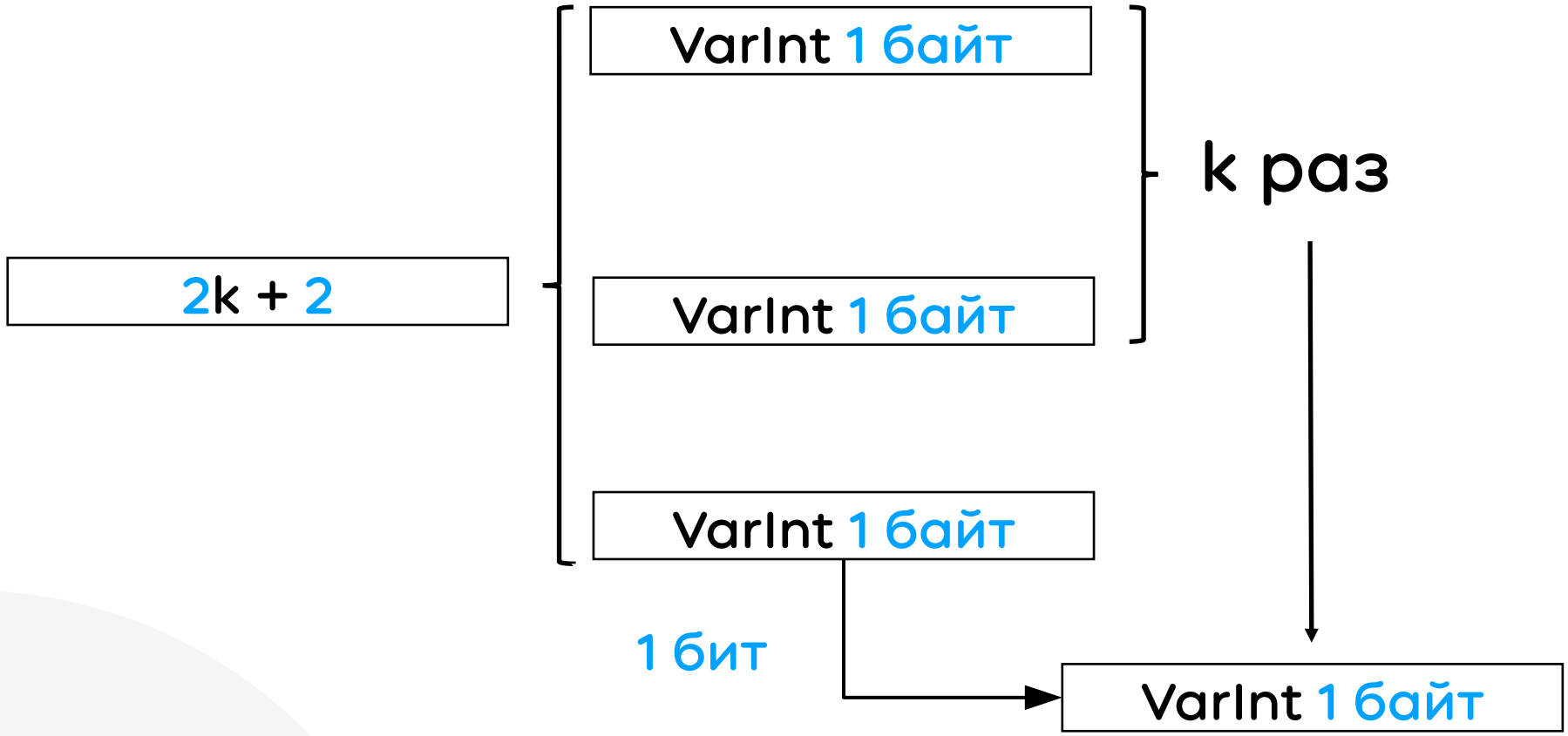
Промежуточный итог



Промежуточный итог



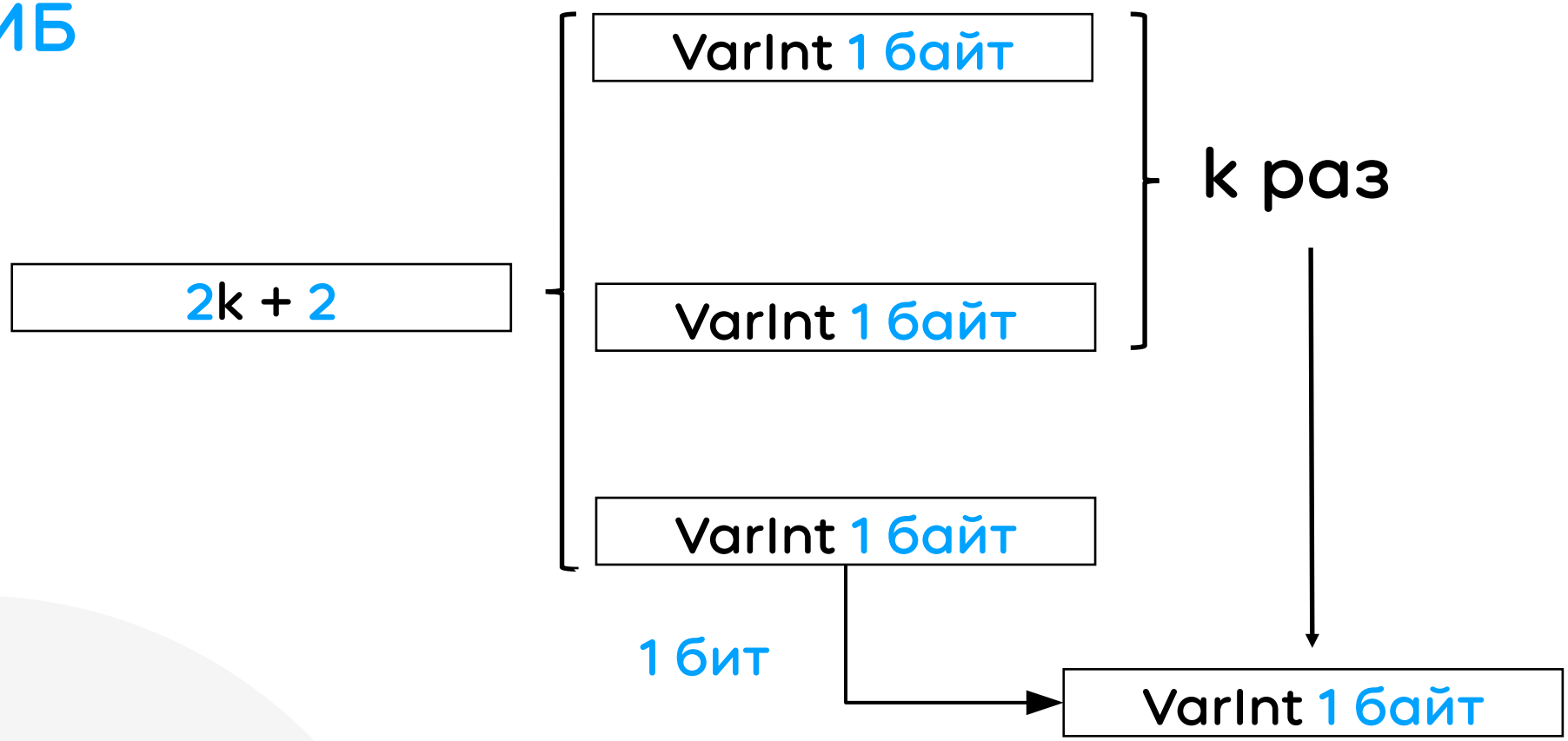
Промежуточный итог



Промежуточный итог

Alloc: 78 млн сэмплов

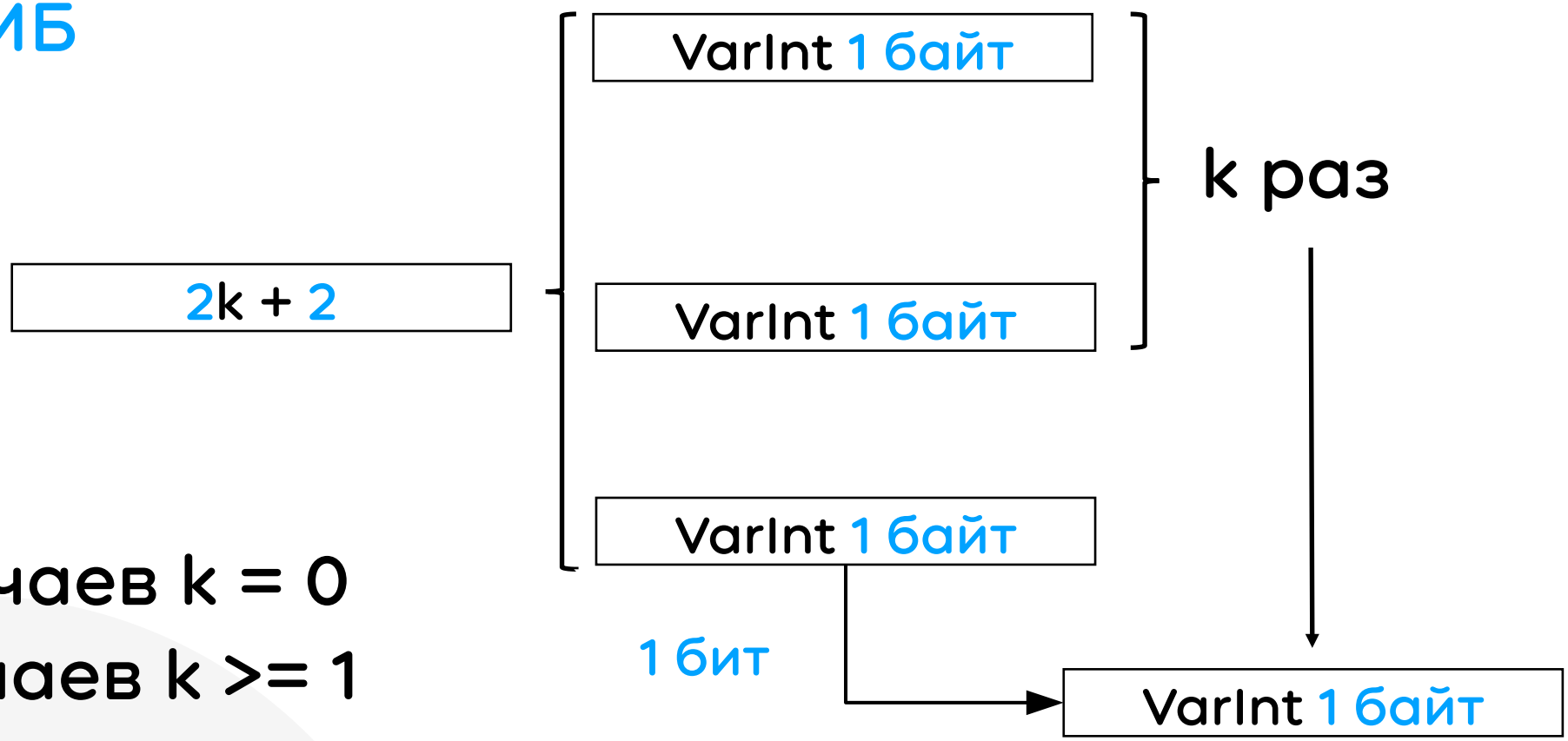
Цель: 170 МБ



Промежуточный итог

Alloc: 78 млн сэмплов

Цель: 170 МБ



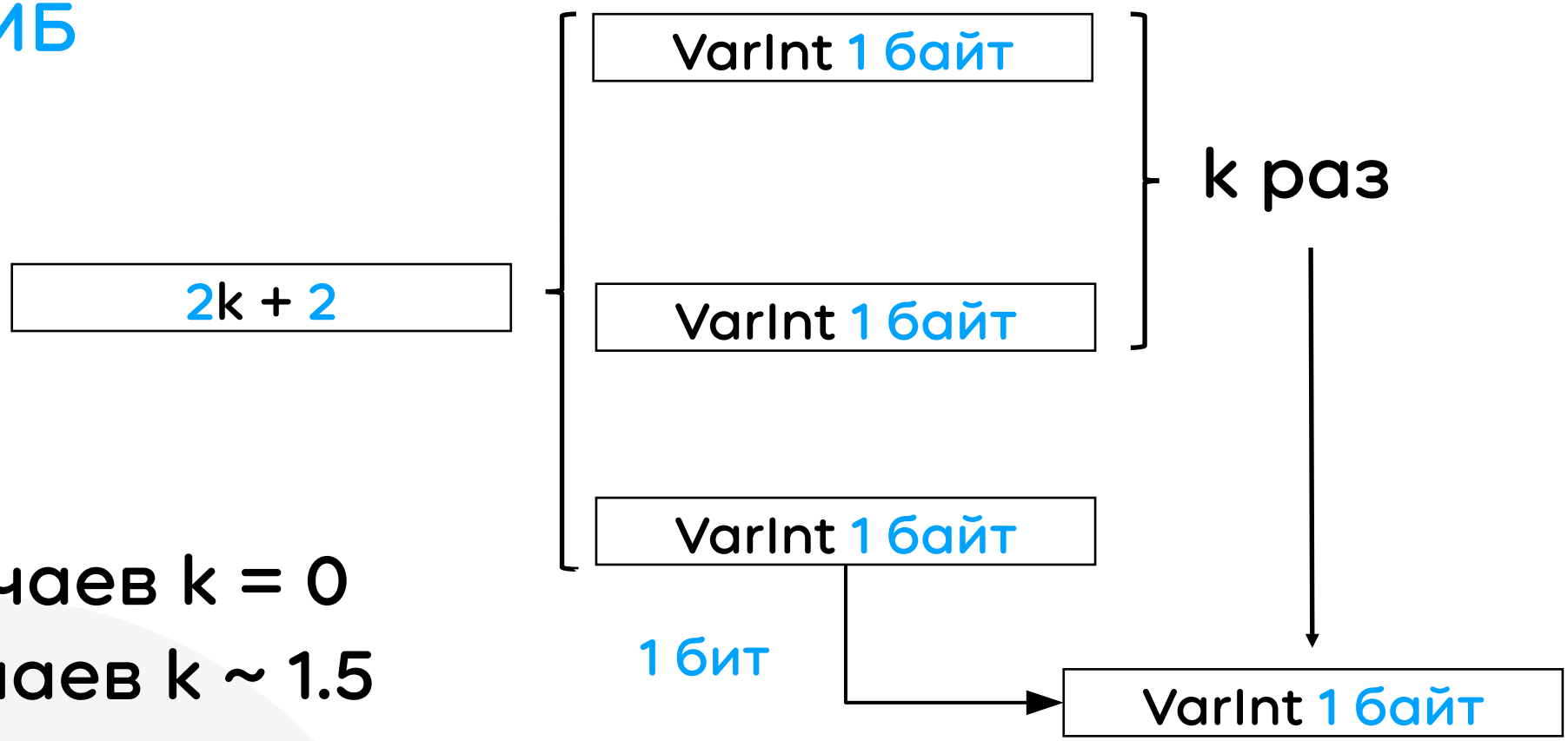
В 90% случаев k = 0

В 10% случаев k >= 1

Промежуточный итог

Alloc: 78 млн сэмплов

Цель: 170 МБ



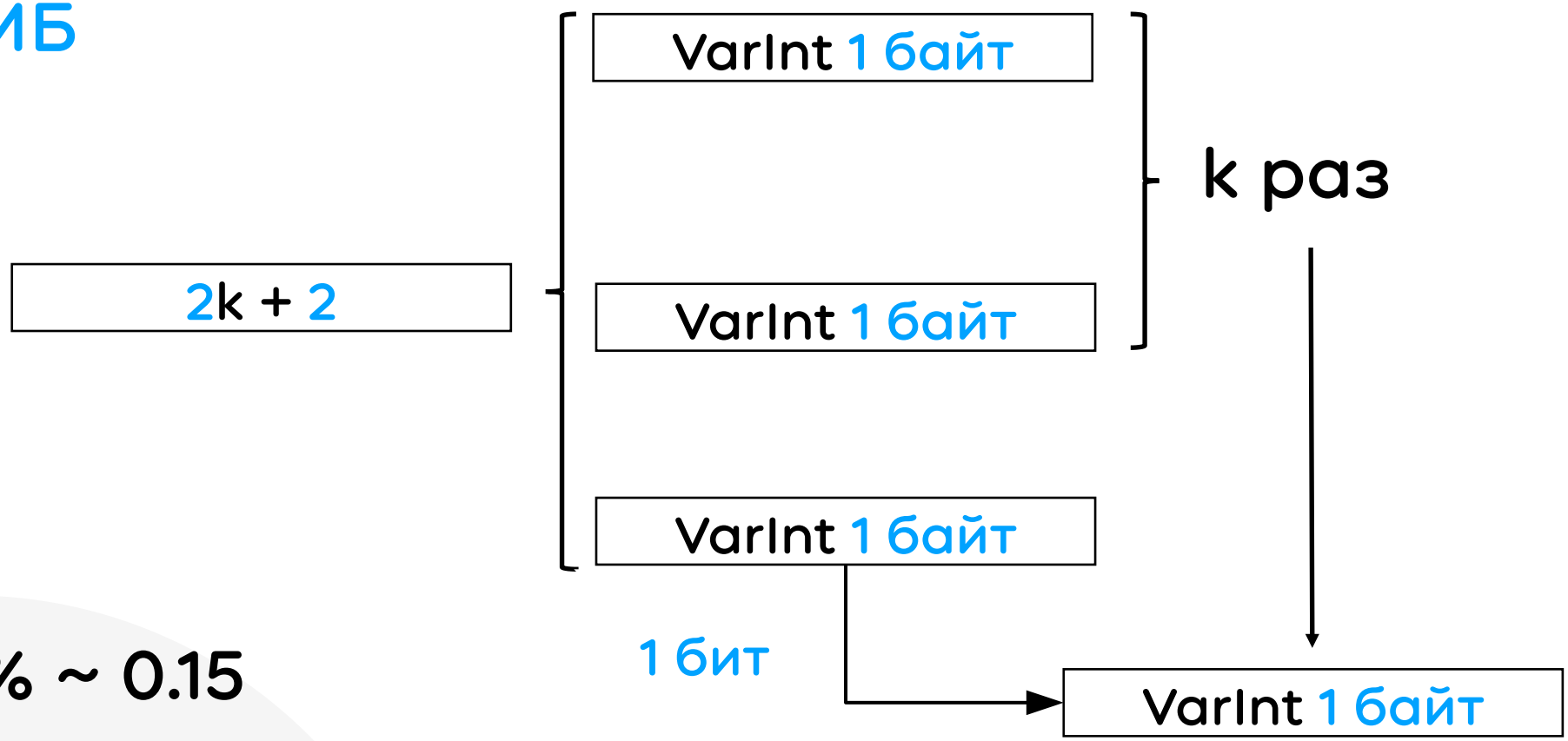
В 90% случаев k = 0

В 10% случаев k ~ 1.5

Промежуточный итог

Alloc: 78 млн сэмплов

Цель: 170 МБ

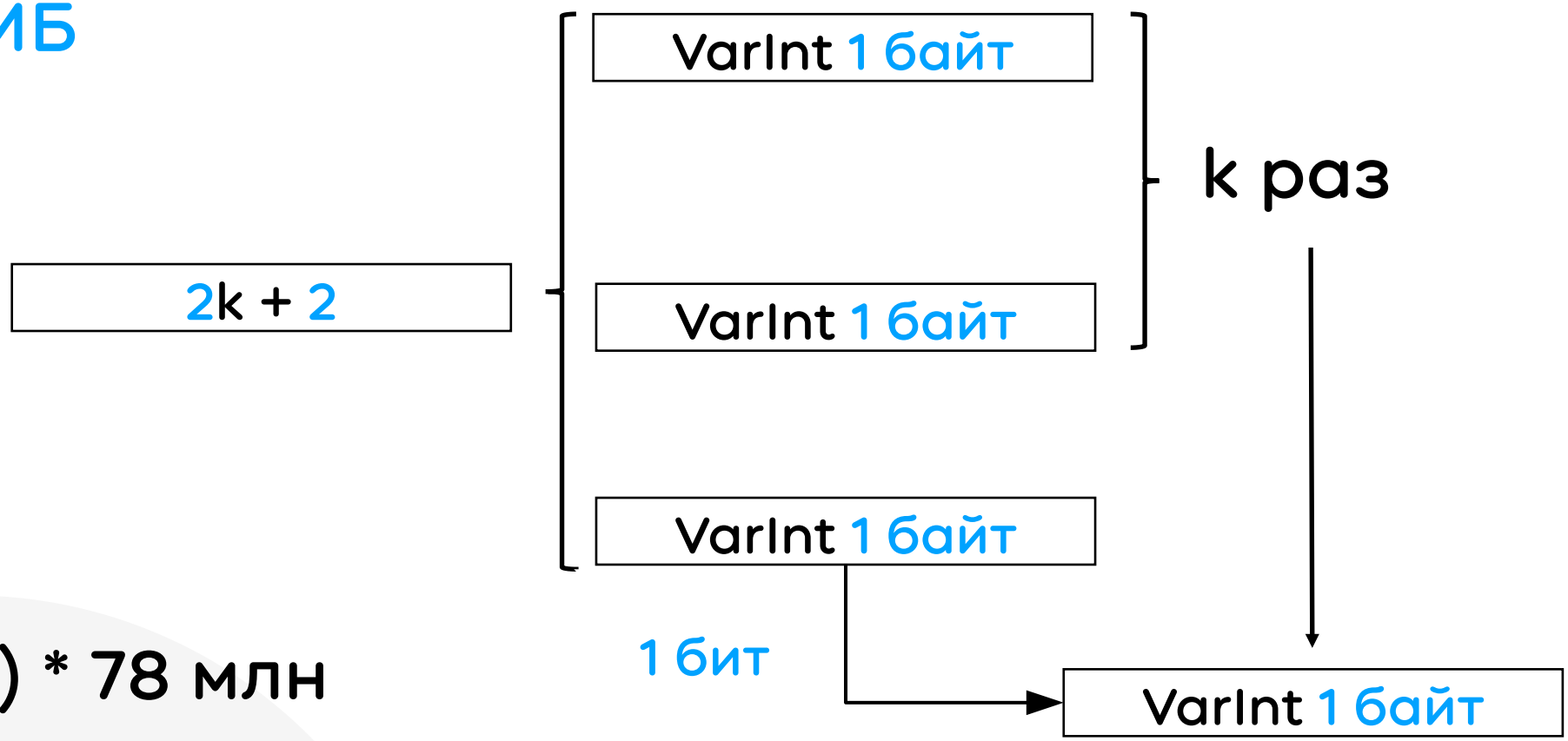


k ~ 1.5 * 10% ~ 0.15

Промежуточный итог

Alloc: 78 млн сэмплов

Цель: 170 МБ

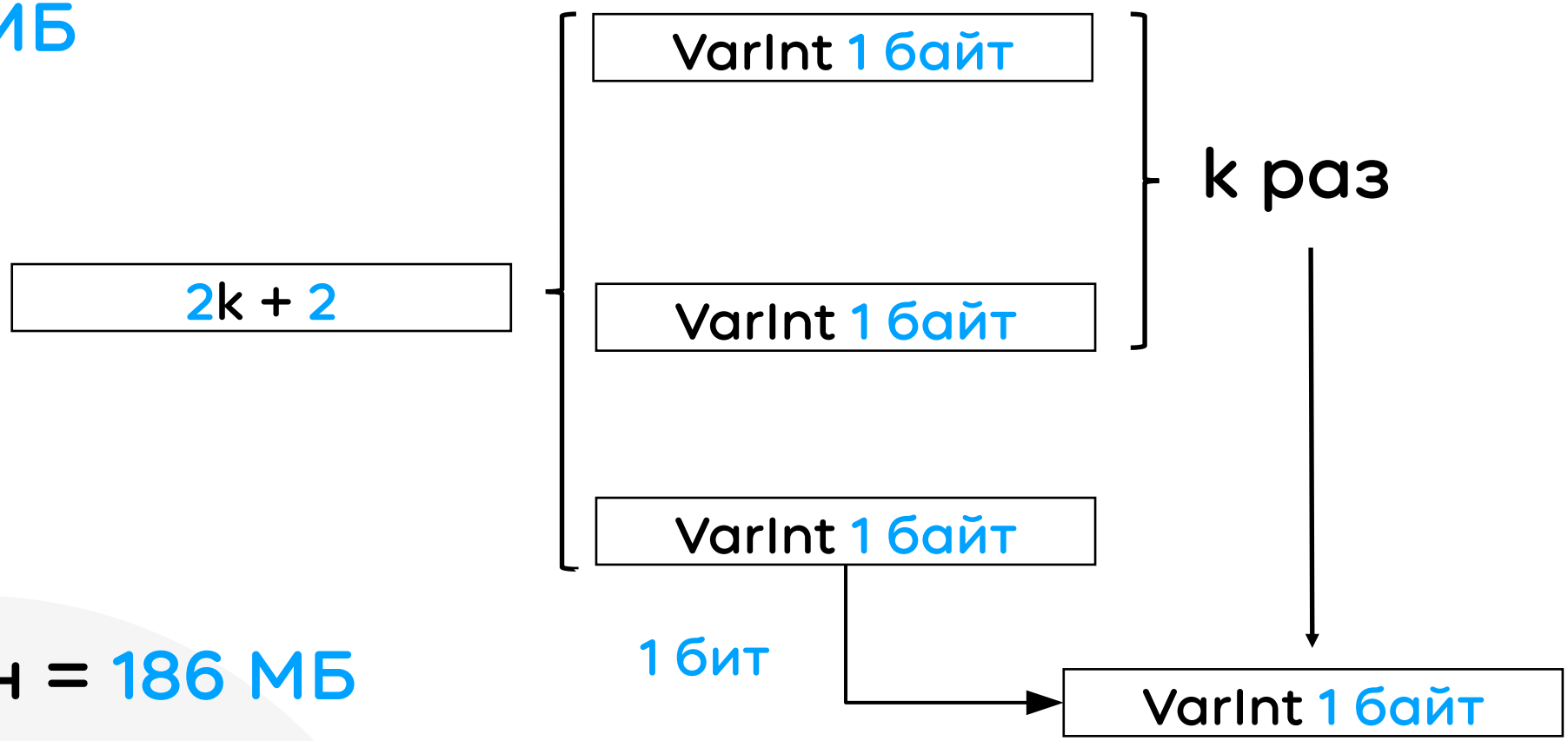


(2 * 0.15 + 2) * 78 млн

Промежуточный итог

Alloc: 78 млн сэмплов

Цель: 170 МБ

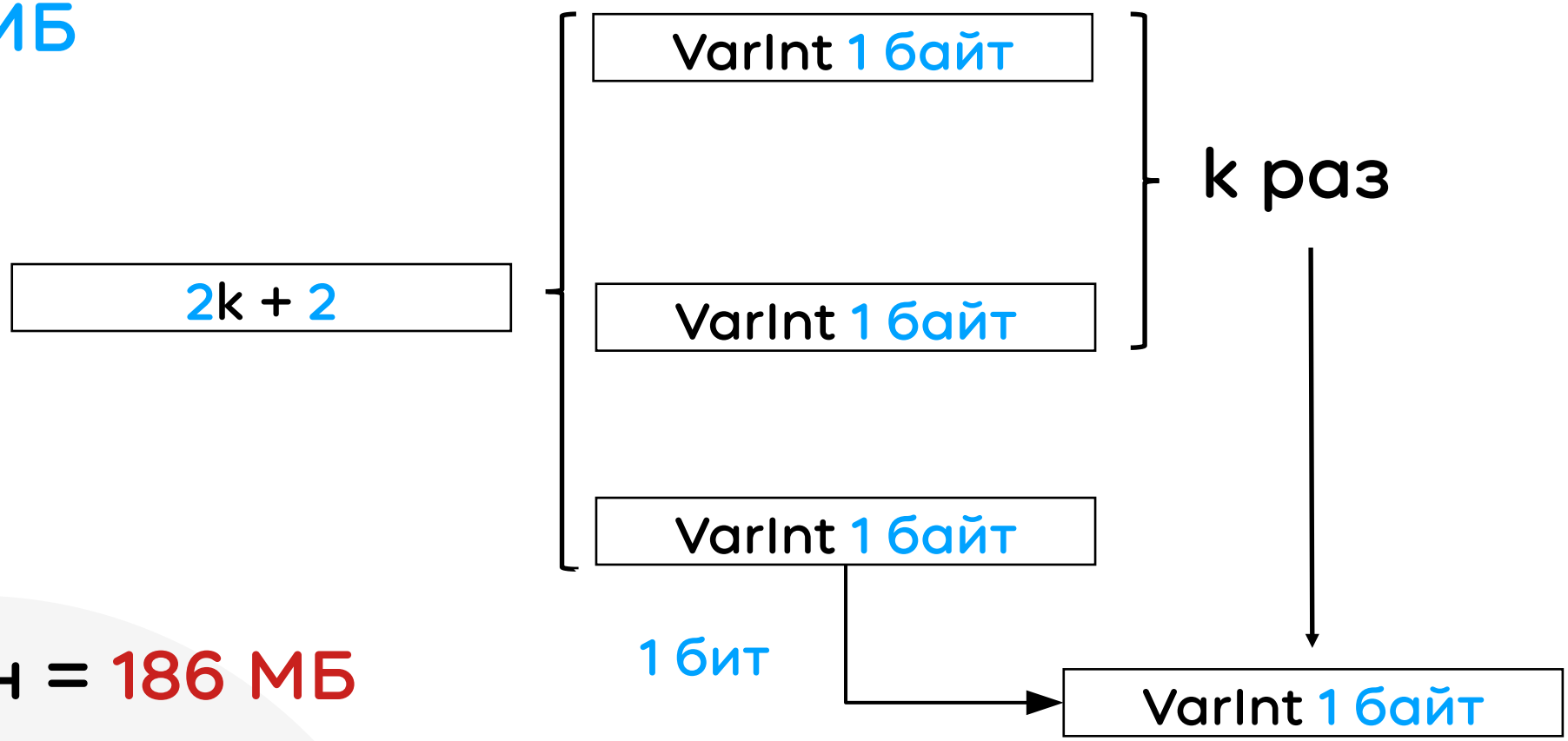


2.3 * 78 млн = 186 МБ

Промежуточный итог

Alloc: 78 млн сэмплов

Цель: 170 МБ



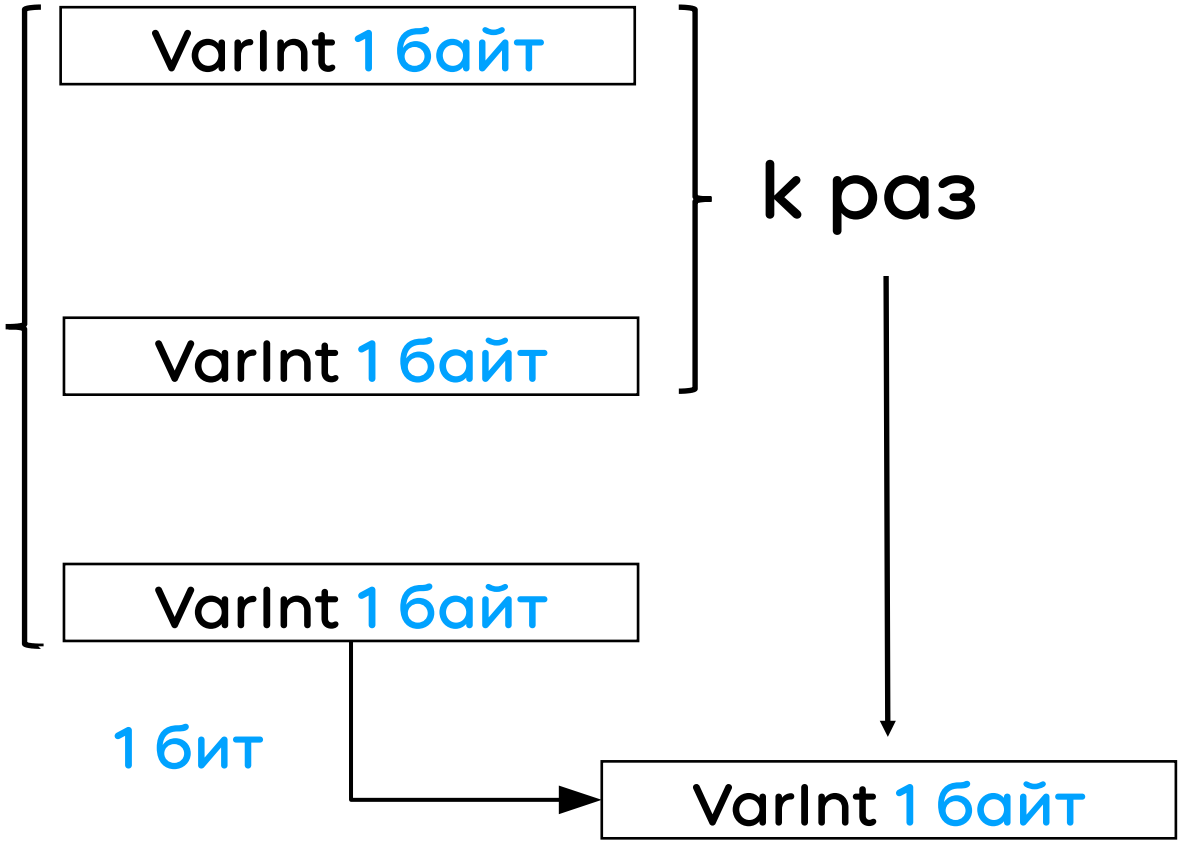
2.3 * 78 млн = 186 МБ

Промежуточный итог

Alloc: 78 млн сэмплов

Цель: 170 МБ

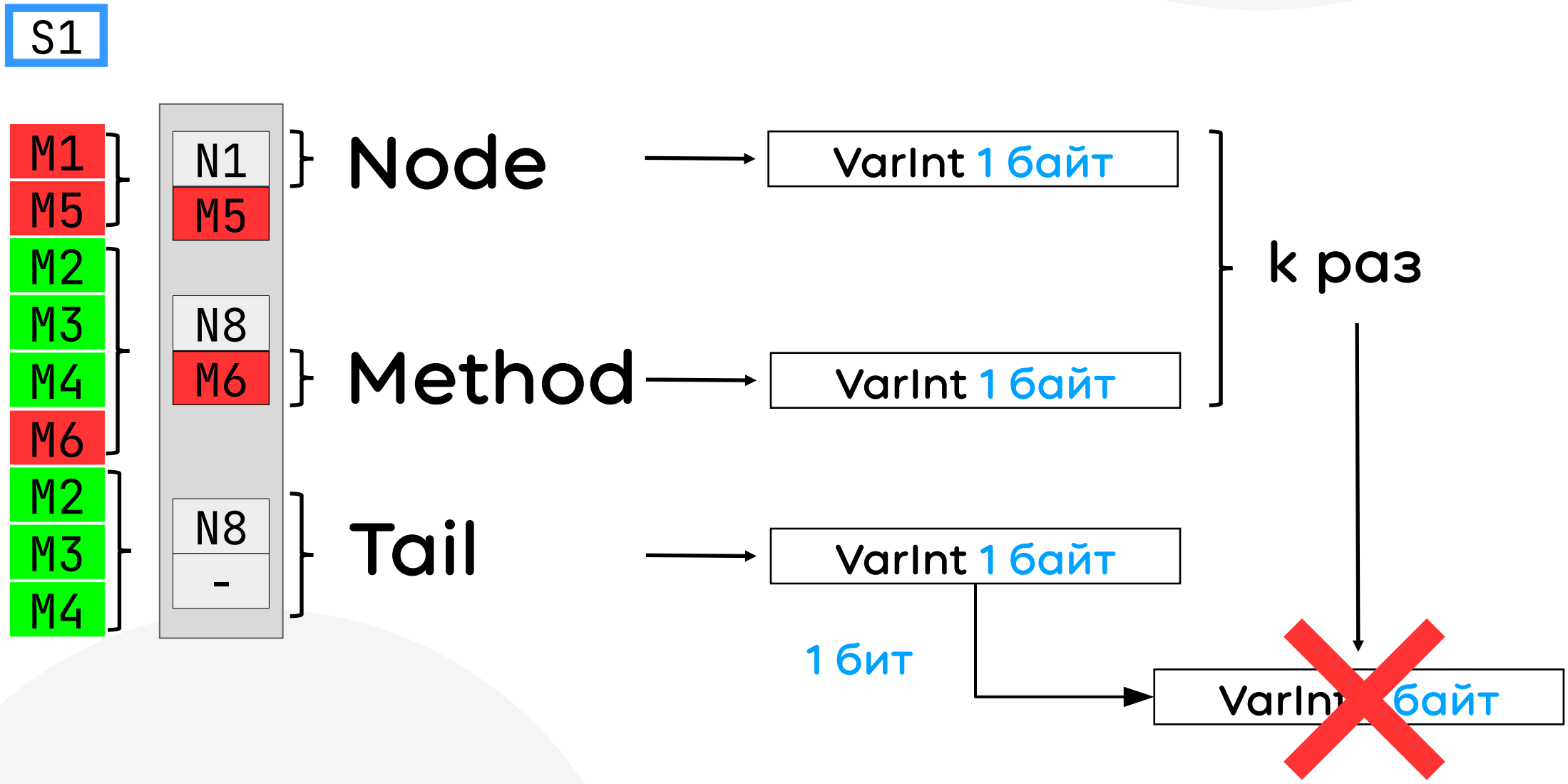
2.3 * 78 млн = 186 МБ



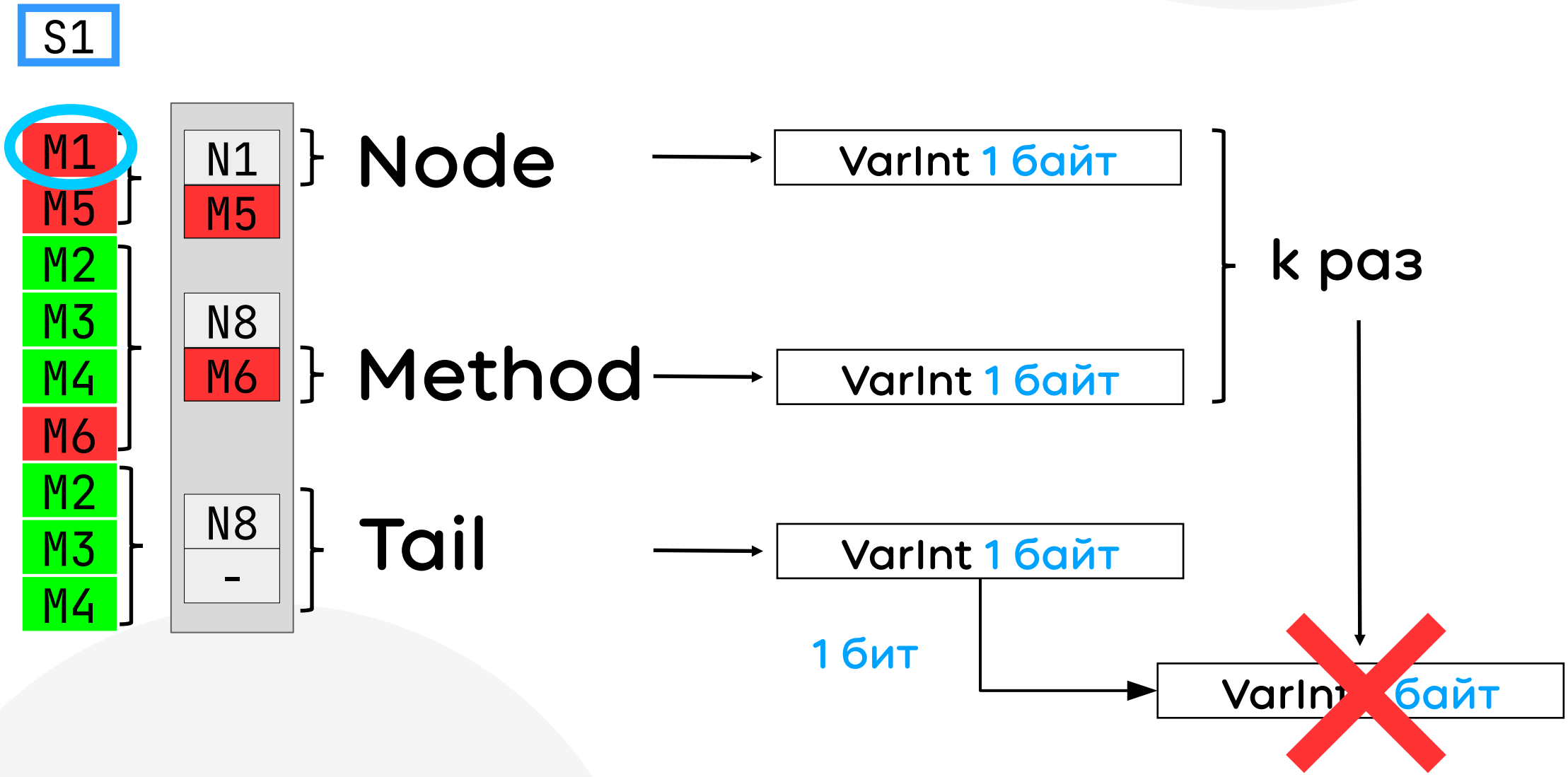
Сжимаем

Со всей силы

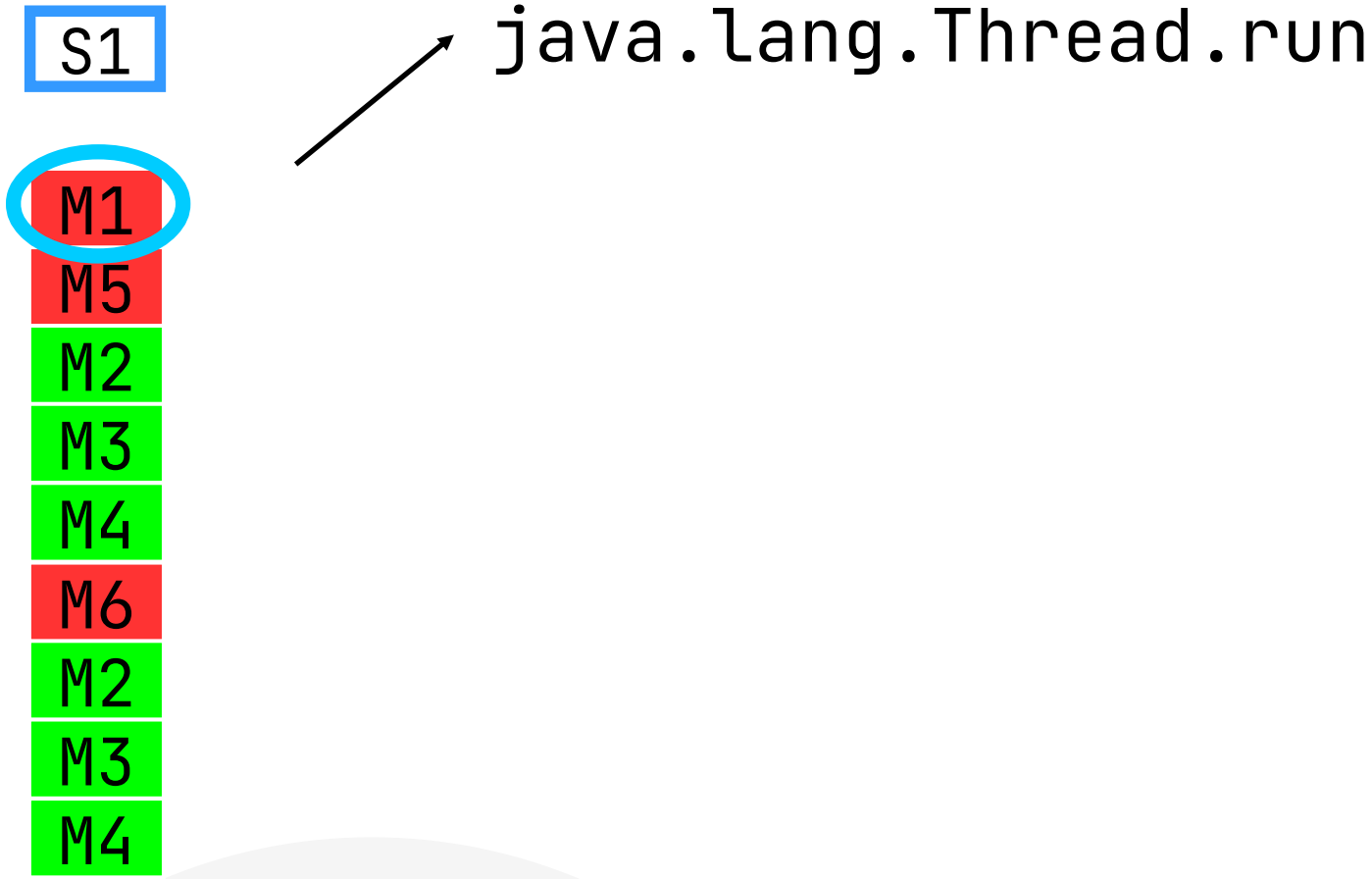
Сжимаем со всей силы



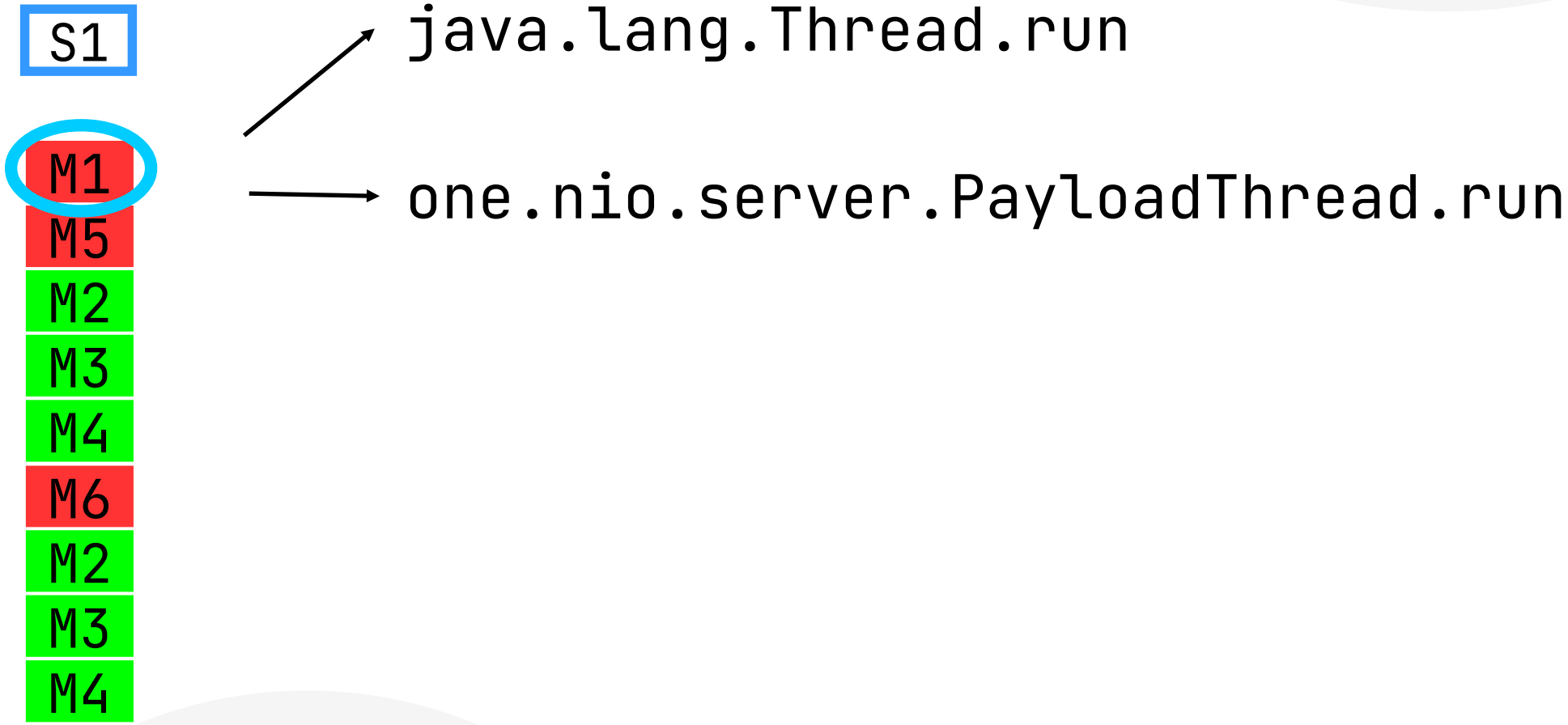
Сжимаем со всей силы



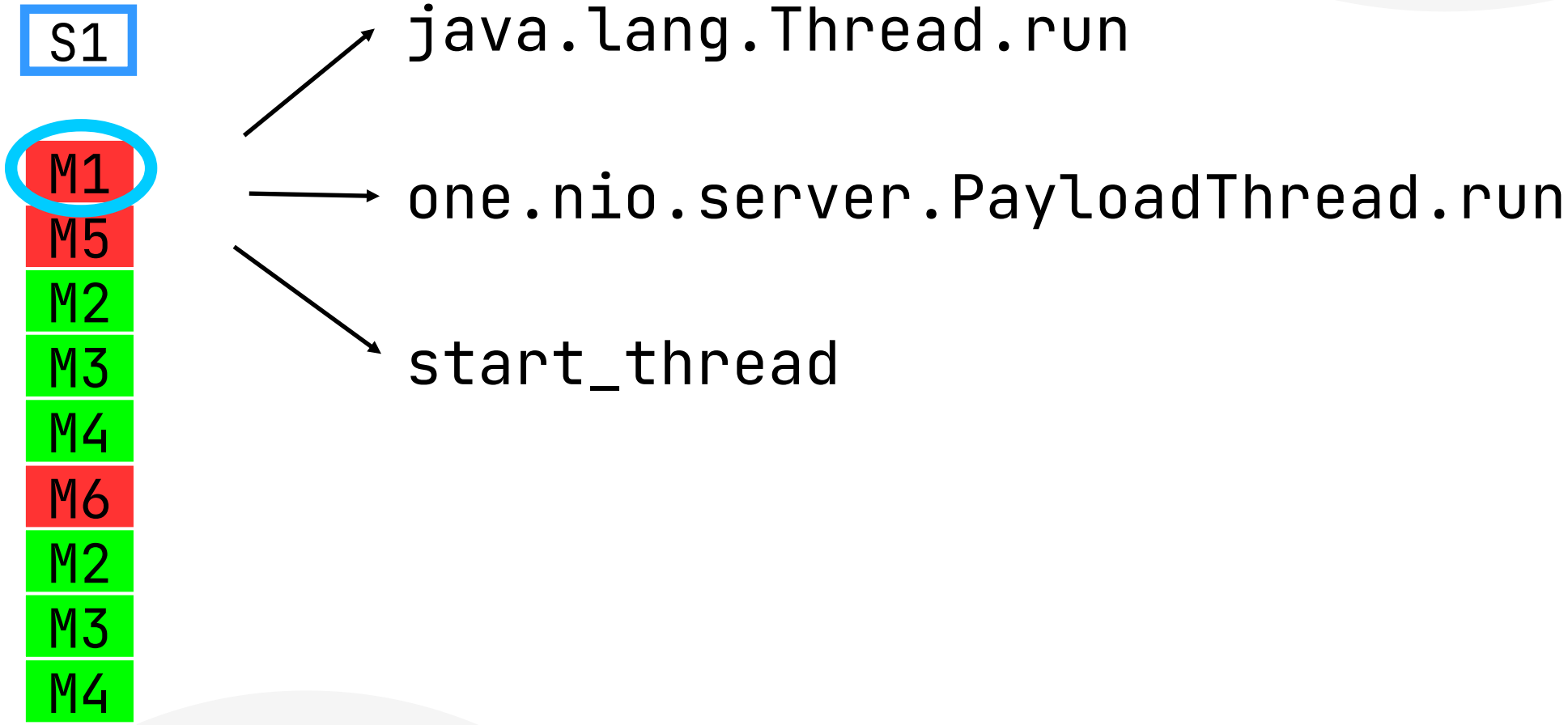
Сжимаем со всей силы



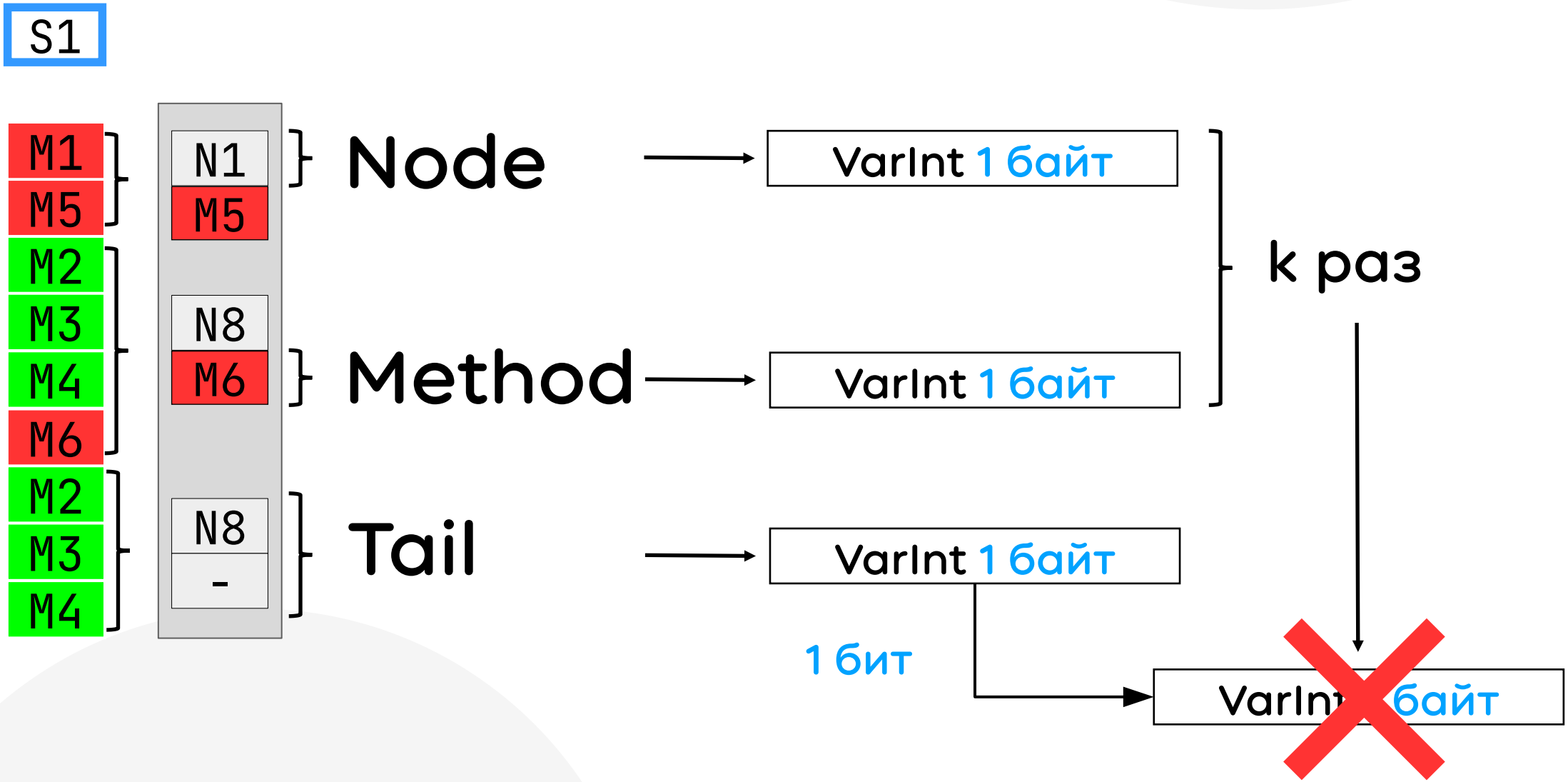
Сжимаем со всей силы



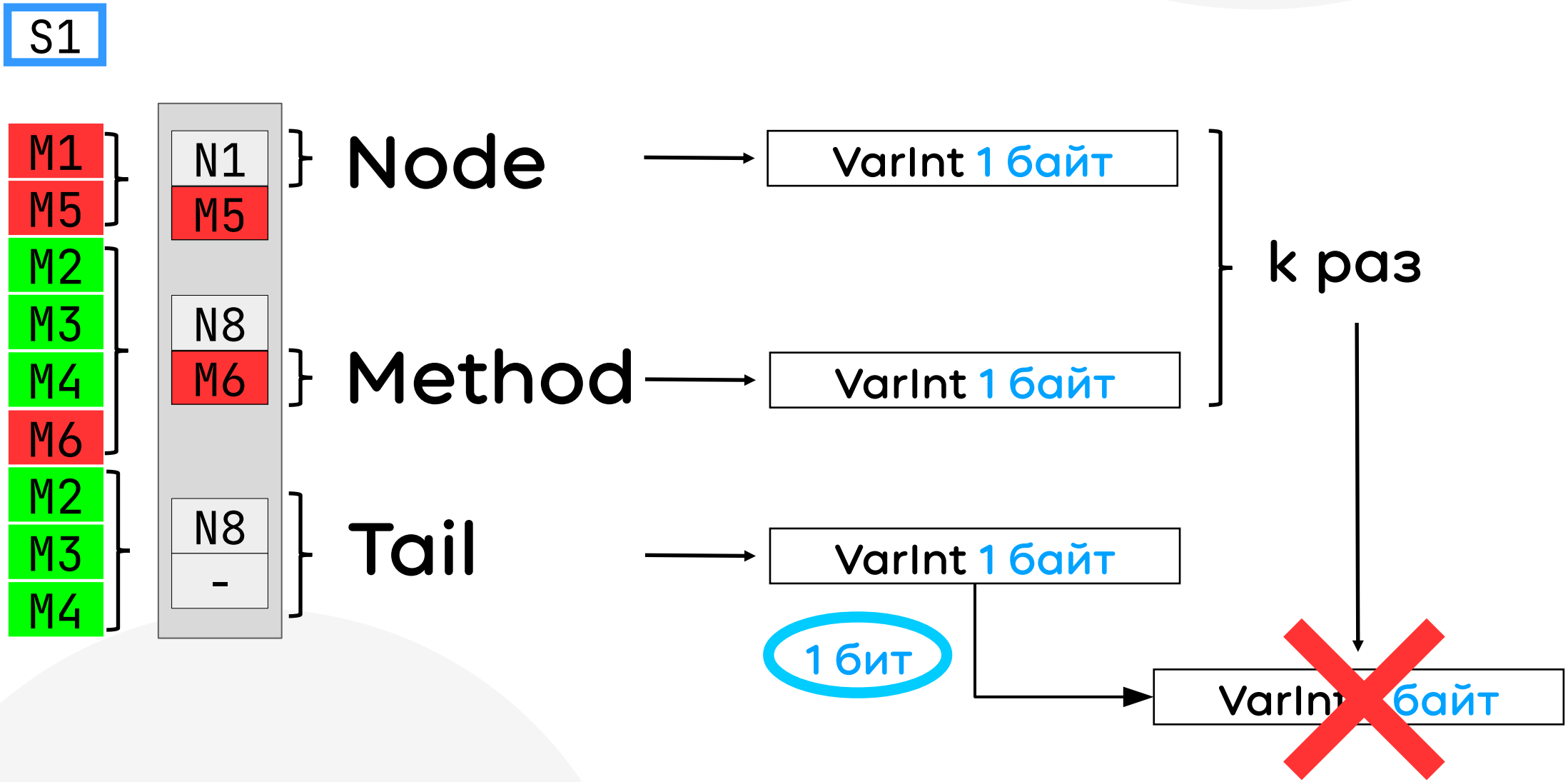
Сжимаем со всей силы



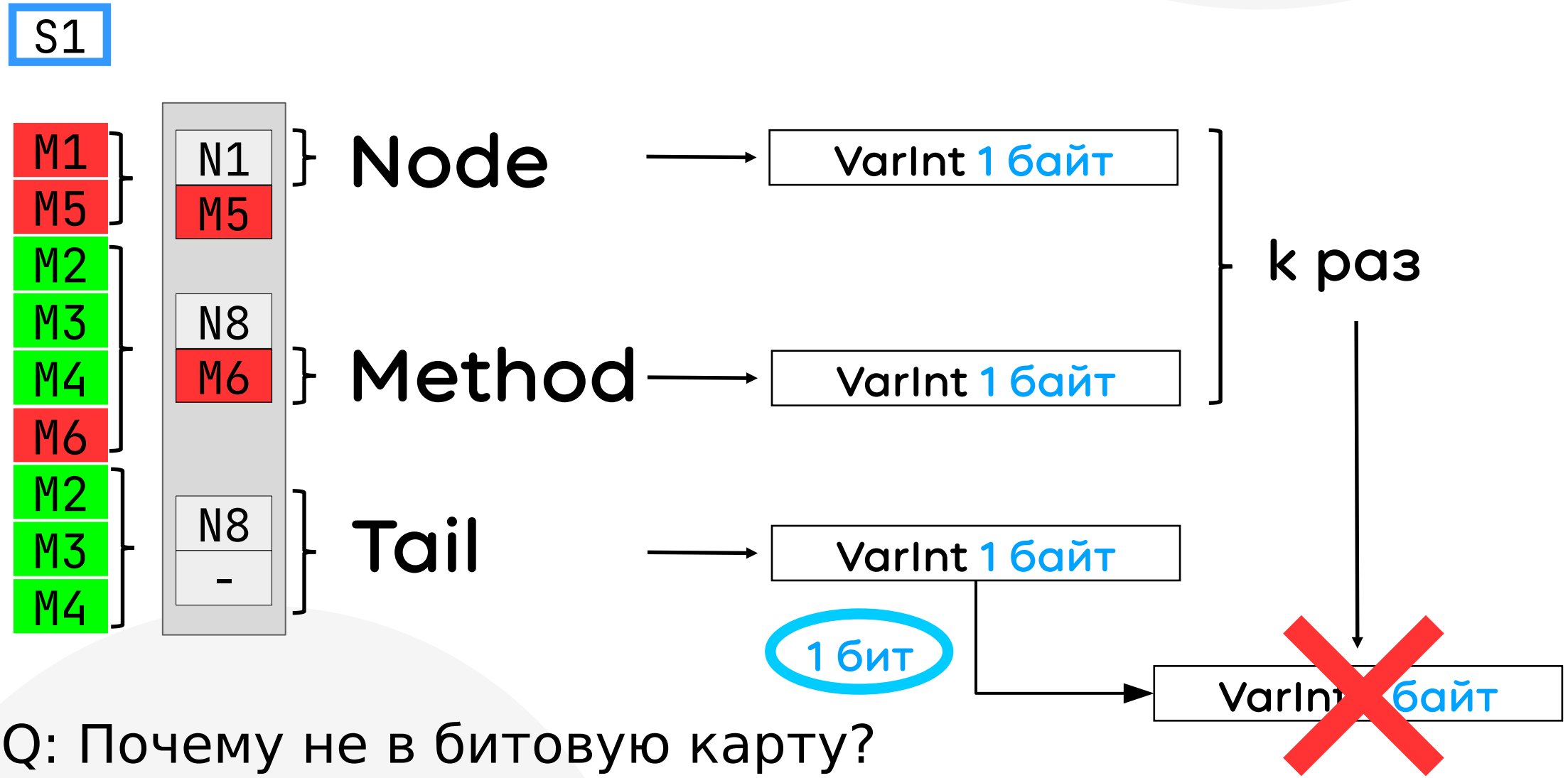
Сжимаем со всей силы



Сжимаем со всей силы

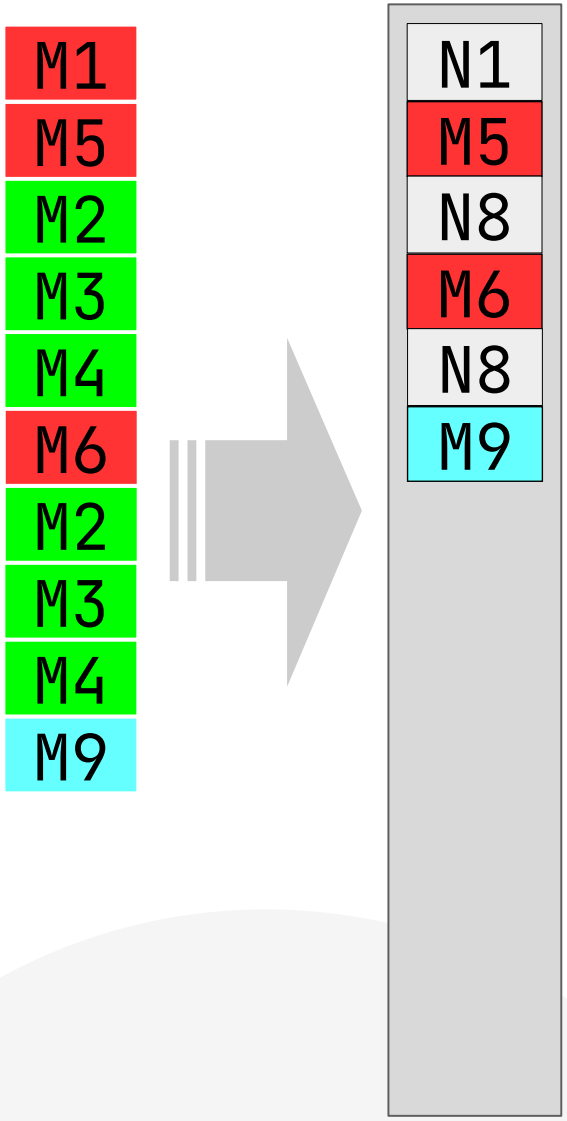


Сжимаем со всей силы



Q: Почему не в битовую карту?

Сжимаем со всей силы

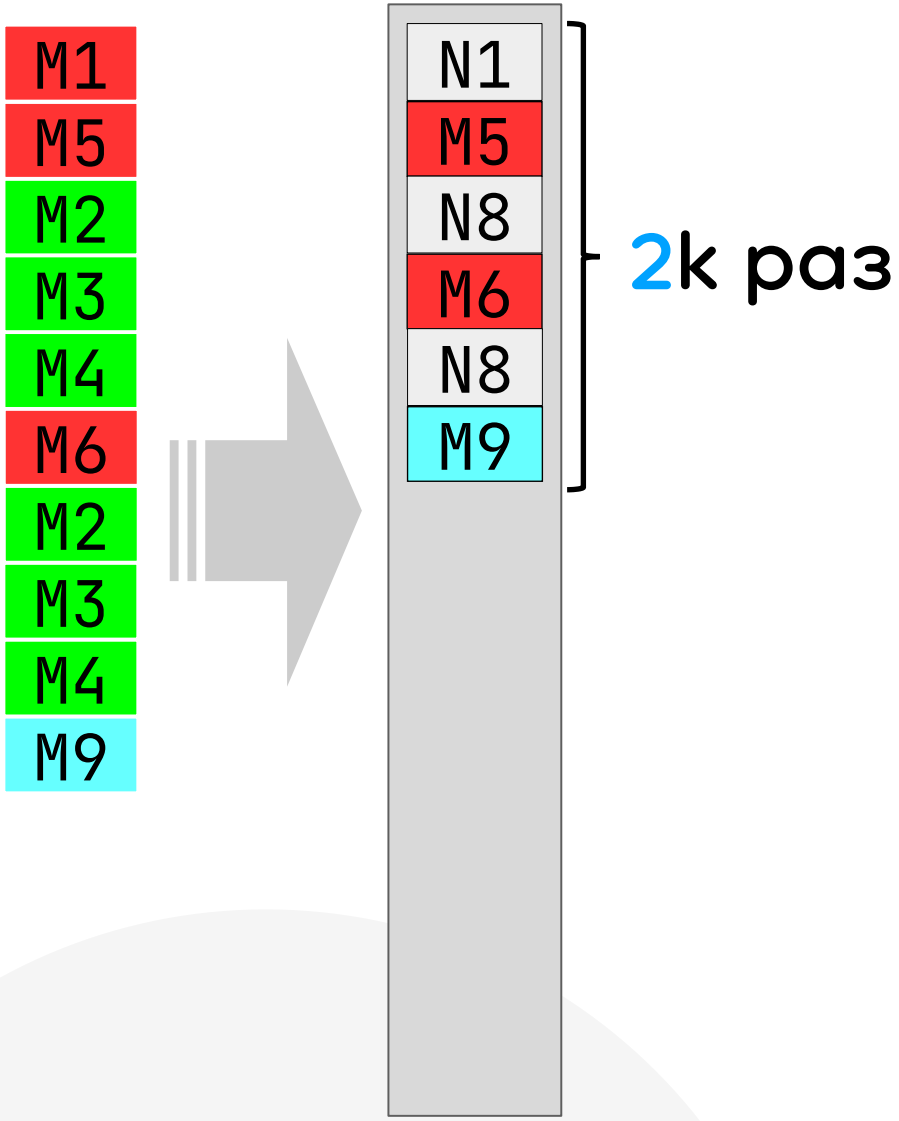


LZ78

Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

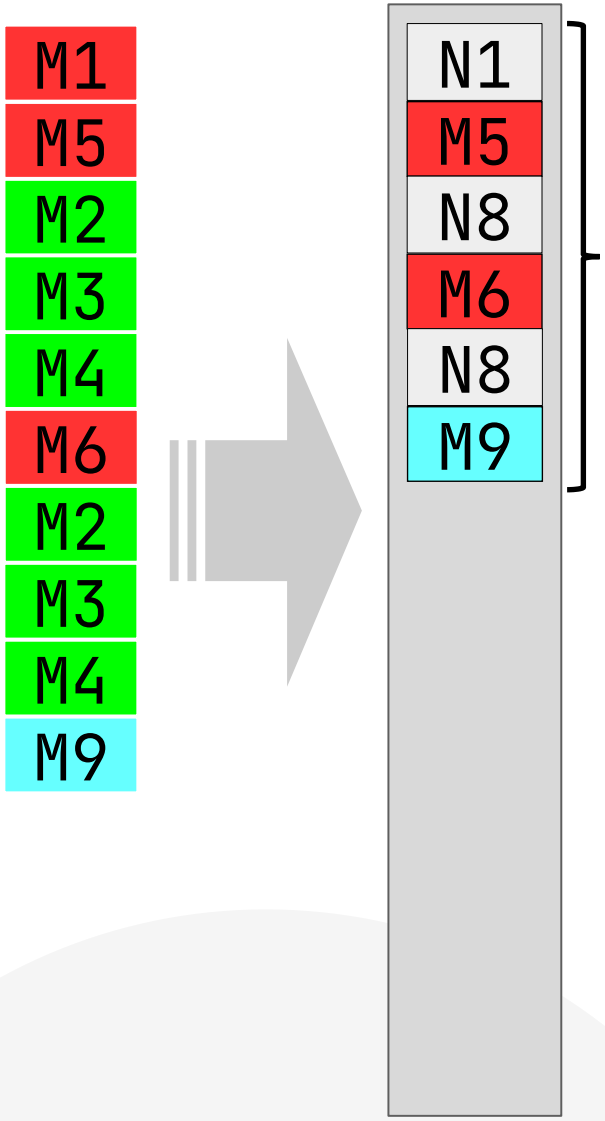
Сжимаем со всей силы



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

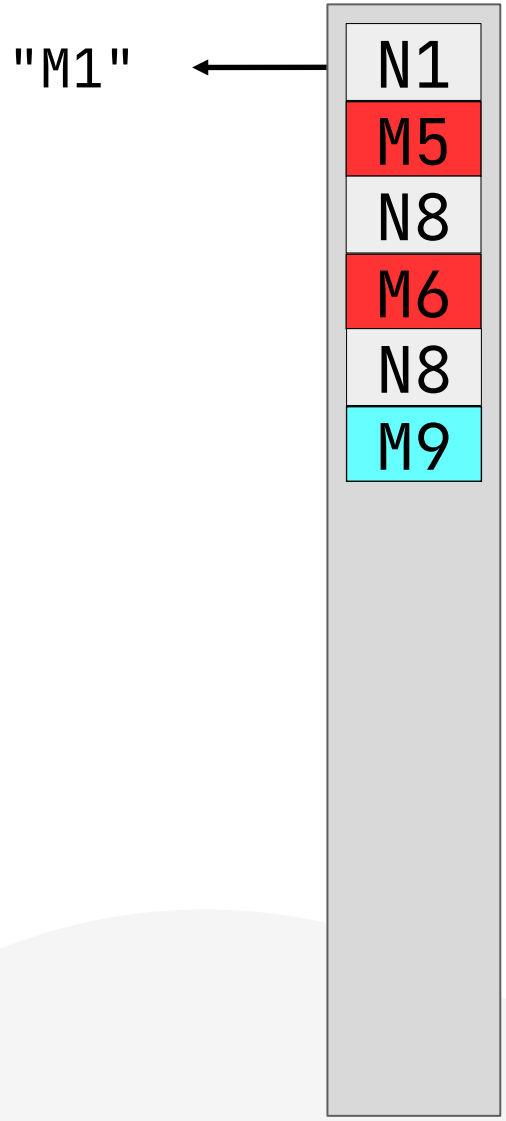
Сжимаем со всей силы



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

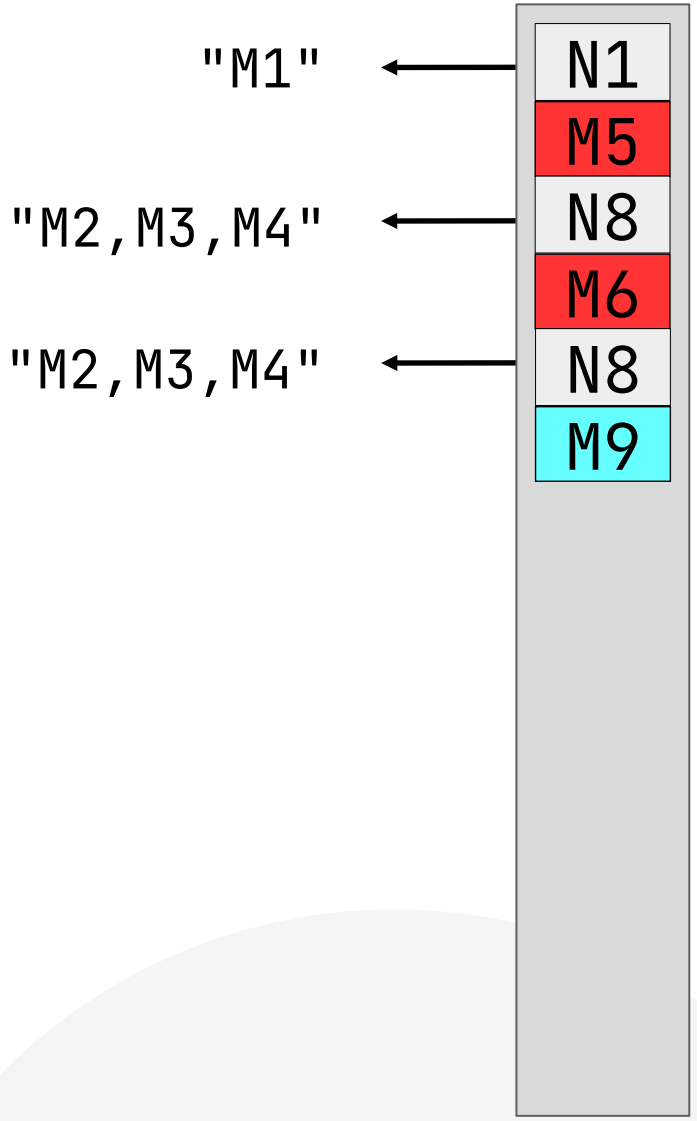
Сжимаем со всей силы



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

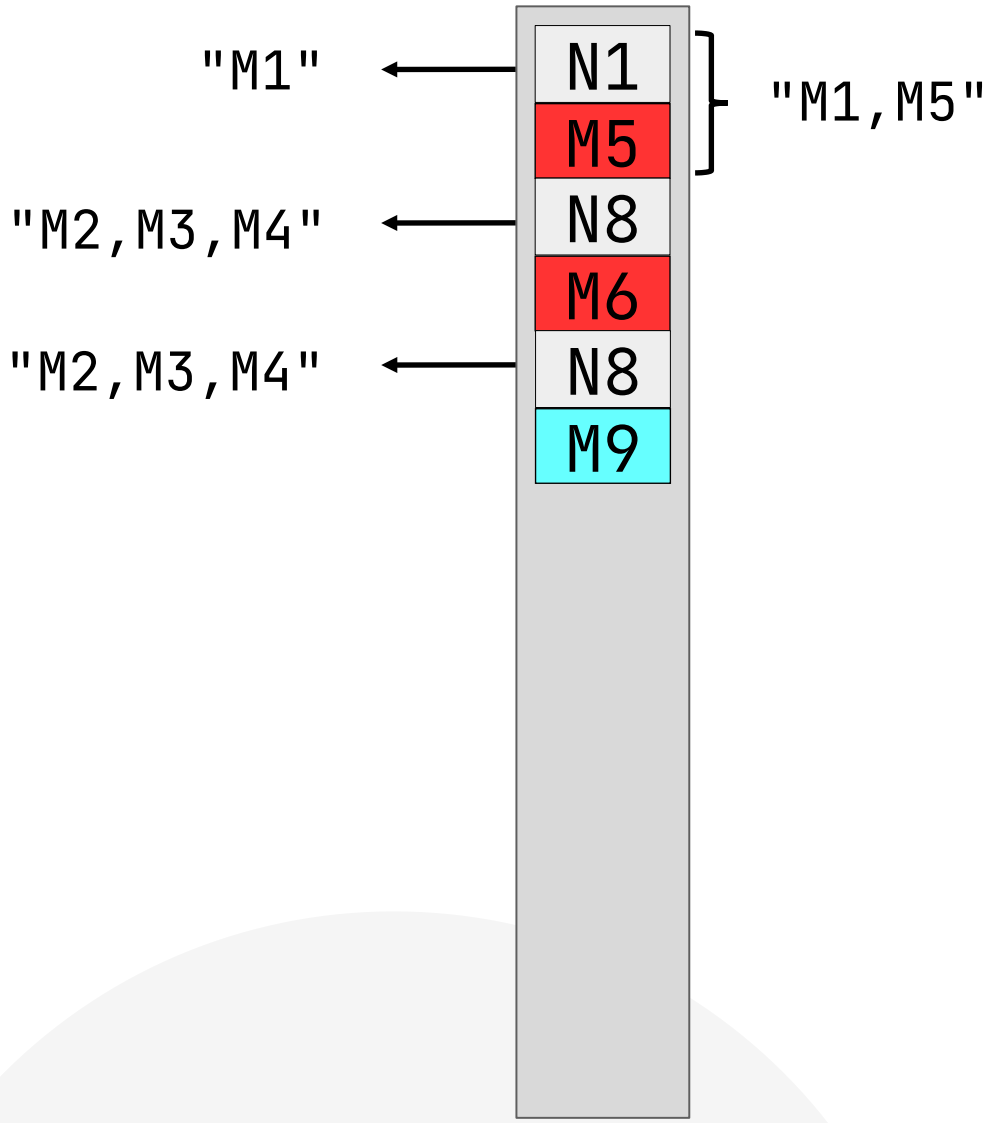
Сжимаем со всей силы



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

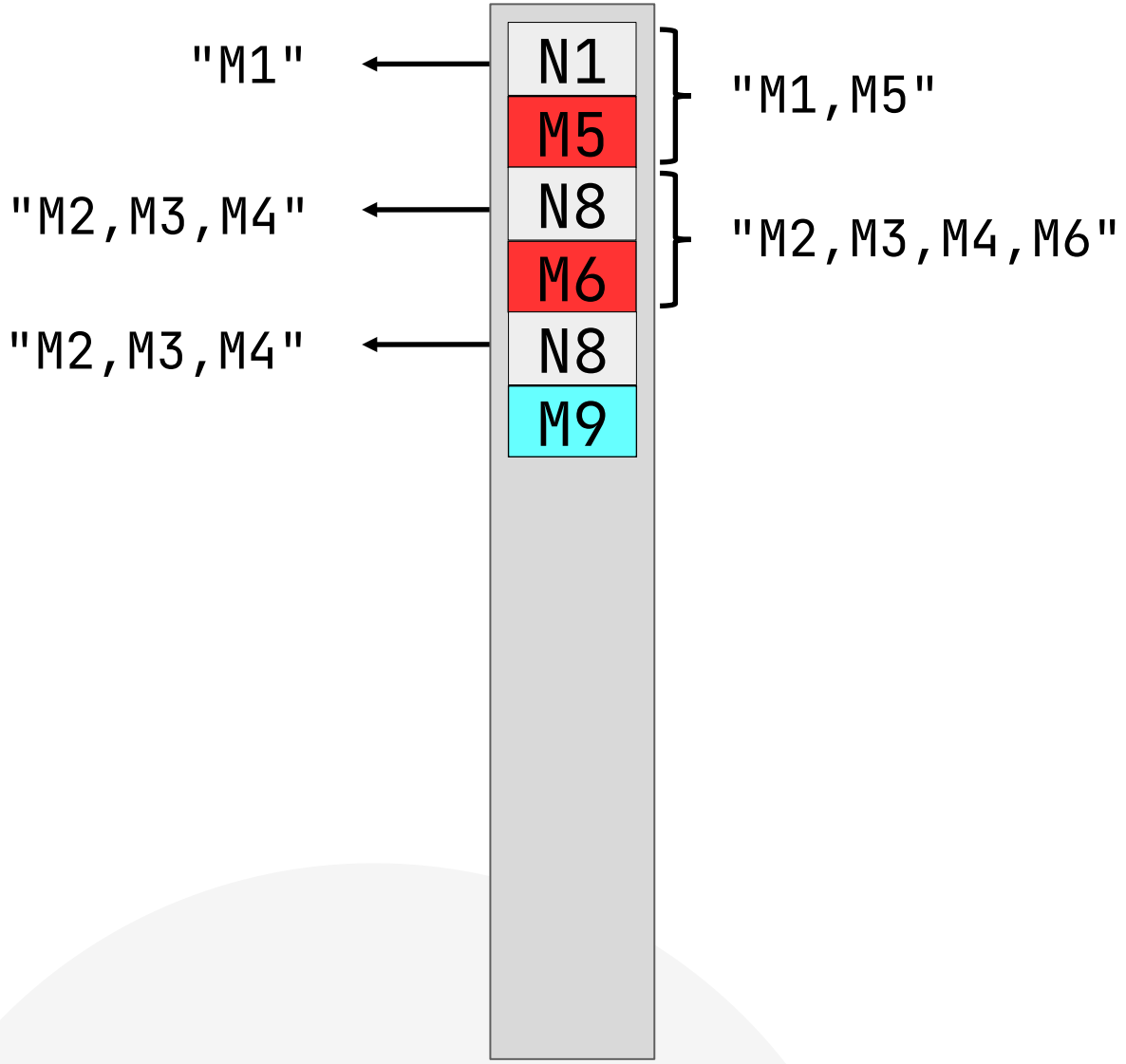
Сжимаем со всей силы



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

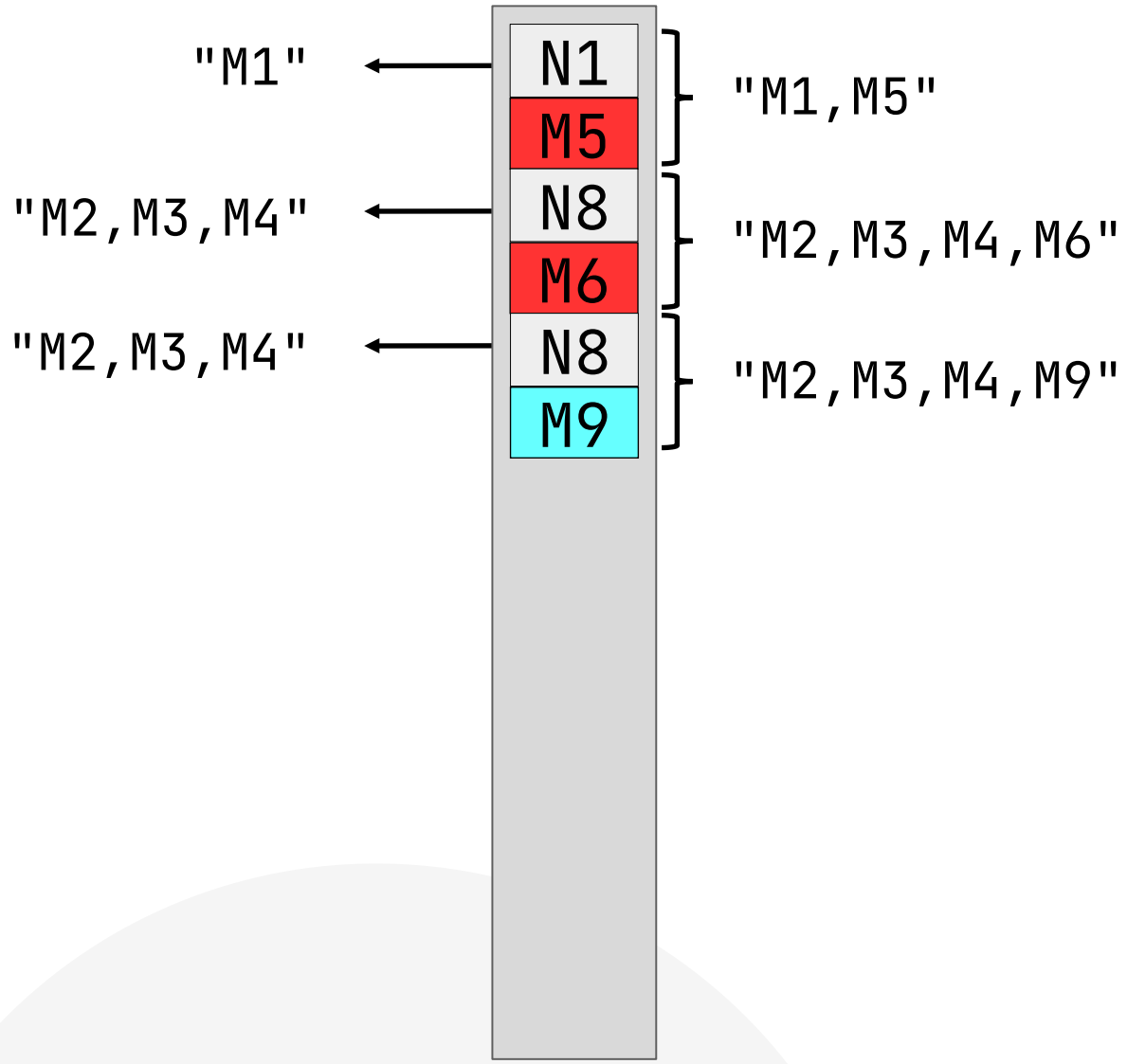
Сжимаем со всей силы



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

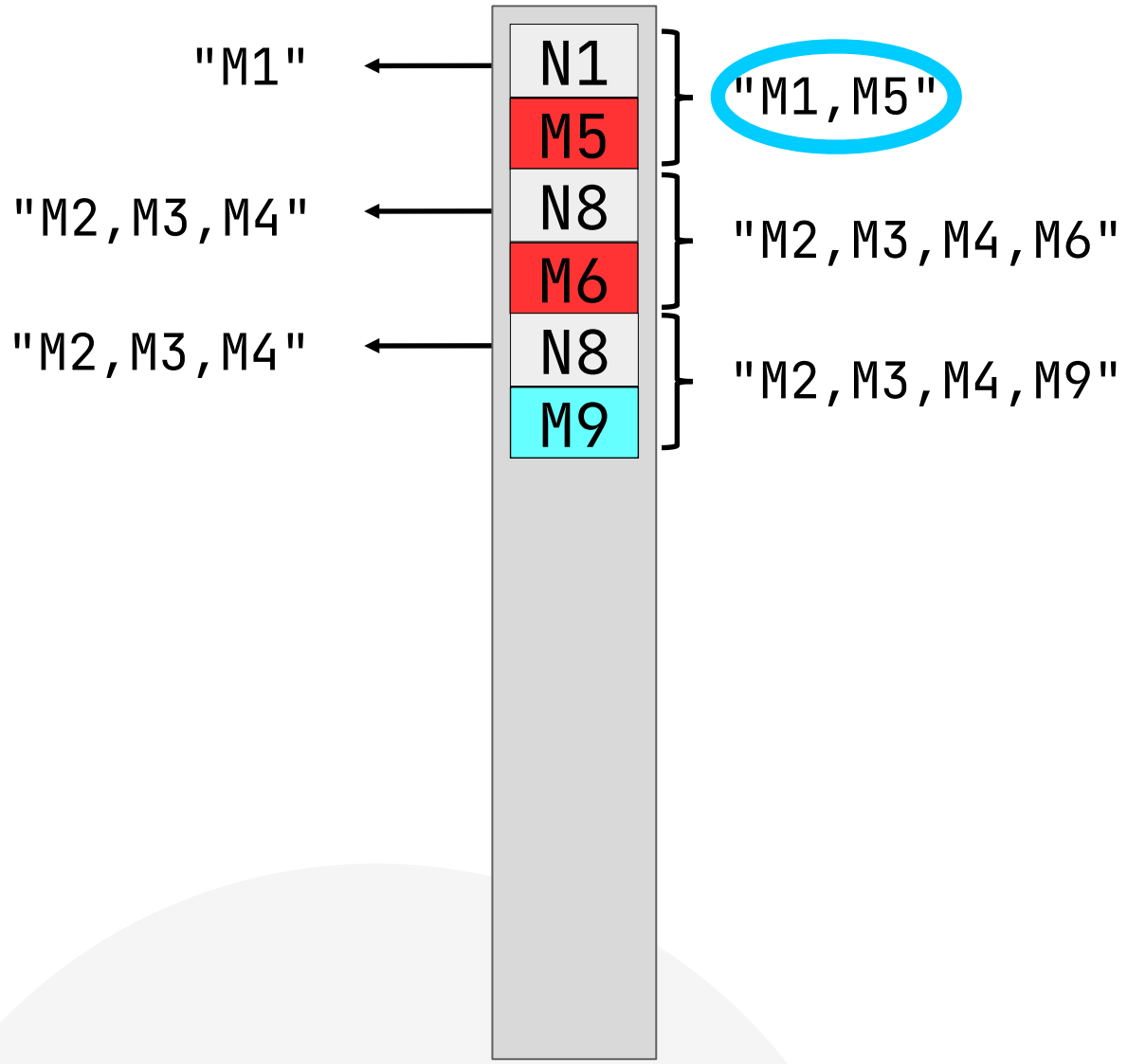
Сжимаем со всей силы



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

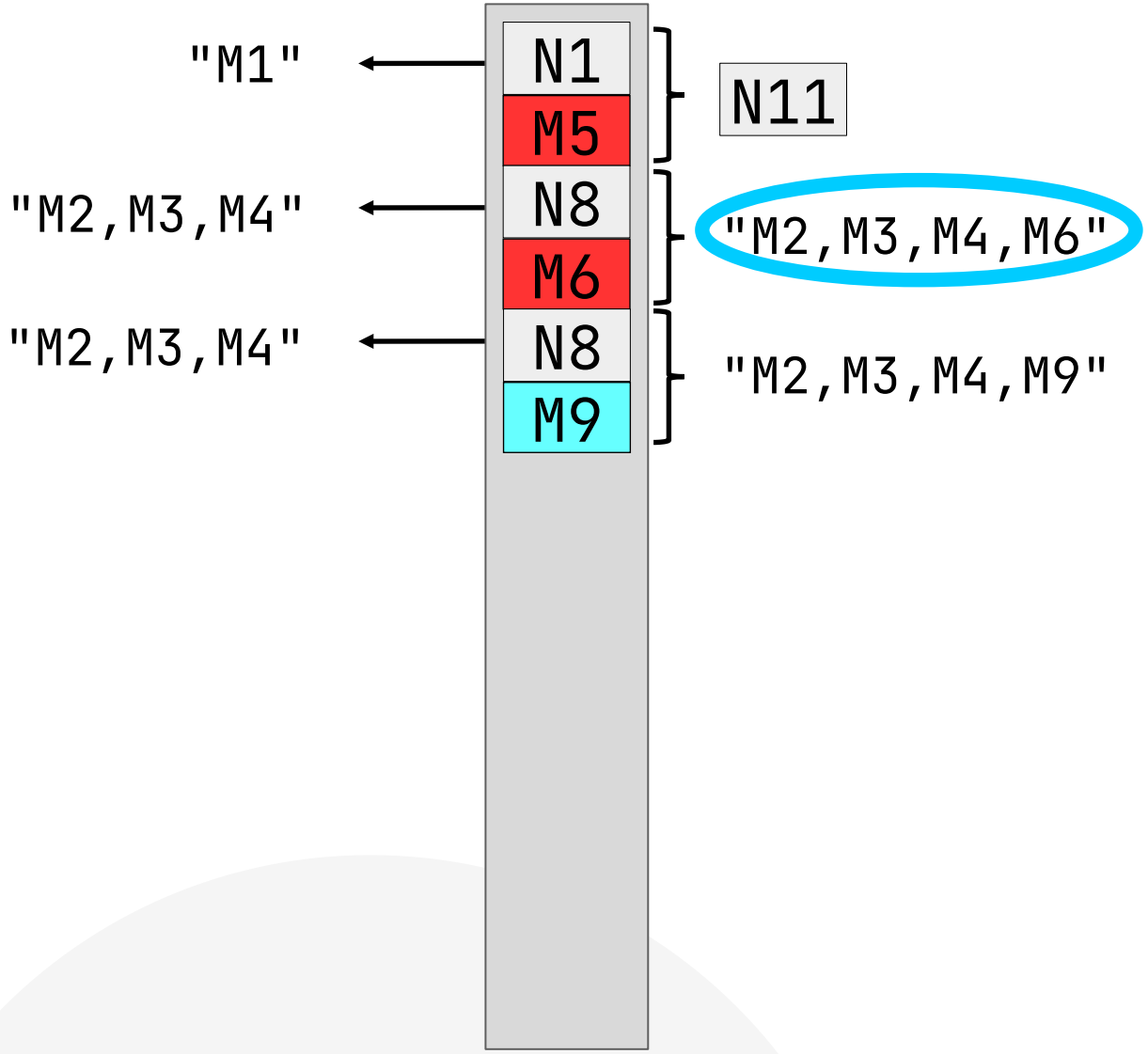
Сжимаем со всей силы



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

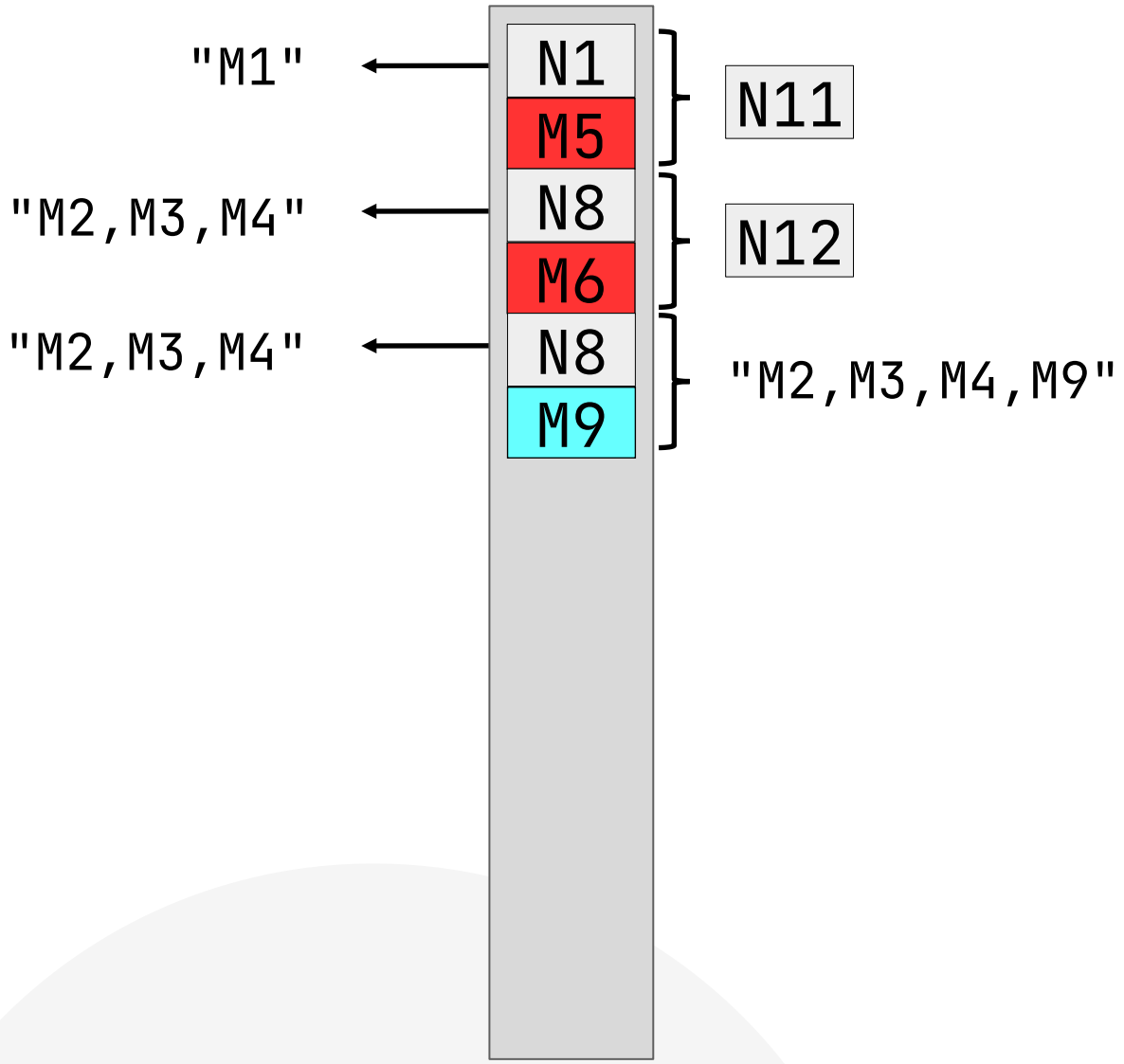
Сжимаем со всей силы



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

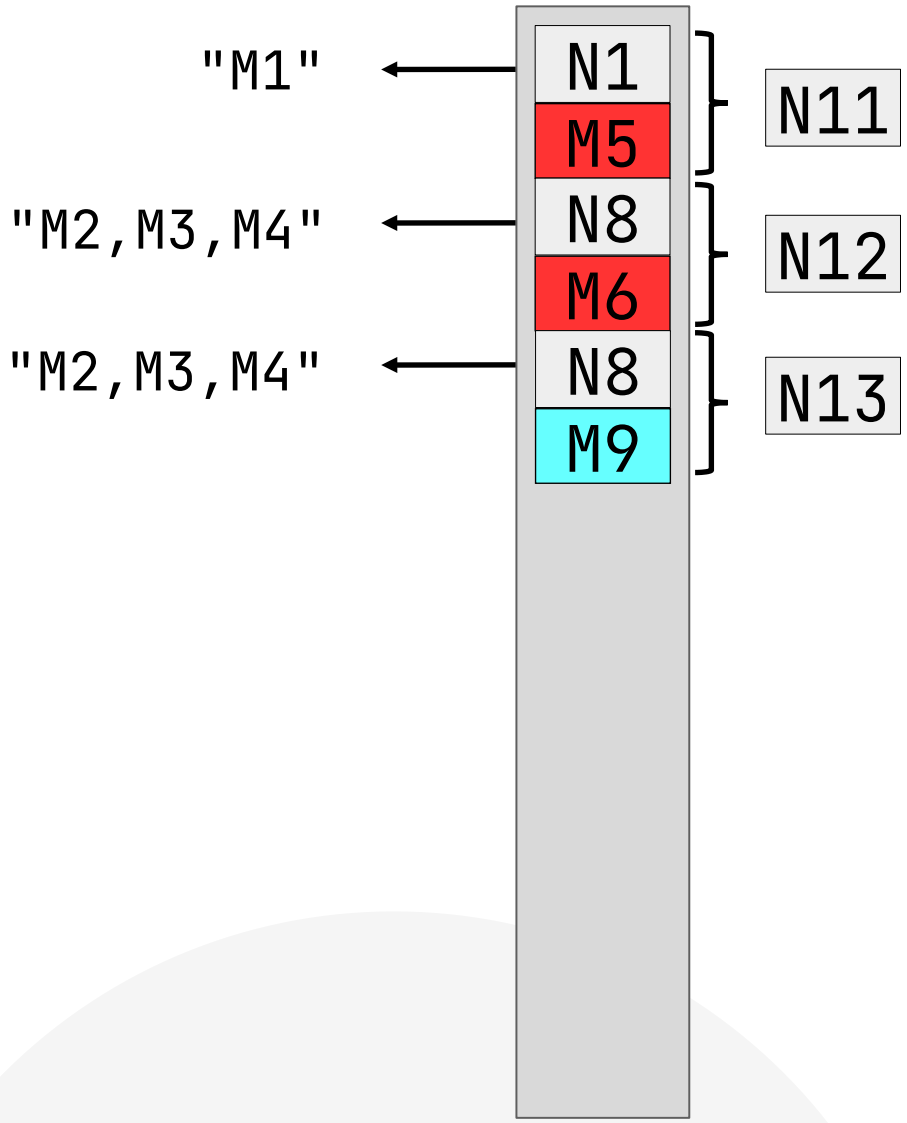
Сжимаем со всей силы



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

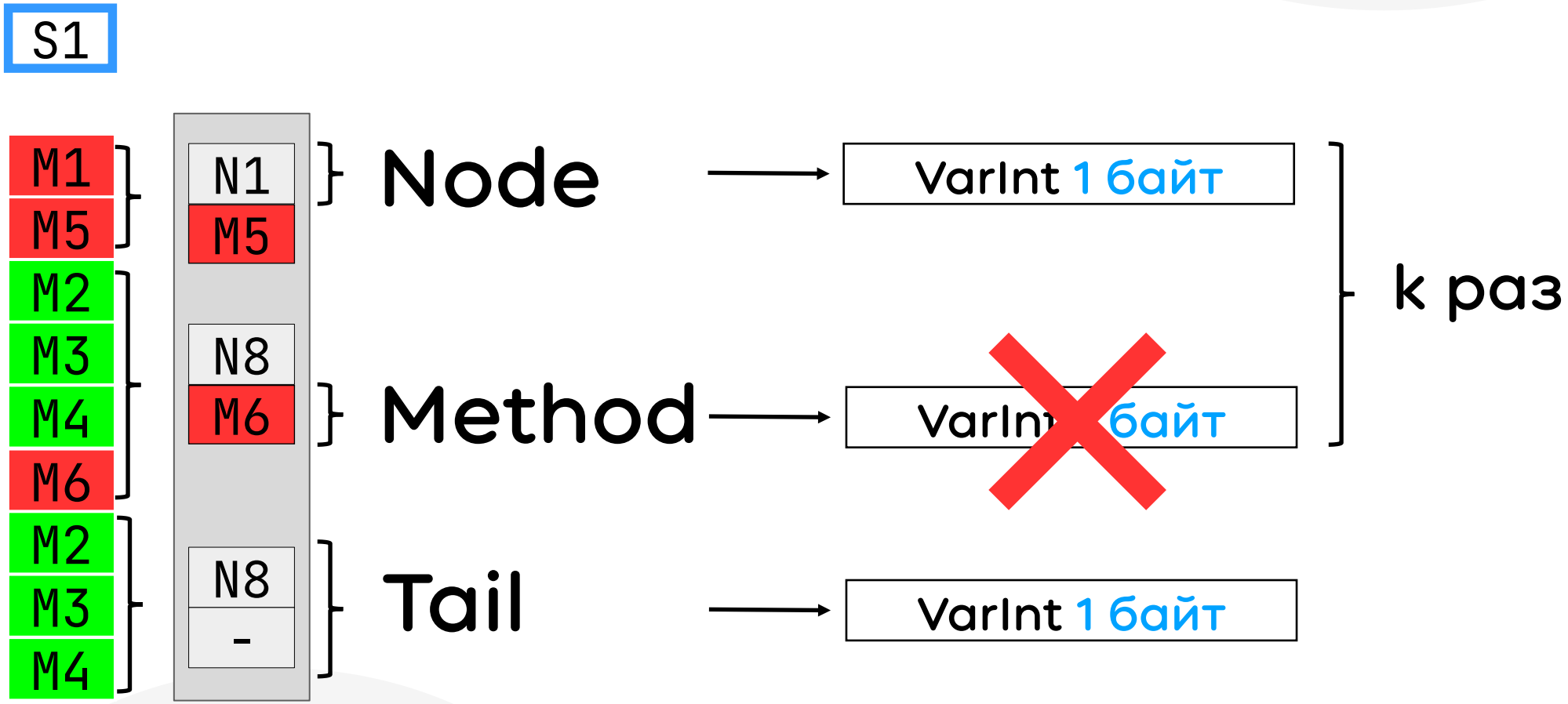
Сжимаем со всей силы



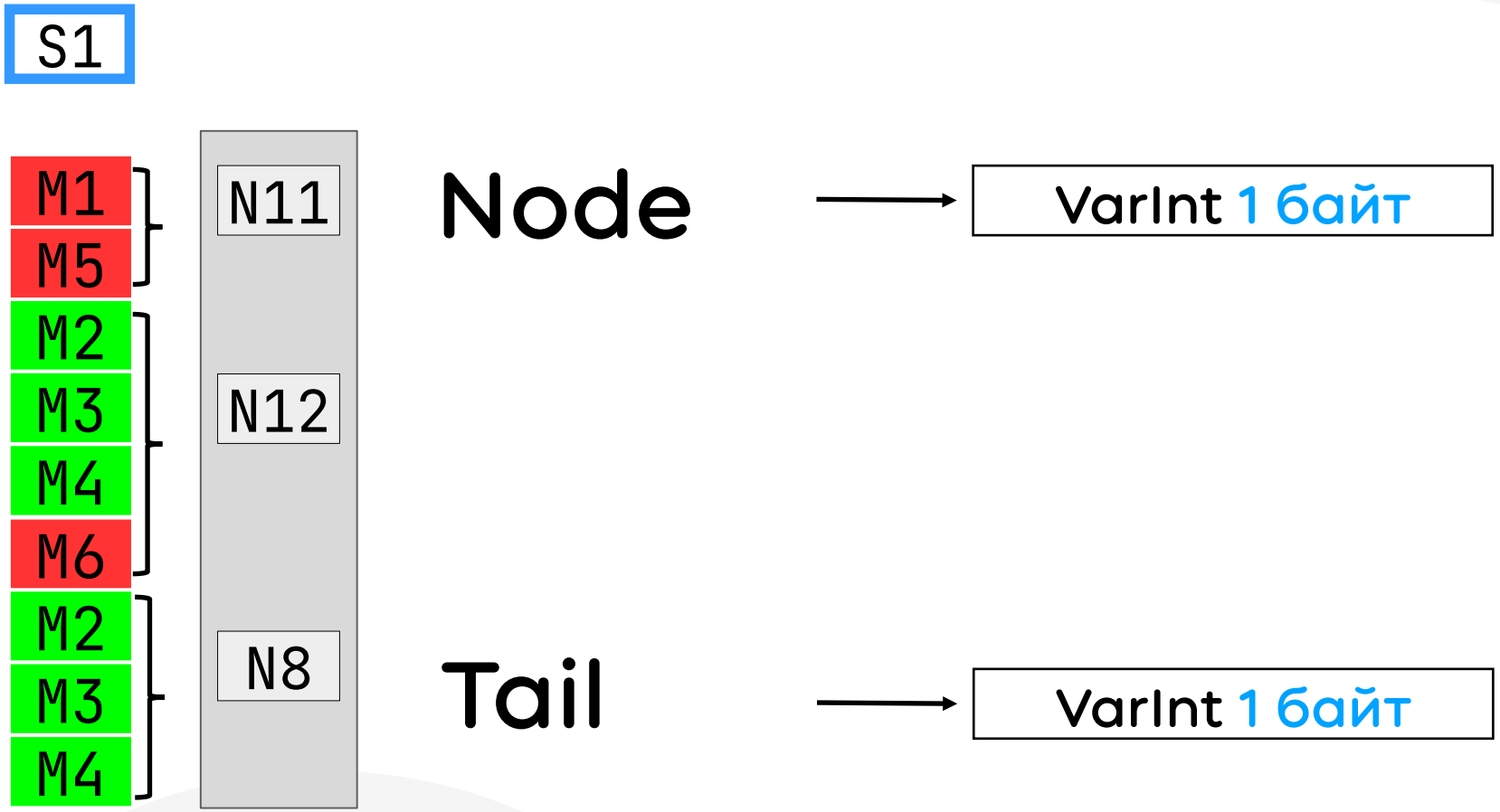
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

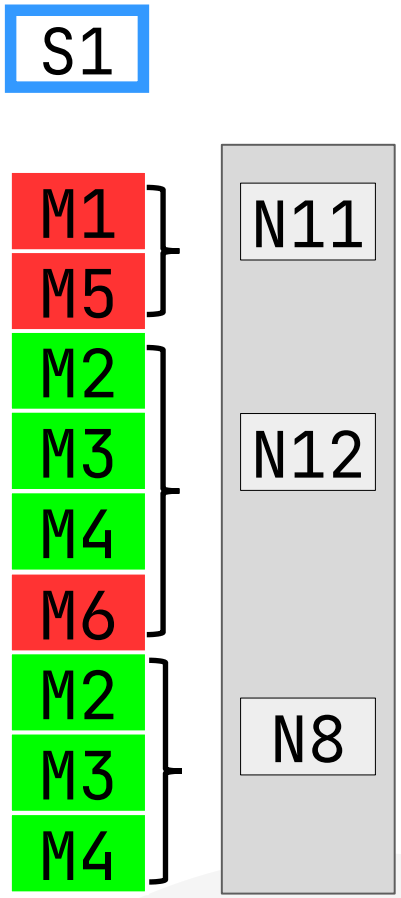
Сжимаем со всей силы



Сжимаем со всей силы



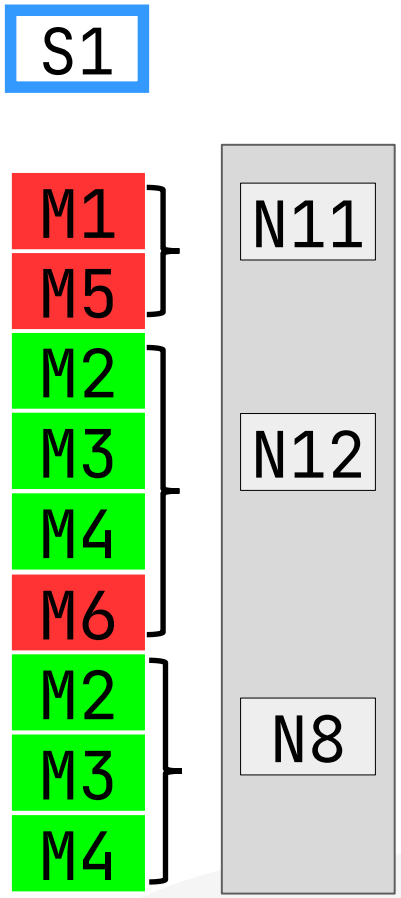
Сжимаем со всей силы



Node → VarInt 1 байт k раз

Tail → VarInt 1 байт

Сжимаем со всей силы

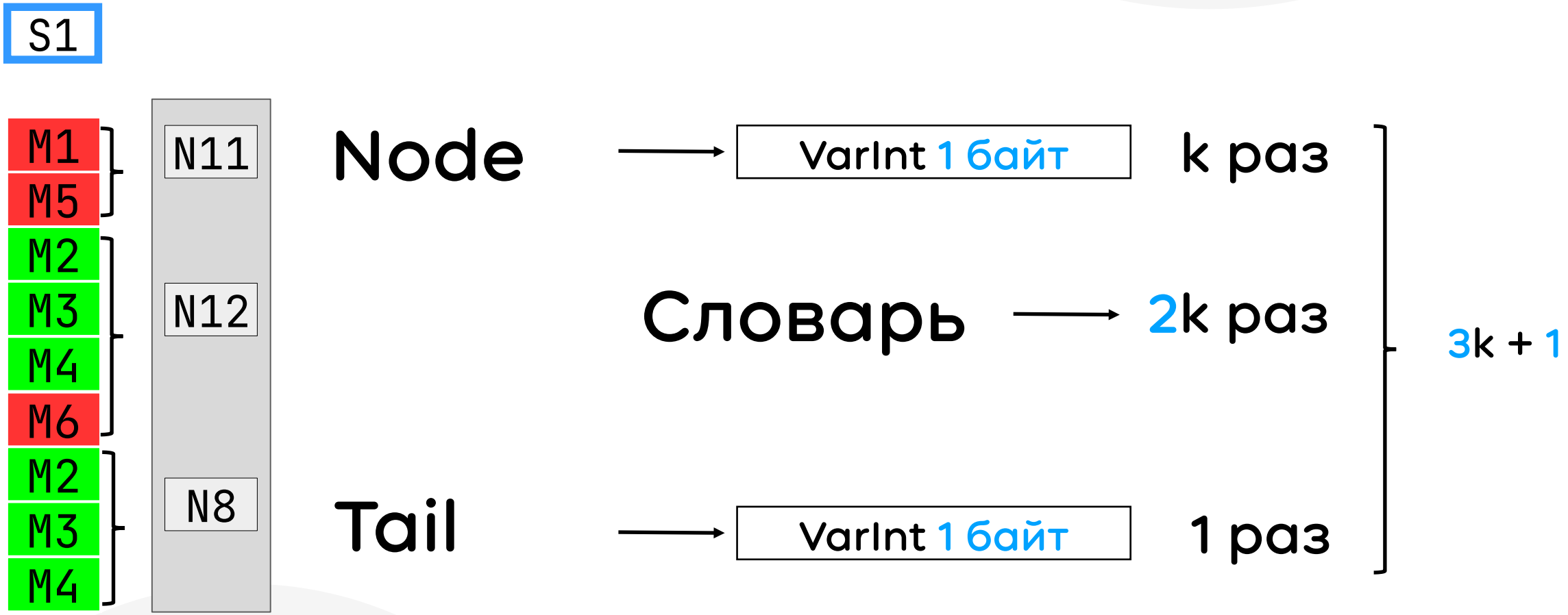


Node → VarInt 1 байт — k раз

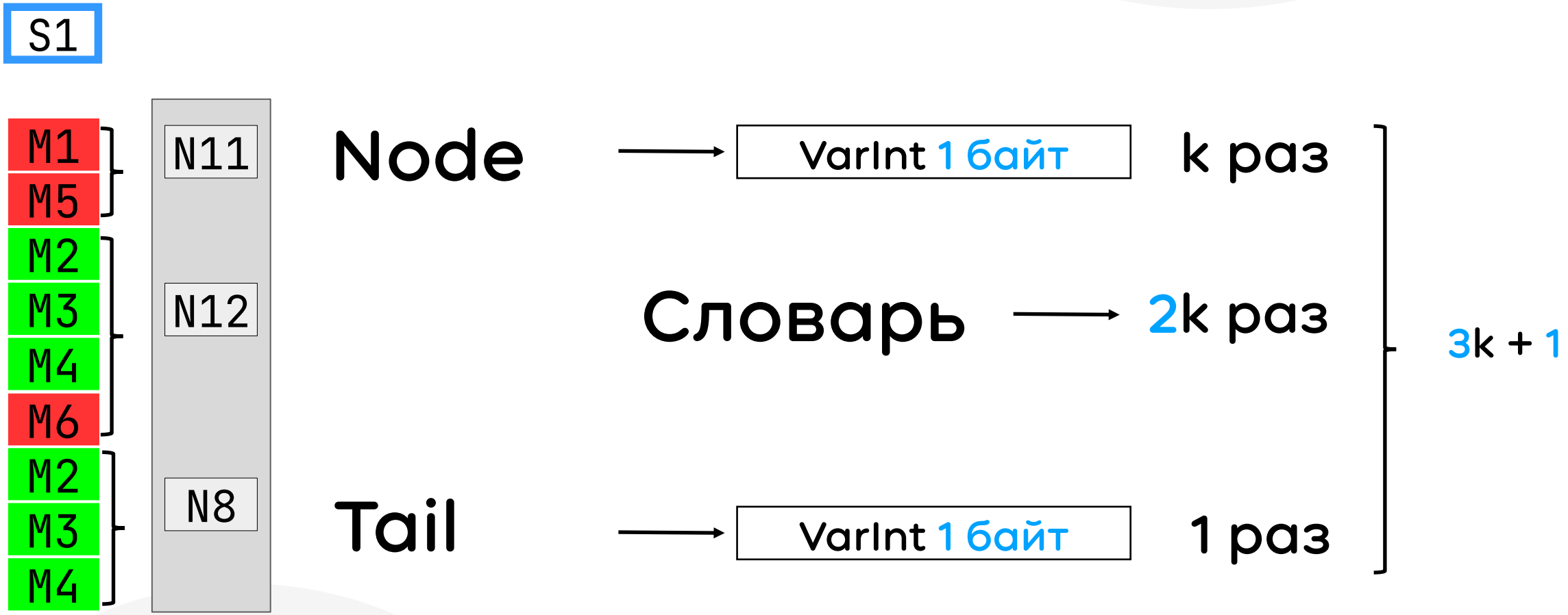
Словарь → 2k раз

Tail → VarInt 1 байт — 1 раз

Сжимаем со всей силы

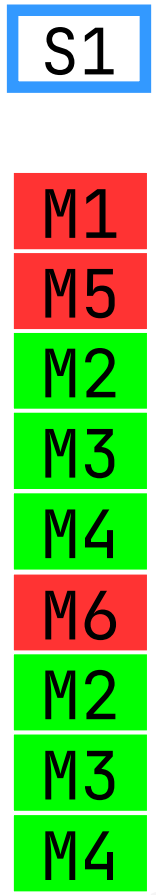


Сжимаем со всей силы



И всё ради одного бита!

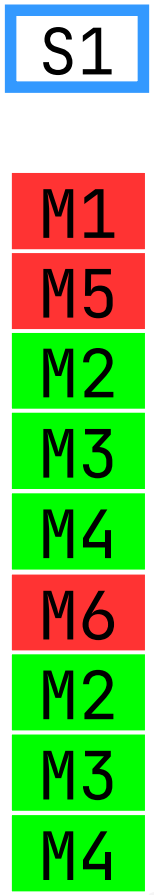
Сжимаем со всей силы



$3k + 1 \sim 1.45$

$k \sim 0.15$

Сжимаем со всей силы



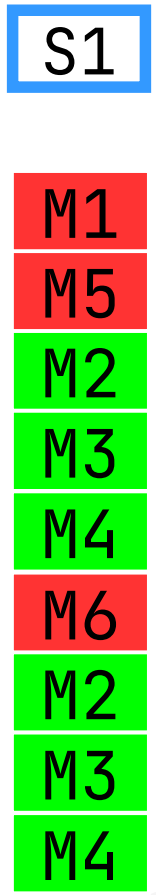
1.45 * 78 млн = 107 МБ

Цель: 170 МБ

3k + 1 ~ 1.45

k ~ 0.15

Сжимаем со всей силы

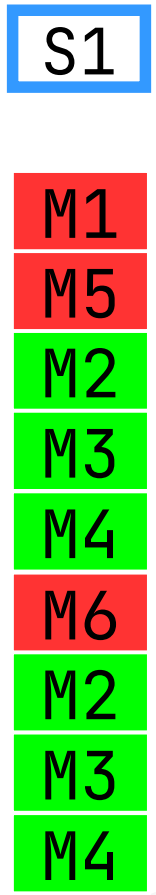


В теории: 107 МБ

Цель: 170 МБ (Alloc)

Реальный результат: 120 МБ (Alloc)

Сжимаем со всей силы



Цель: 170 МБ (Alloc)

Реальный результат: 120 МБ (Alloc)

Цель: 30 МБ (CPU)

Реальный результат: 20 МБ (CPU)

Сжимаем со всей силы

S1

M1
M5
M2
M3
M4
M6
M2
M3
M4

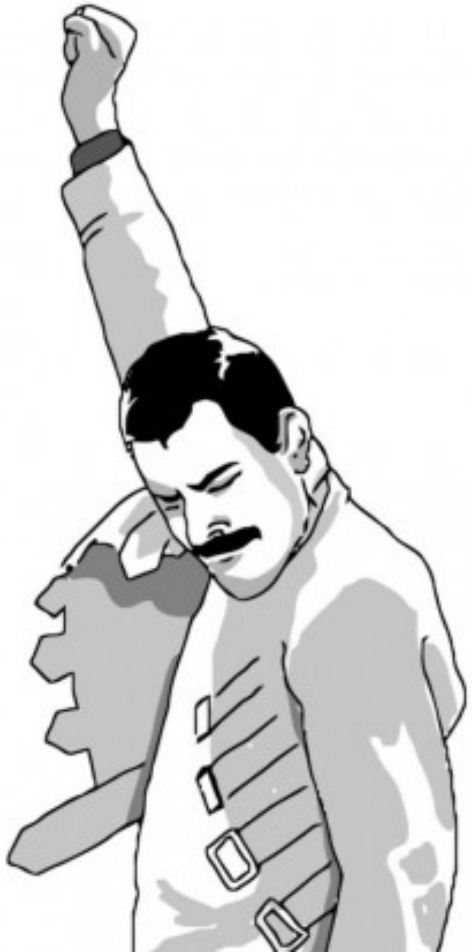
Цель: 170 МБ (Alloc)

Реальный результат: 120 МБ (Alloc)

Цель: 30 МБ (CPU)

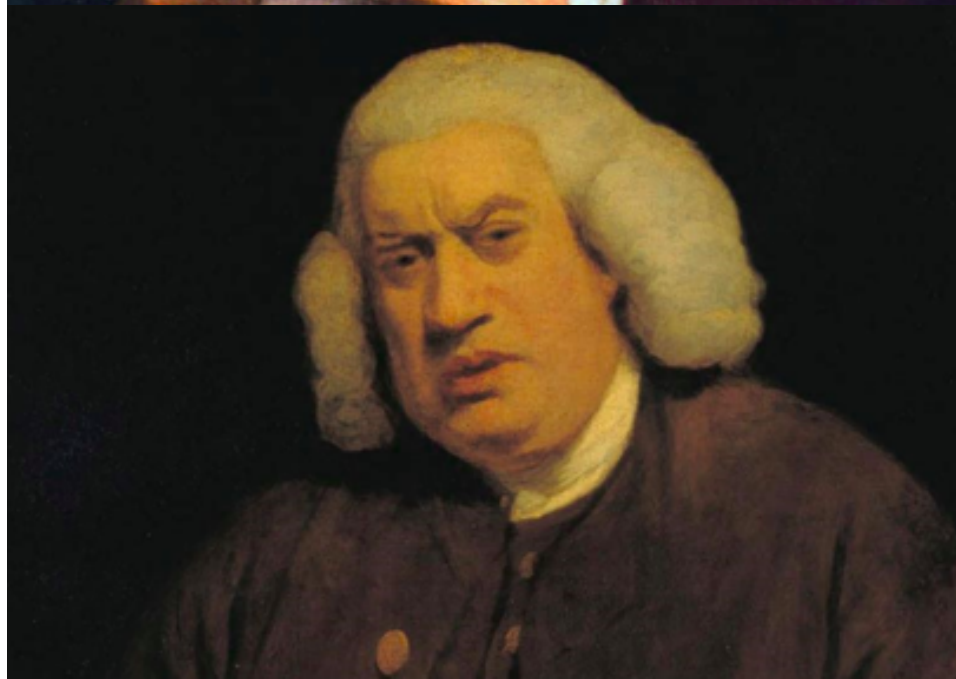
Реальный результат: 20 МБ (CPU)

Исходные данные: 4 ГБ



Данные готовы

Перерыв на вопросы



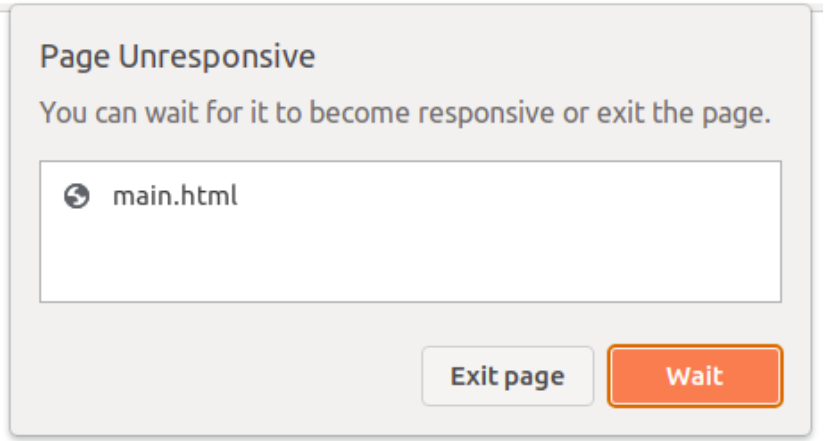
Распаковка

Распаковка

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов

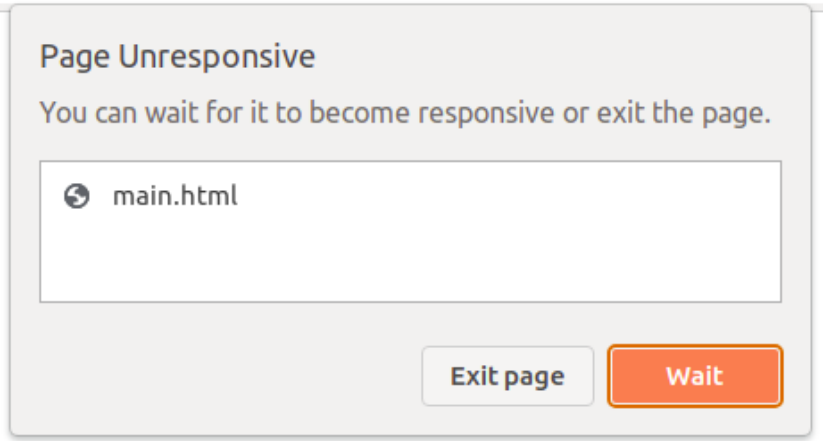
Распаковка

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов



Распаковка

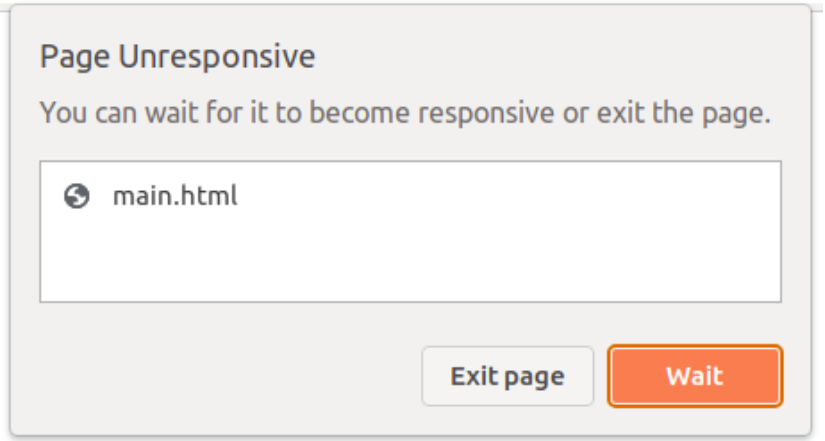
- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов



CPU → 10 сек

Распаковка

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов

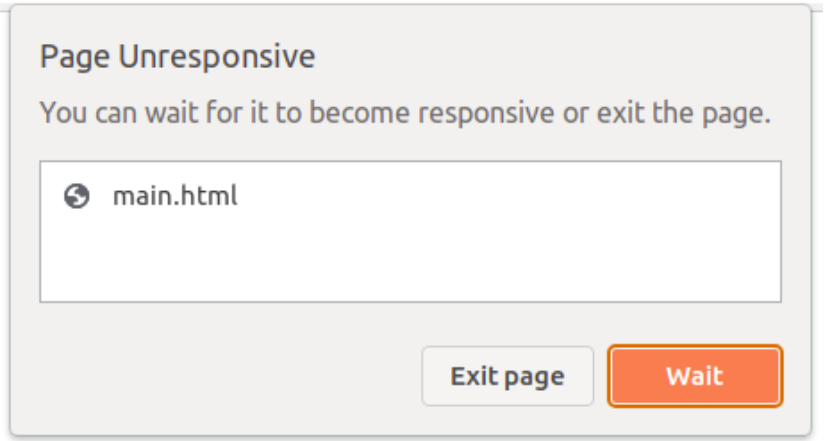


CPU → 10 сек

Alloc → 20 сек

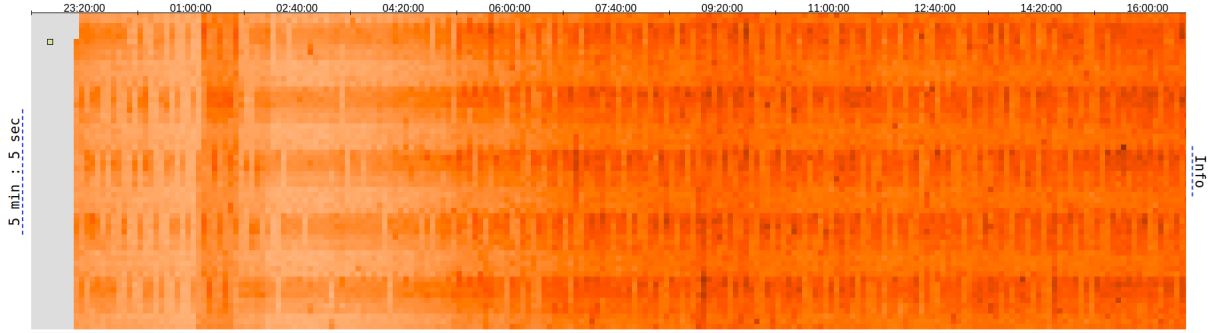
Распаковка

- JFR: 4 ГБ
- CPU: 6 млн сэмплов
- Alloc: 78 млн сэмплов

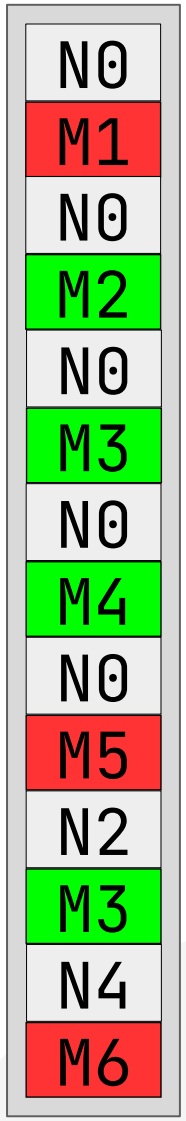


CPU → 10 сек

Alloc → 20 сек



Распаковка словаря

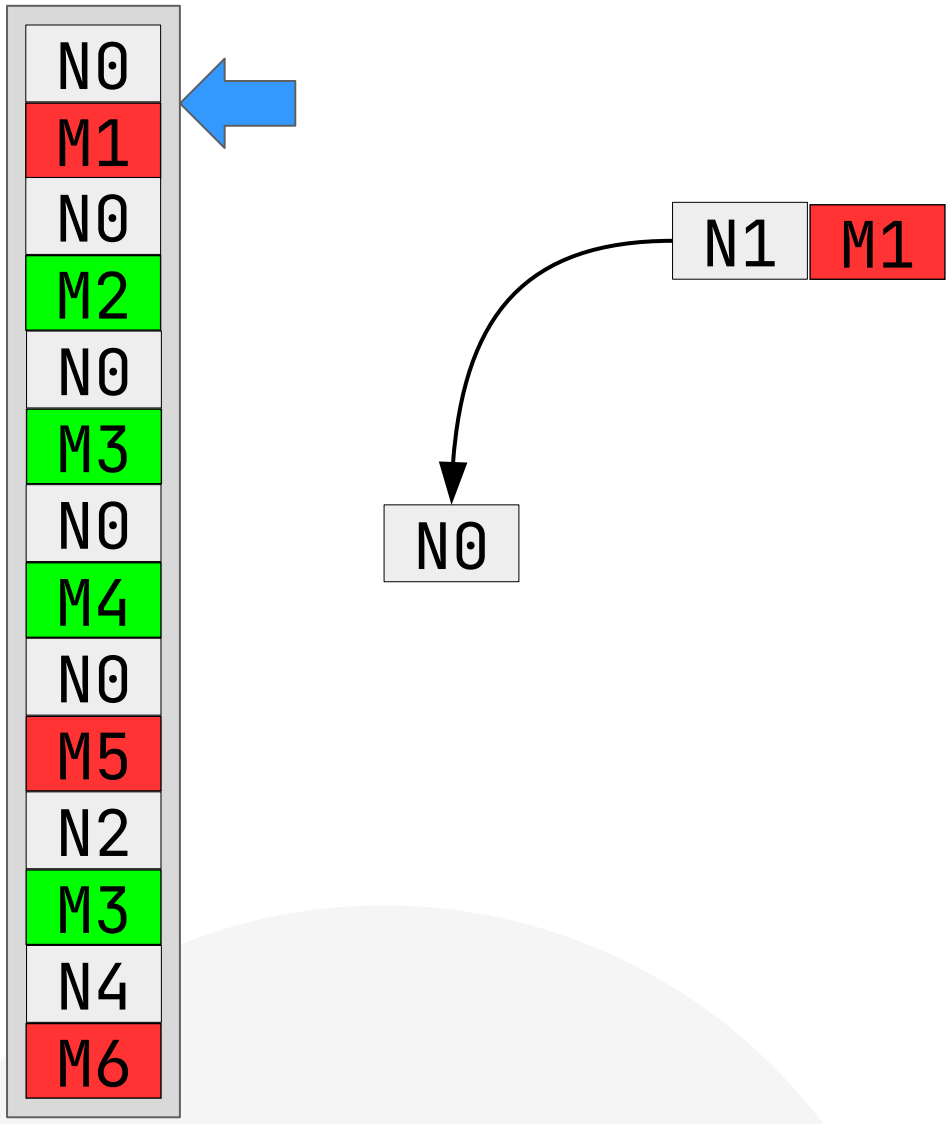


N0

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

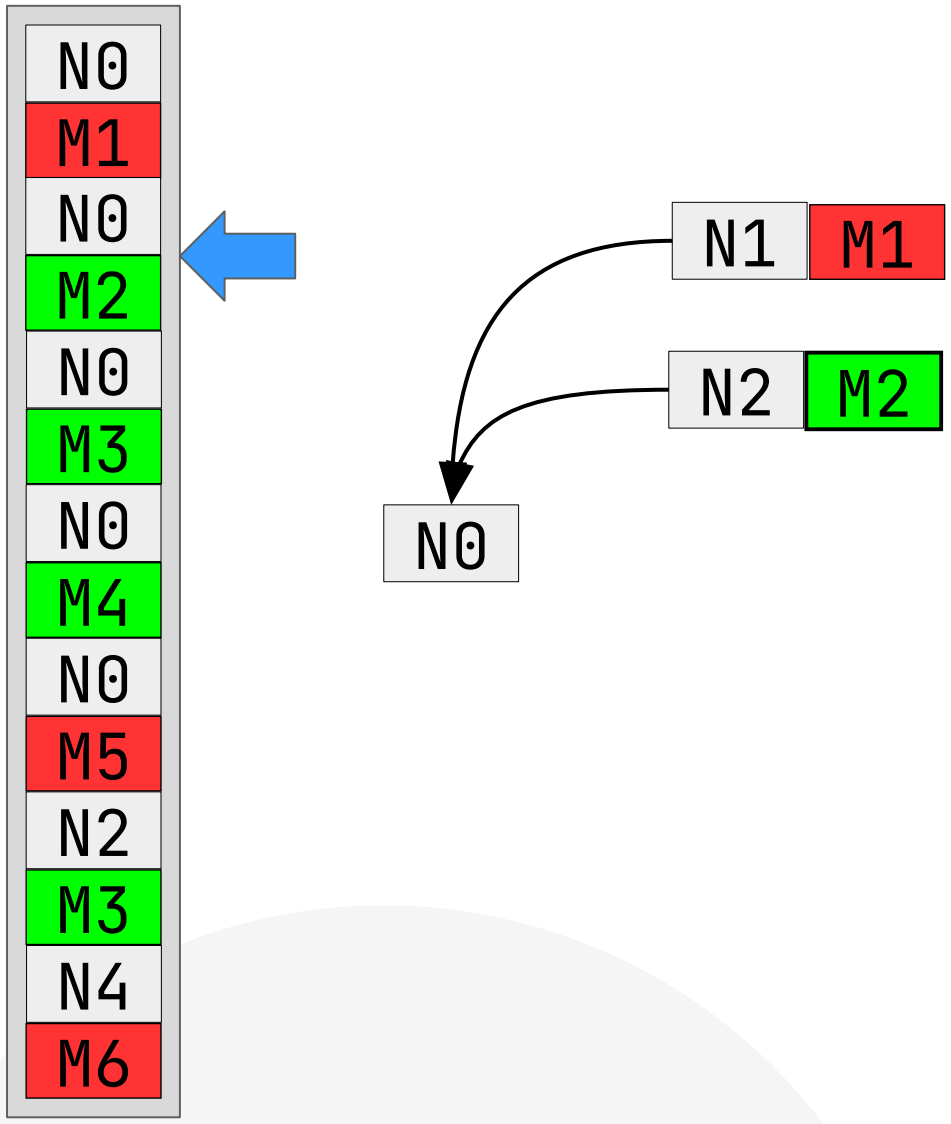
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

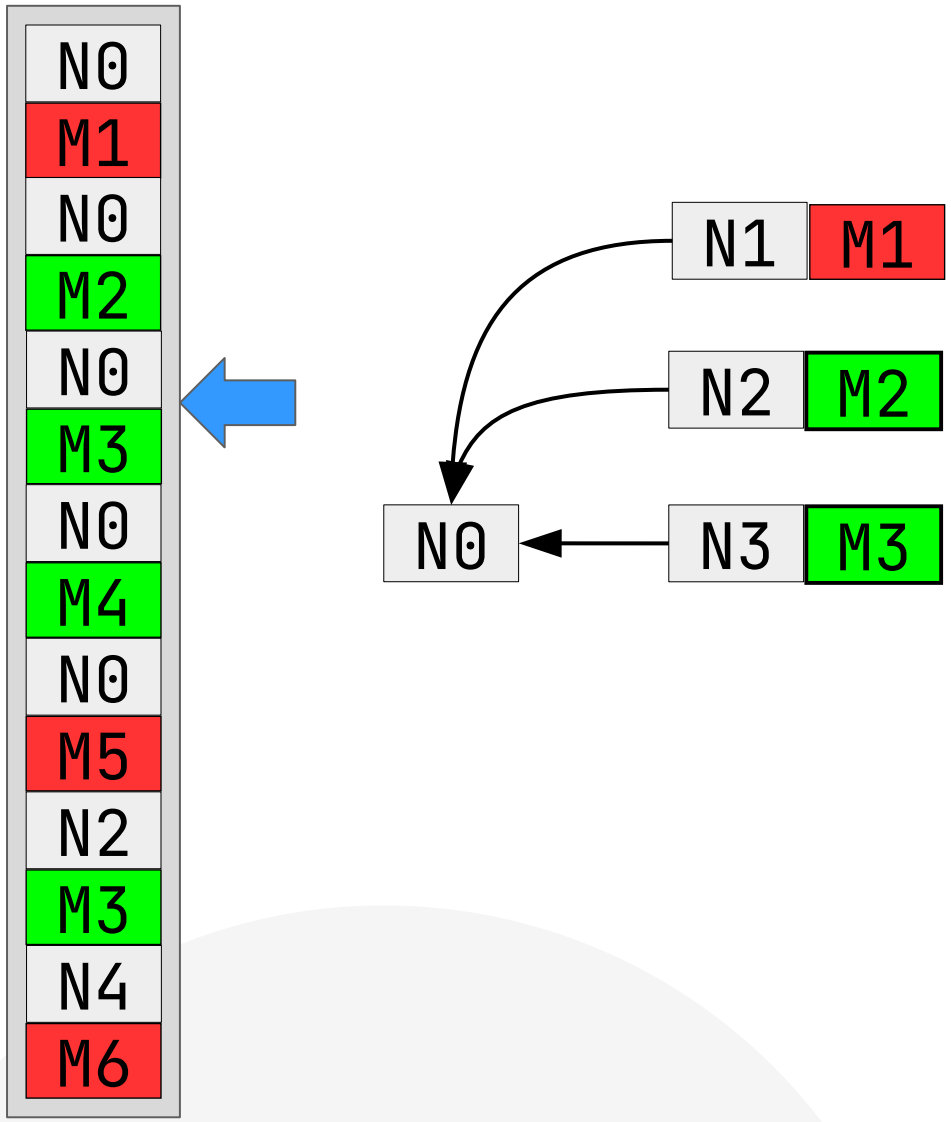
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

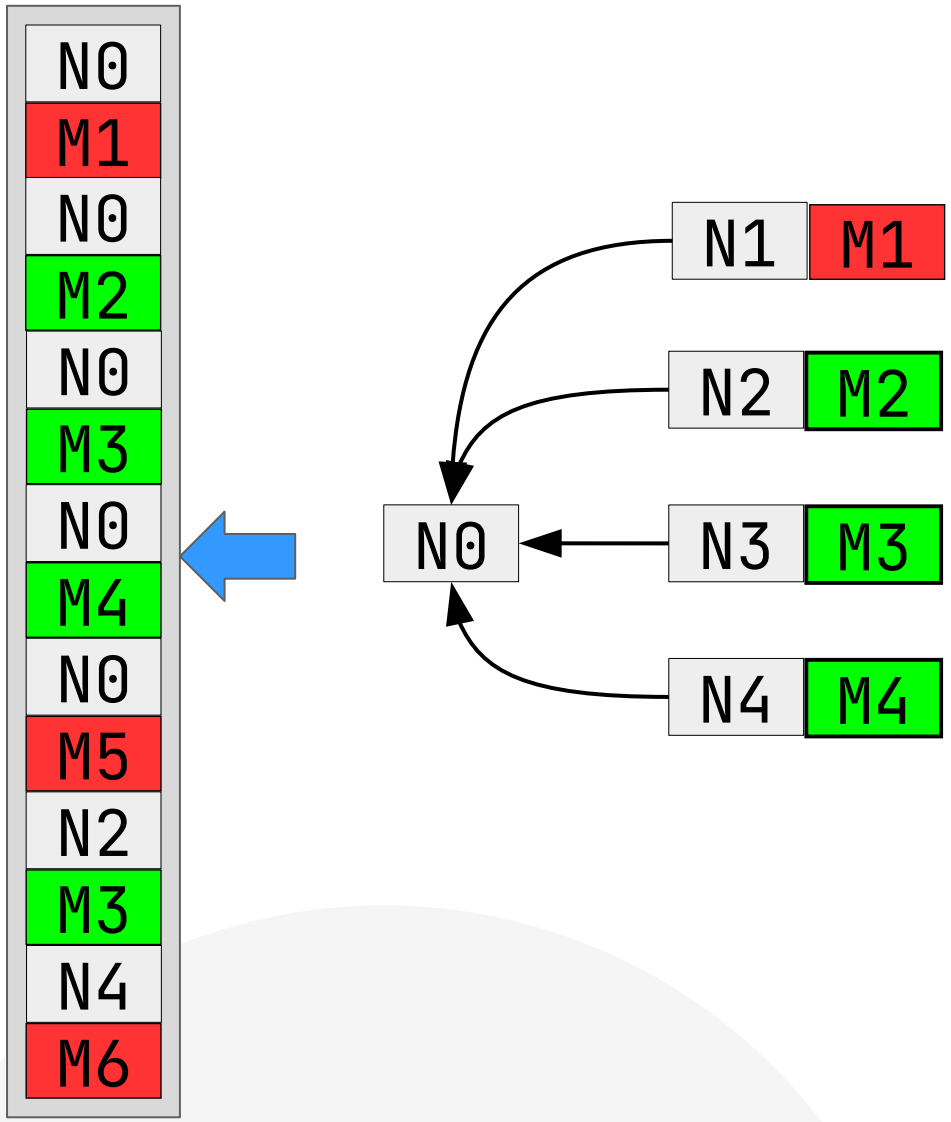
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

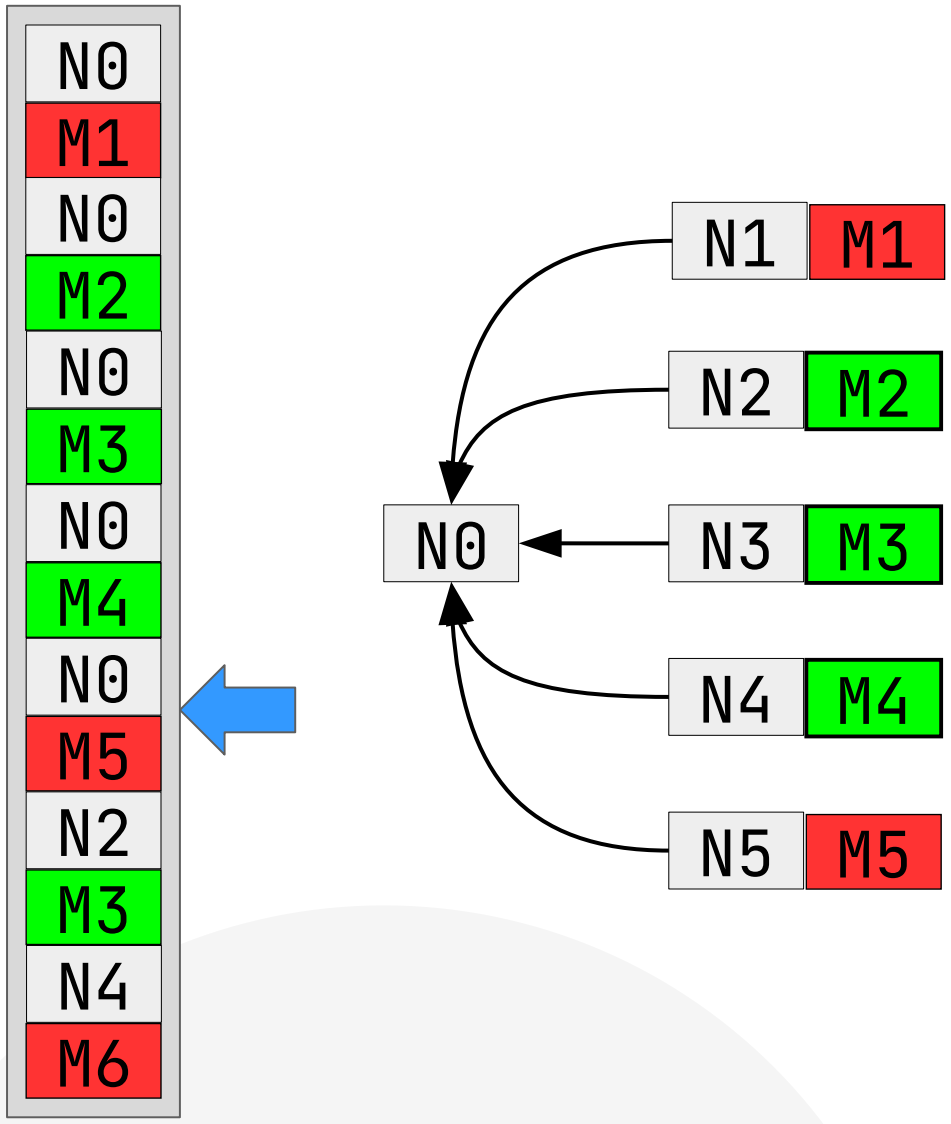
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

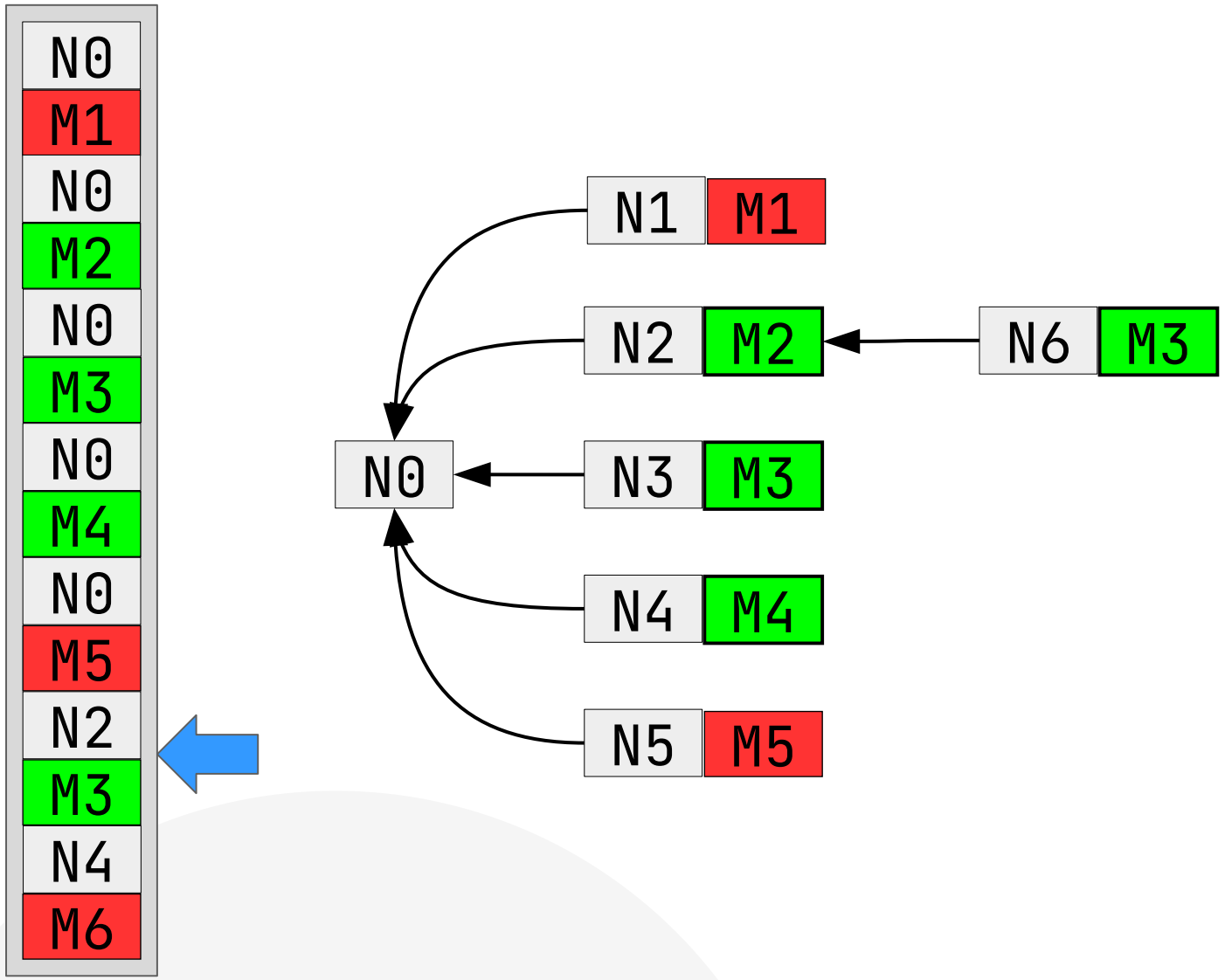
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

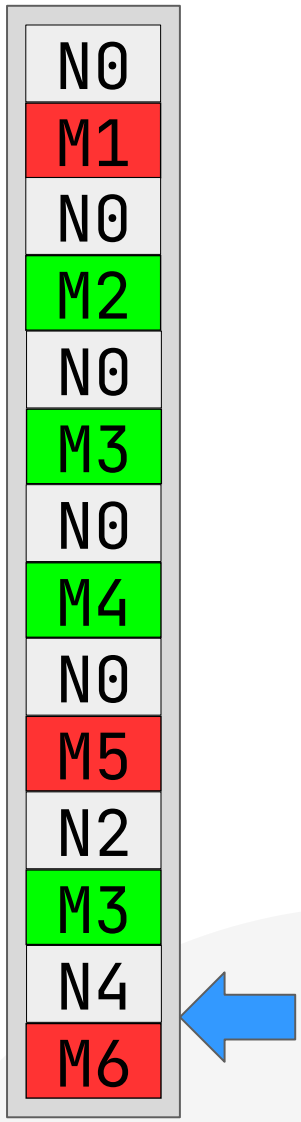
Распаковка словаря

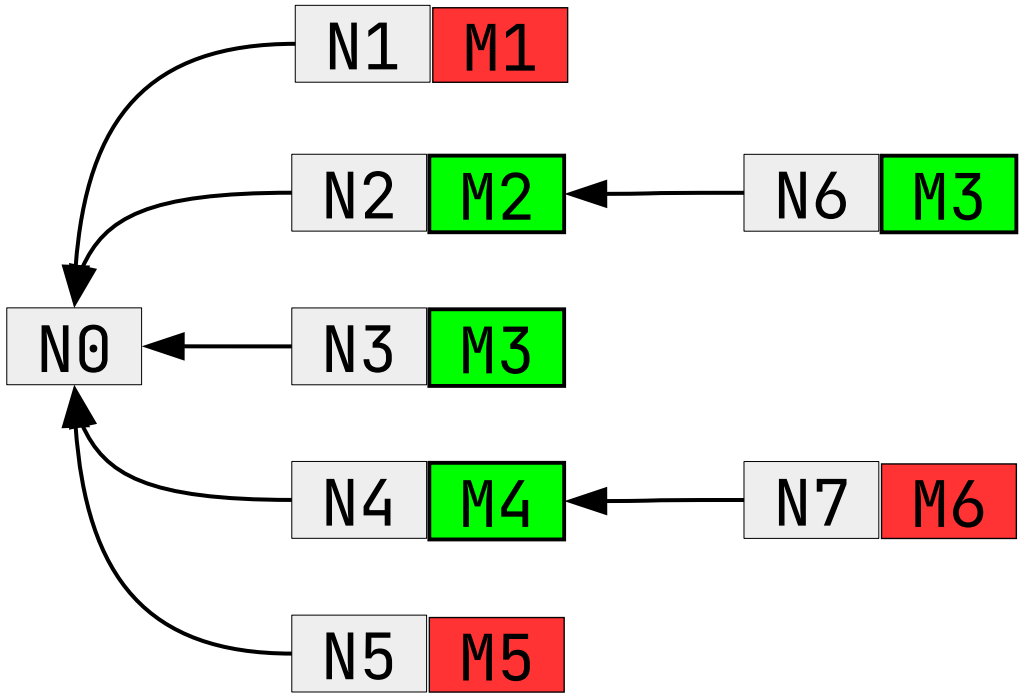


Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

Распаковка словаря

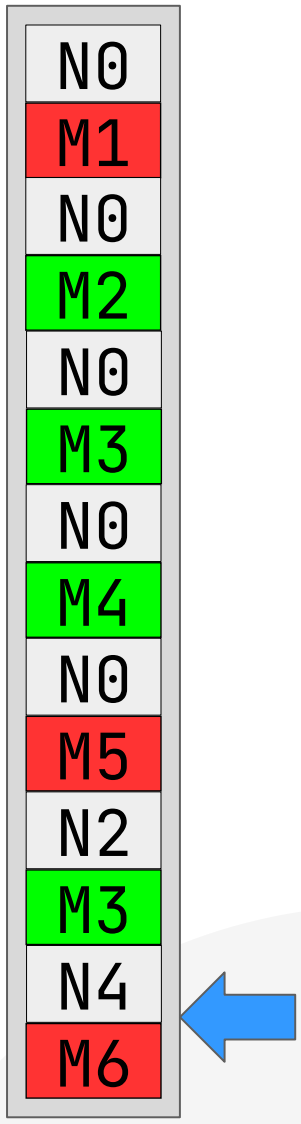


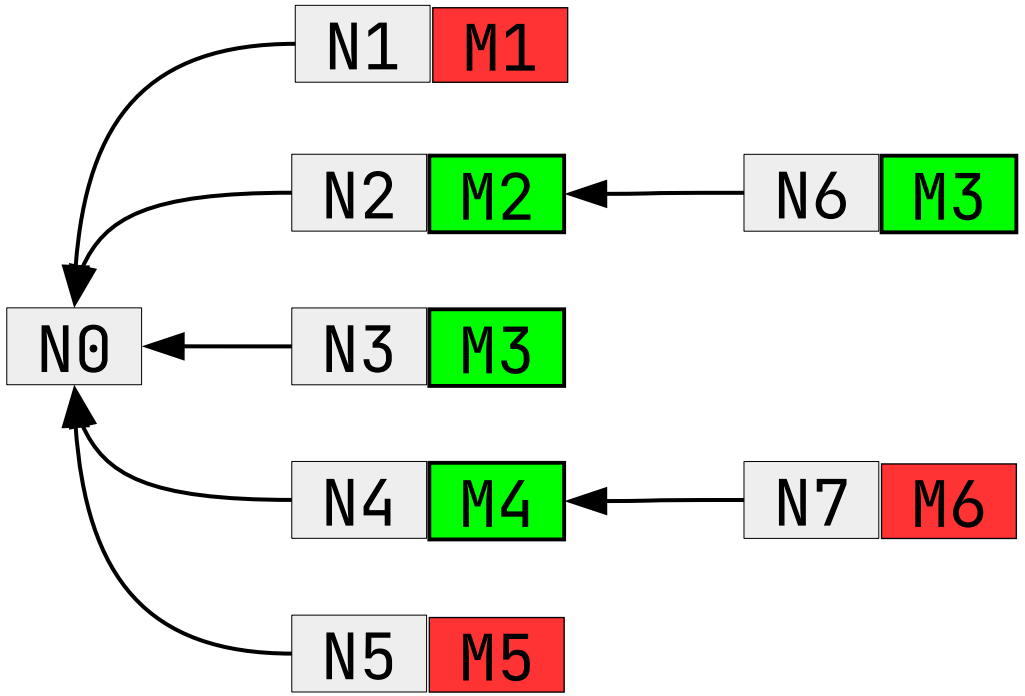


Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
```

Распаковка словаря





Опасно для психического здоровья!

Опасно для психического здоровья!

Распаковка словаря

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```

Распаковка словаря

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
   const parentId = data.nextVarInt();
   const methodId = data.nextVarInt();

   lz78Nodes[i] = {
       parent: lz78Nodes[parentId],
       method: methodId
   }
}
```

O(N)?

Распаковка словаря

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```

O(N*N)!

Распаковка словаря

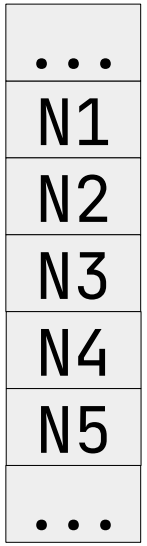
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```

O(N*N)!

* при достаточном натяжении совы

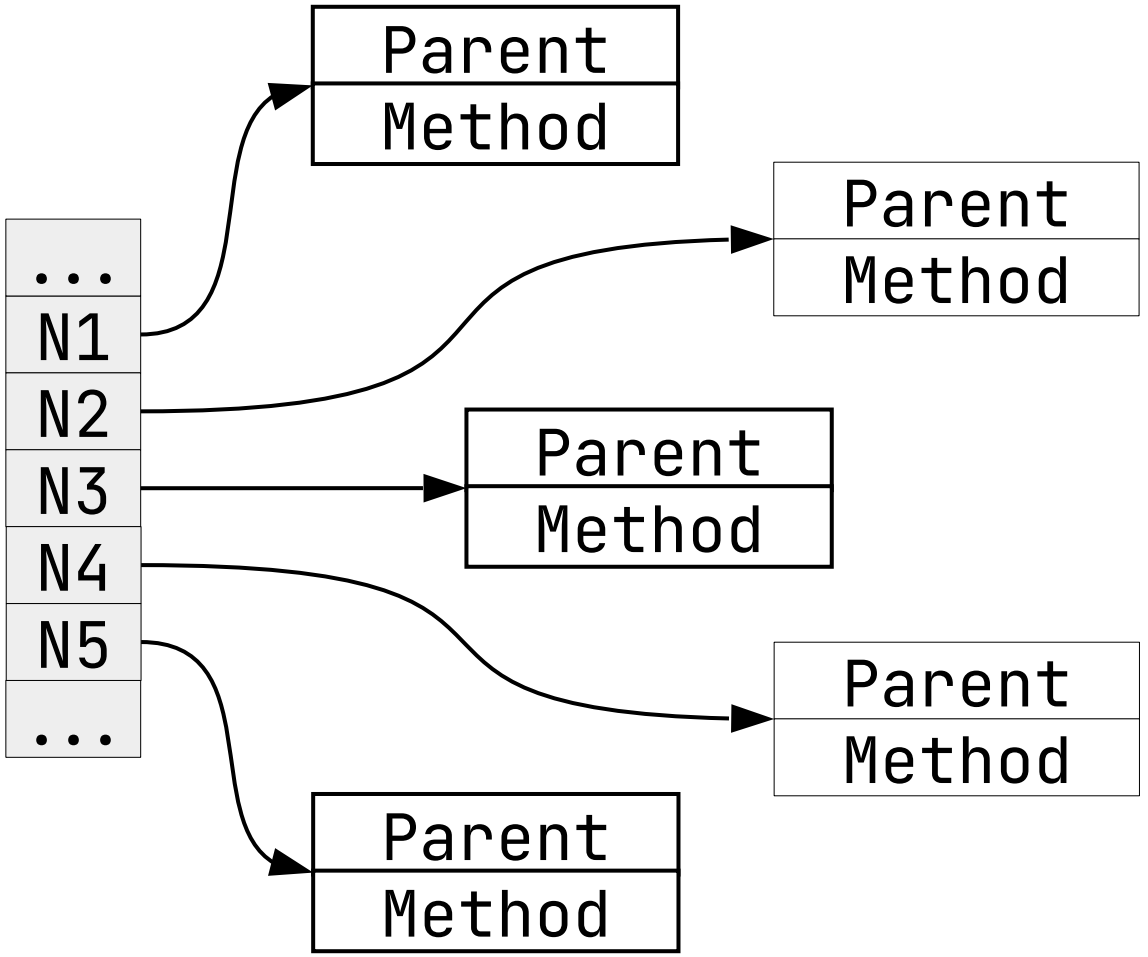
Распаковка словаря

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



Распаковка словаря

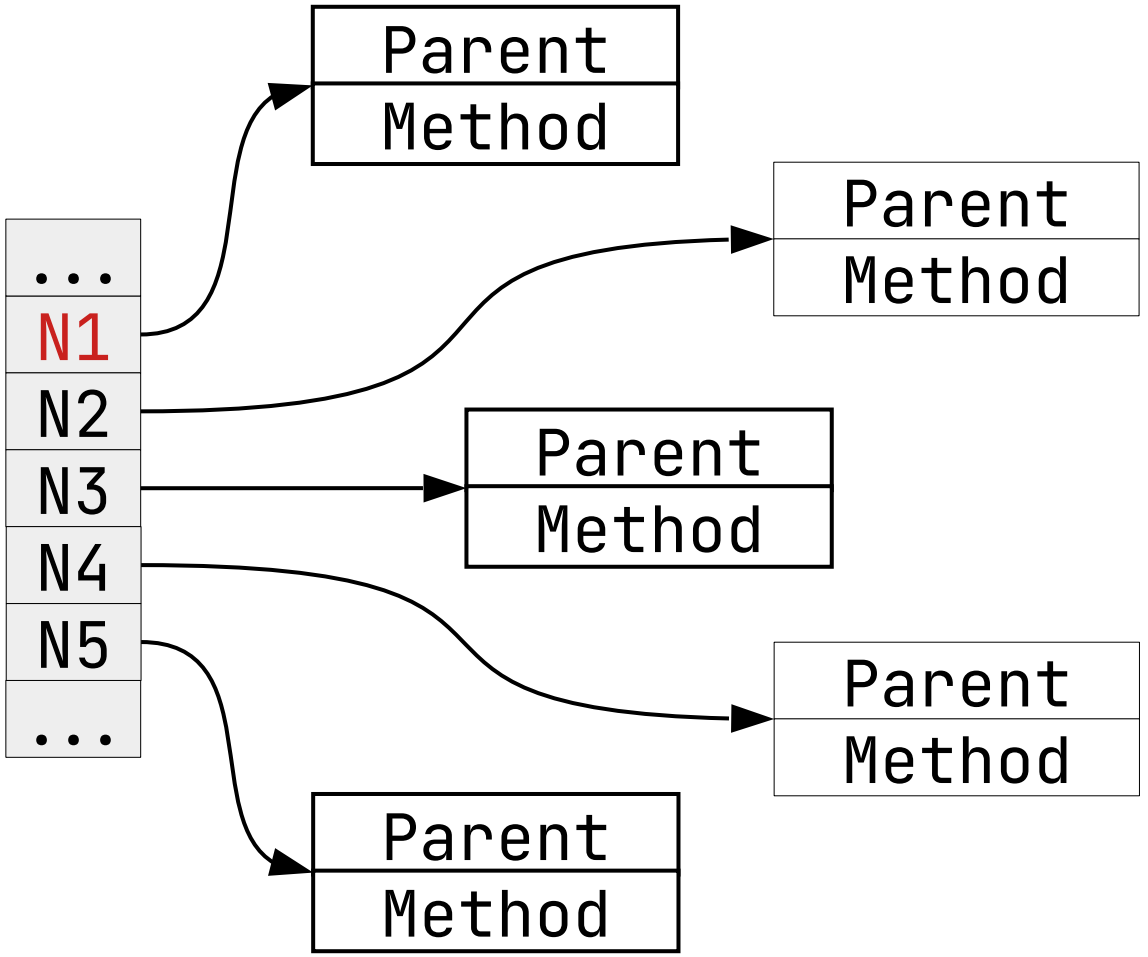
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

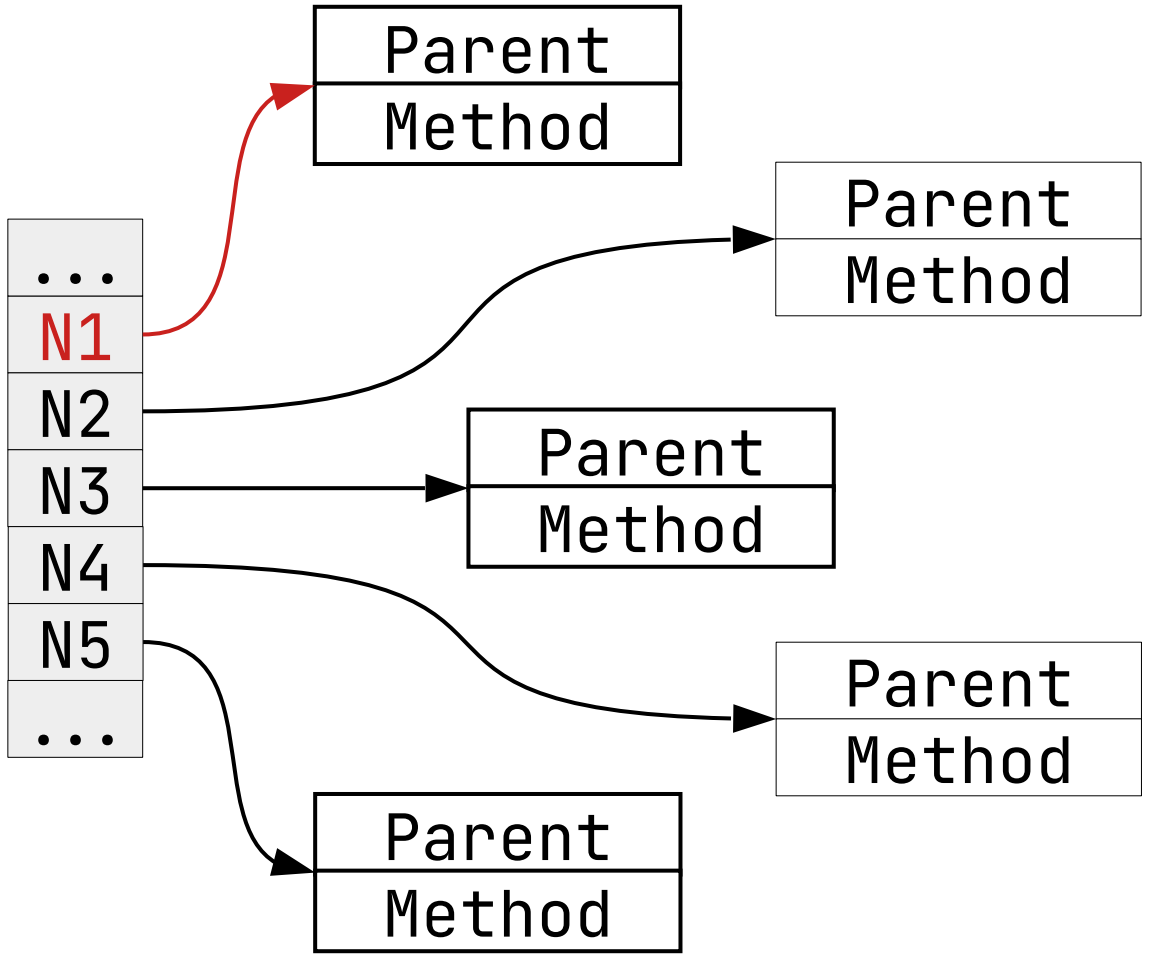
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

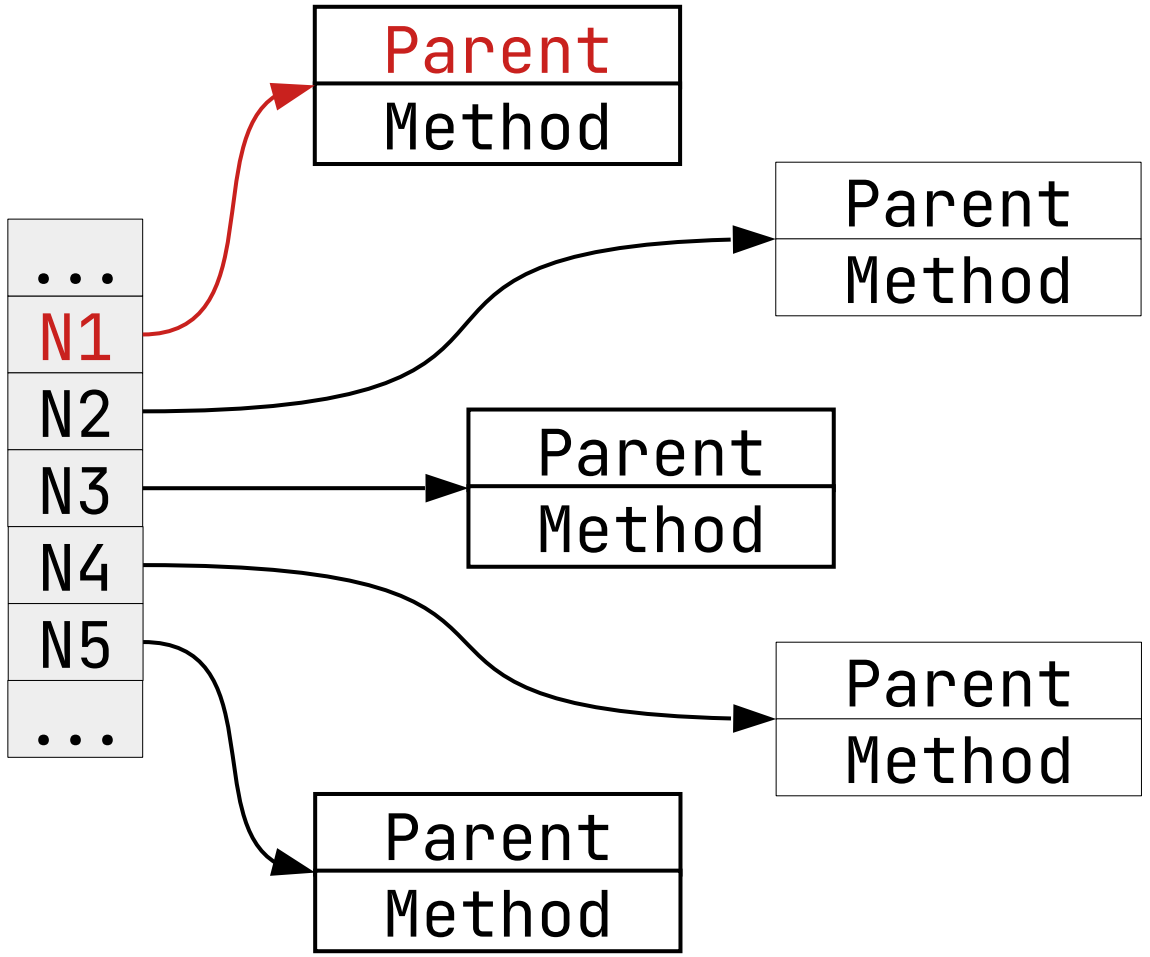
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

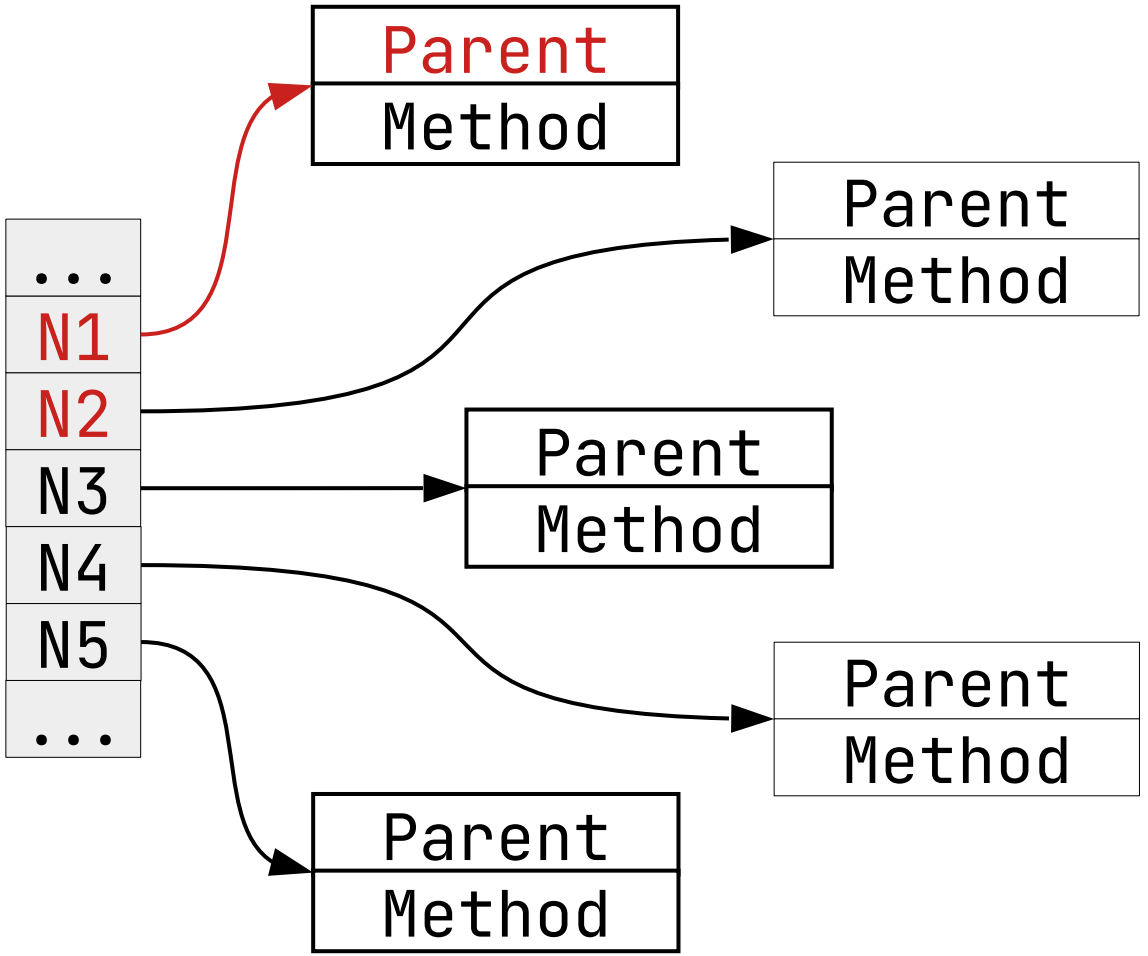
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

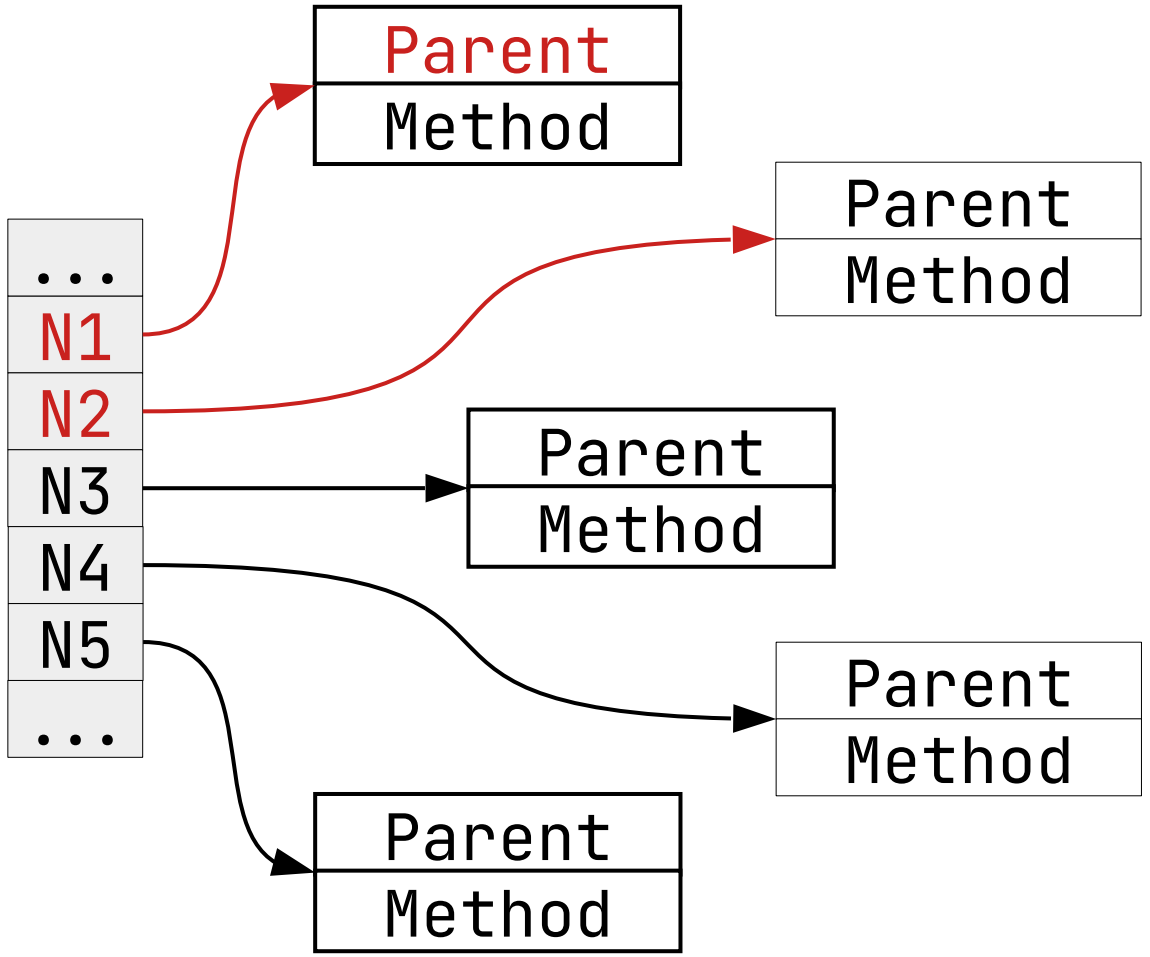
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

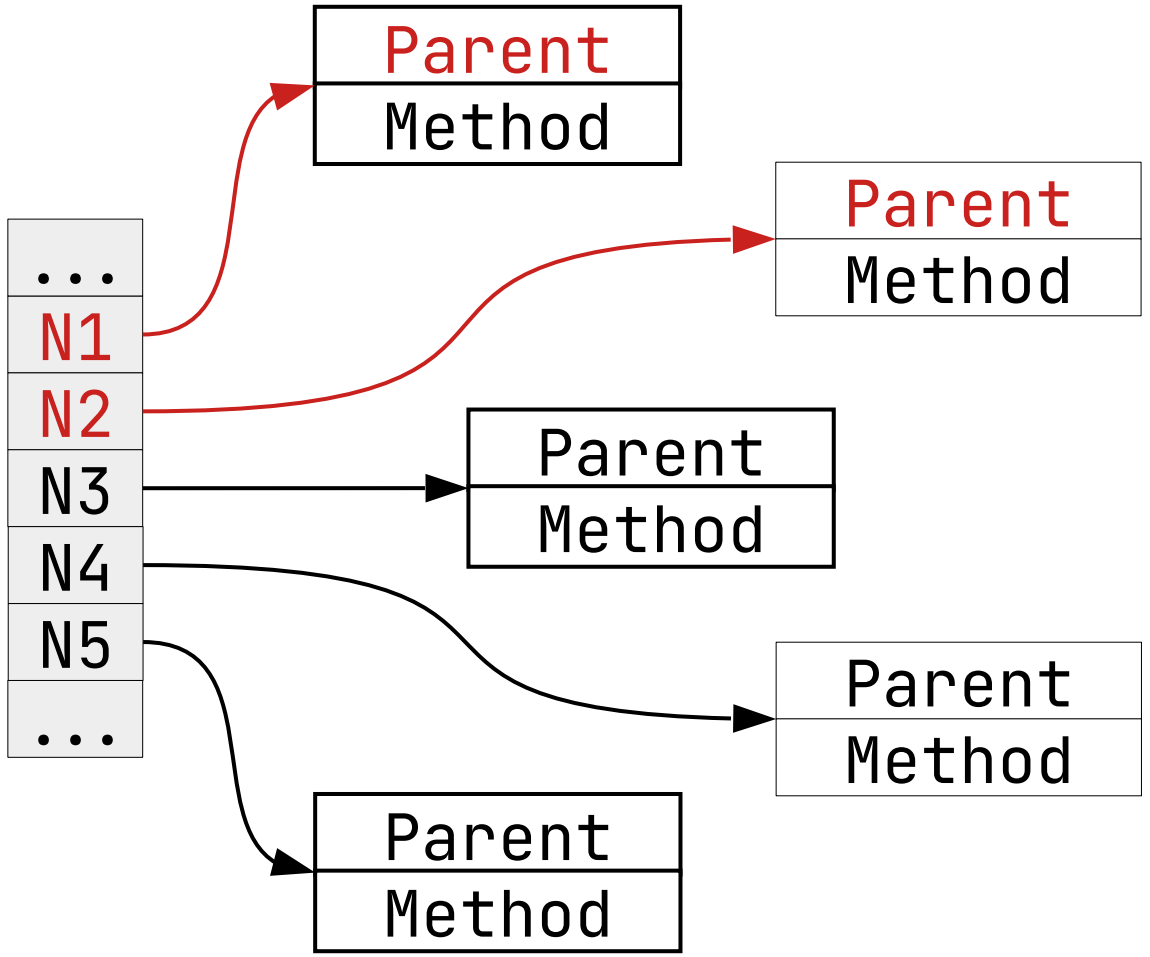
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

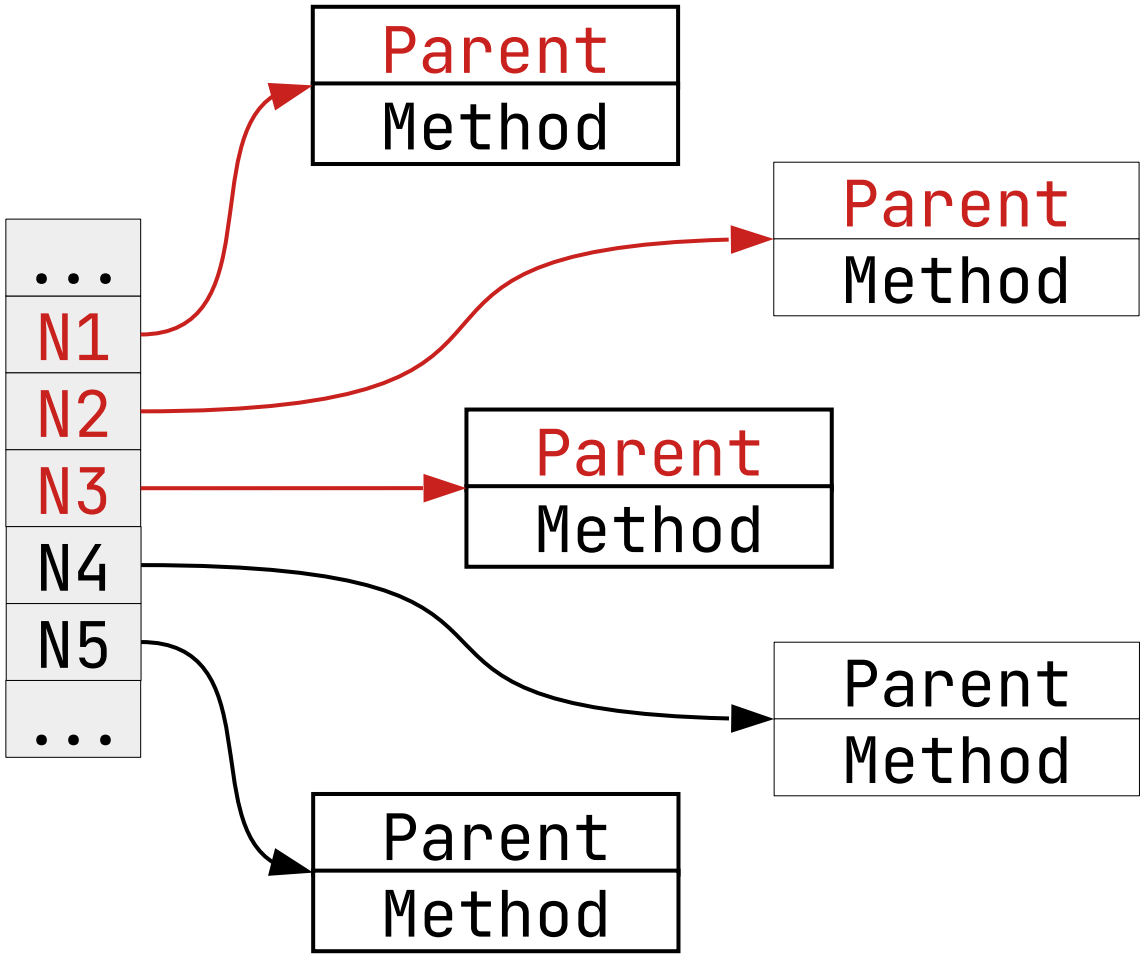
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

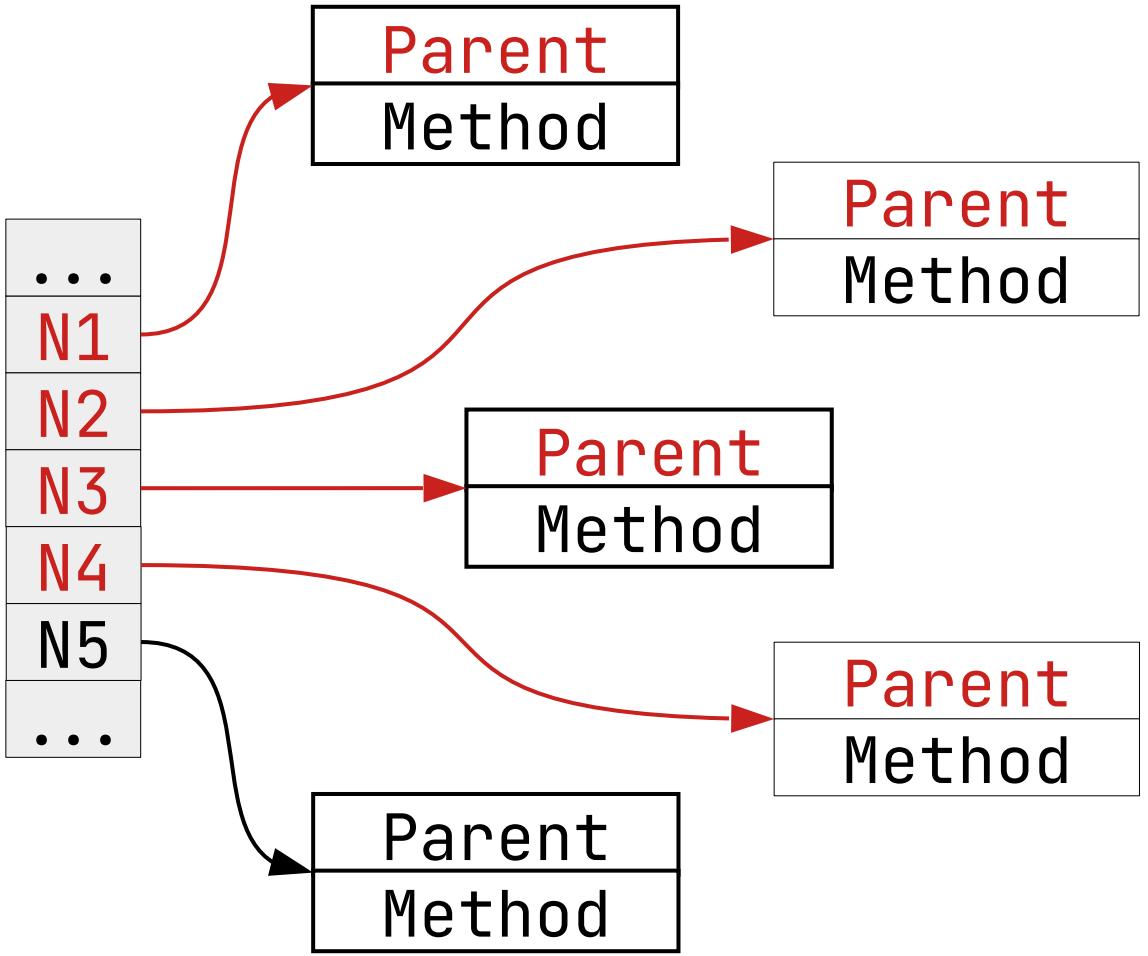
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

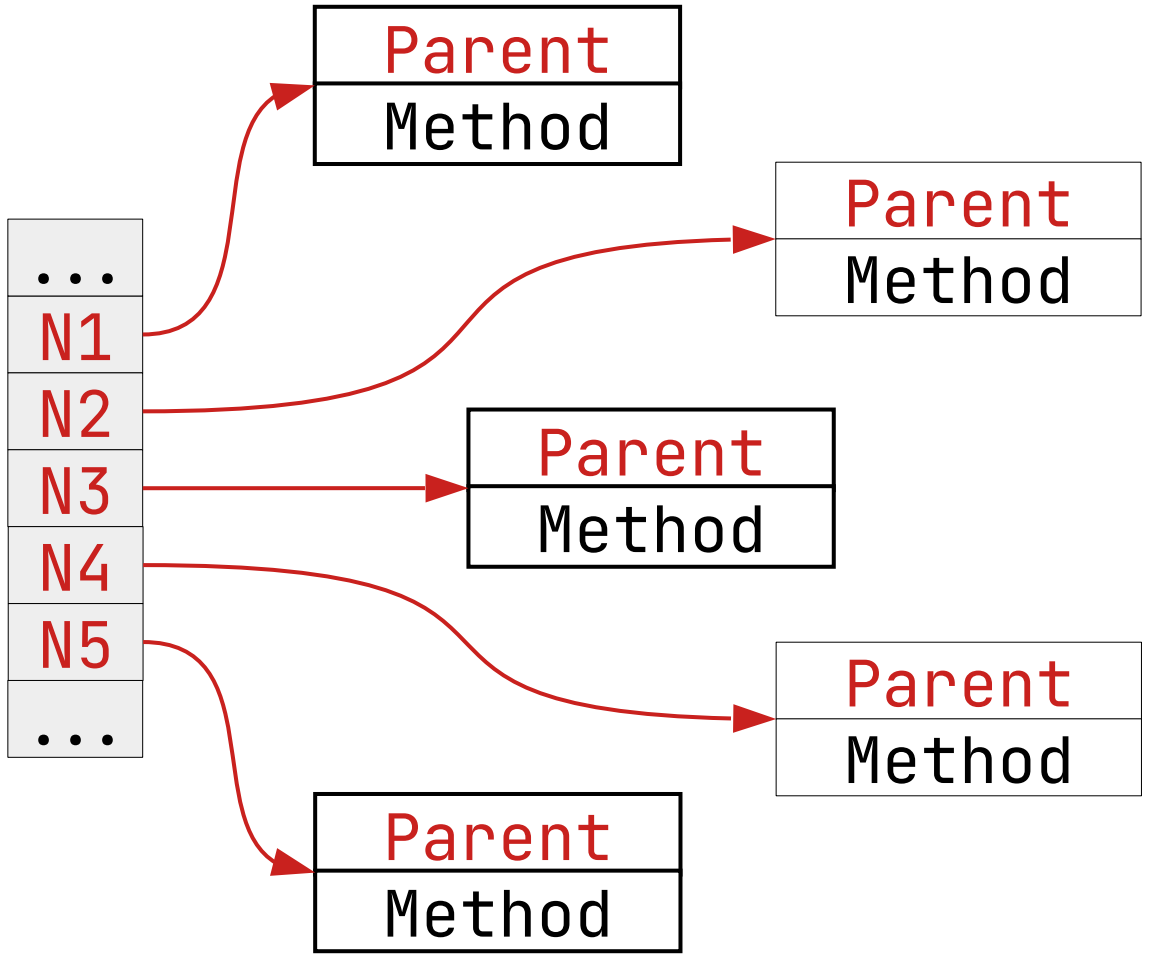
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```



* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```

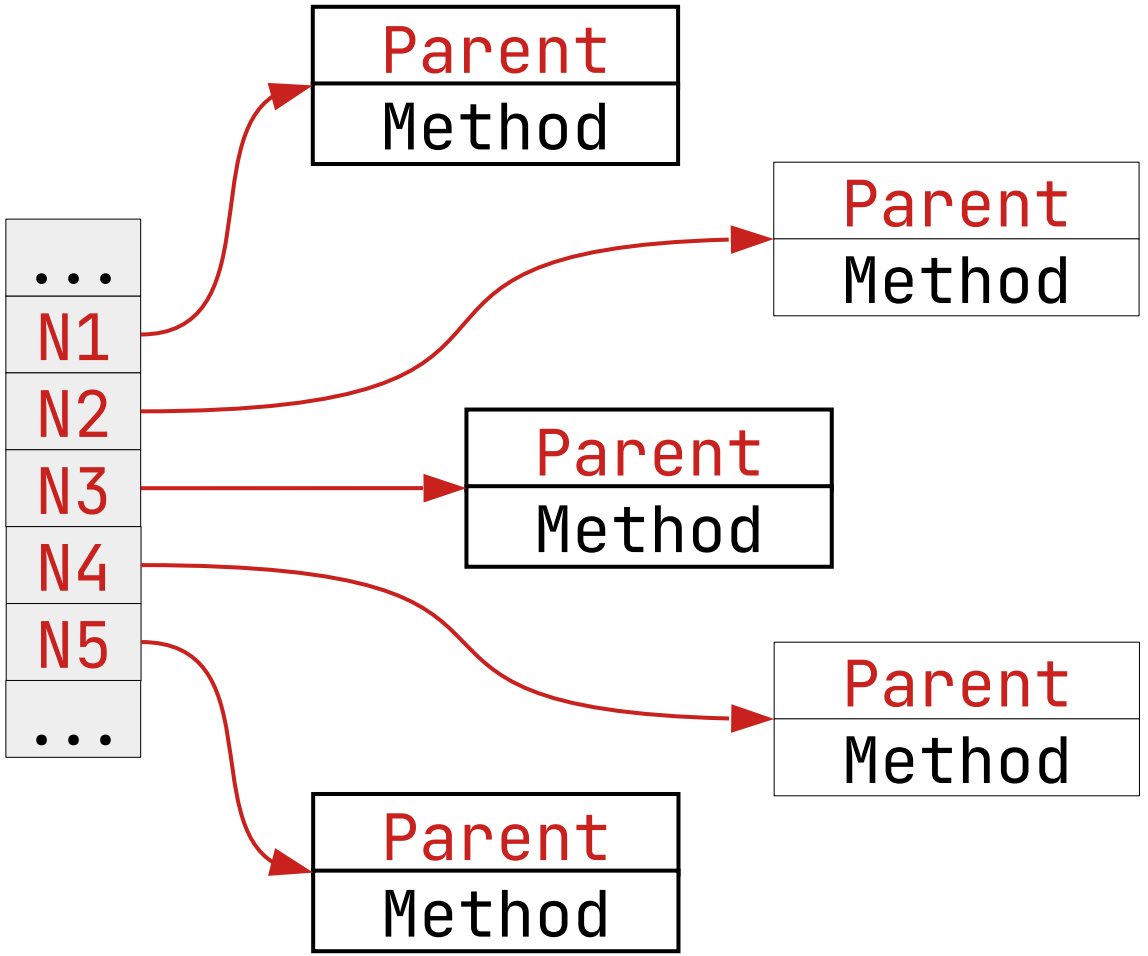


* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```

CPU → 10 сек

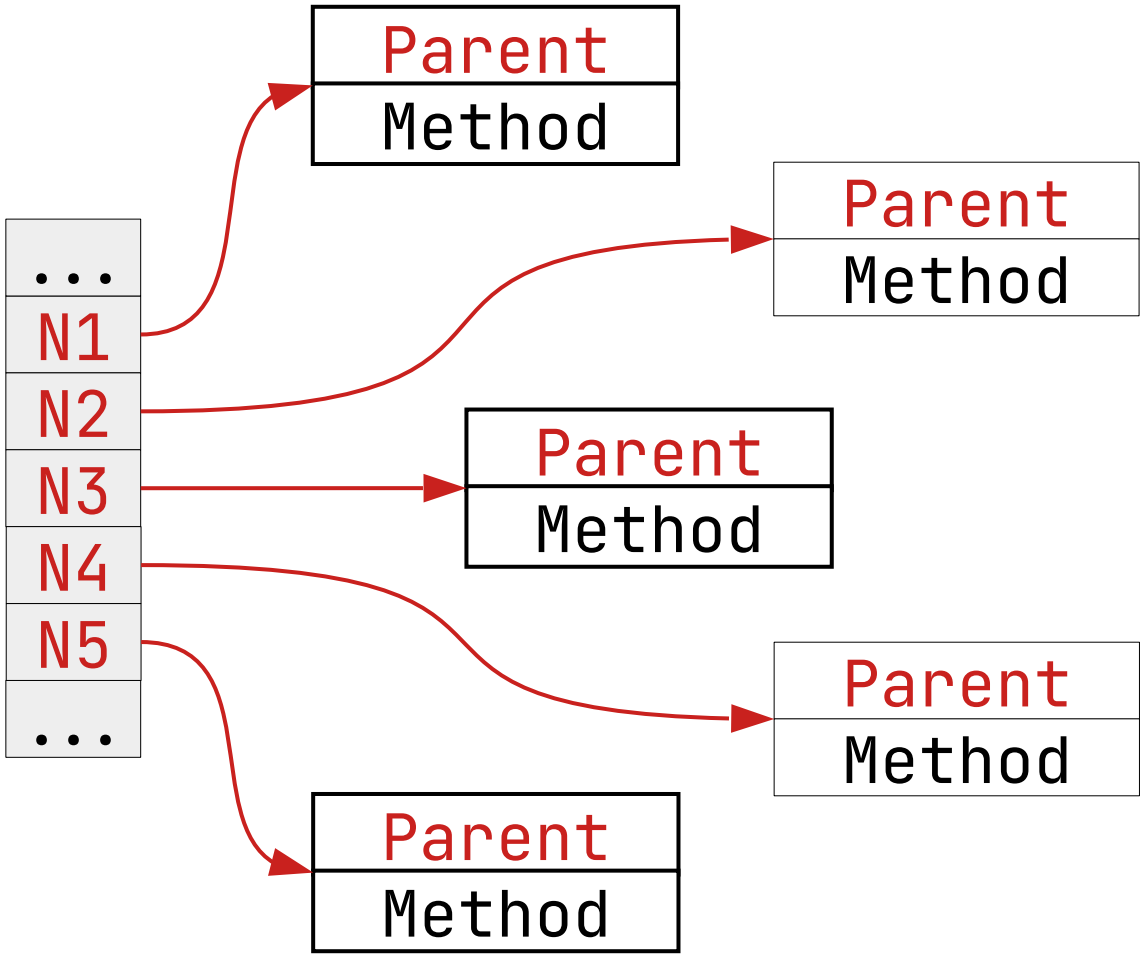


* https://deepu.tech/memory-management-in-v8/

Распаковка словаря

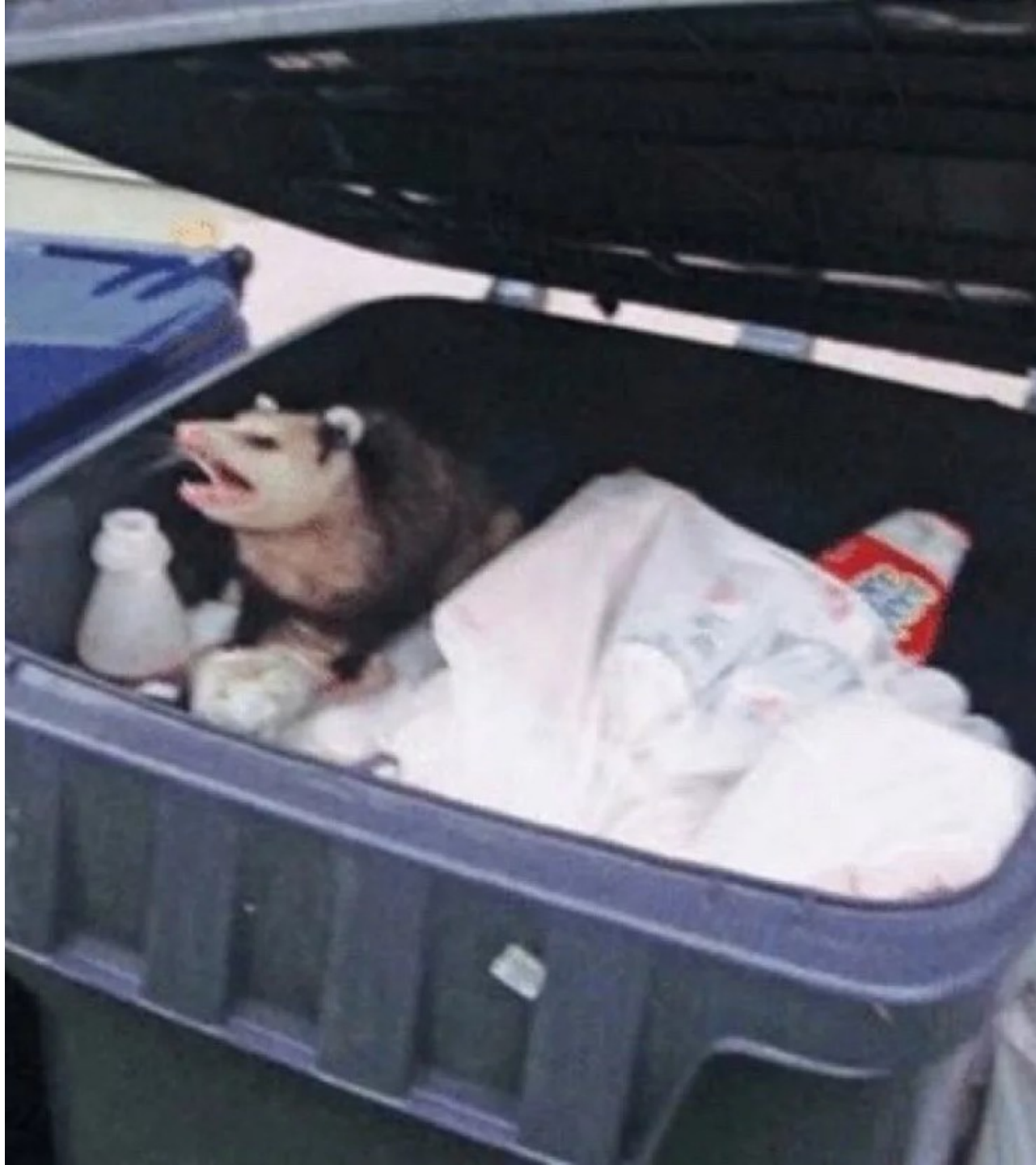
```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```

CPU → 15 сек



* https://deepu.tech/memory-management-in-v8/

Не трожь мой мусор!



GC страдает

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```

GC страдает

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
   const parentId = data.nextVarInt();
   const methodId = data.nextVarInt();

   lz78Nodes[i] = {
       parent: lz78Nodes[parentId],
       method: methodId
   }
}
```

GC доволен

```
const lz78OwnMethods = new Uint32Array(...);
const lz78Parents = new Uint32Array(...);

for (let i = 1; i < lz78RecordsCount; i++) {
  const parentId = data.nextVarInt();
  const methodId = data.nextVarInt();


  lz78Parents[i] = parentId;
  lz78OwnMethods[i] = methodId;

}
```

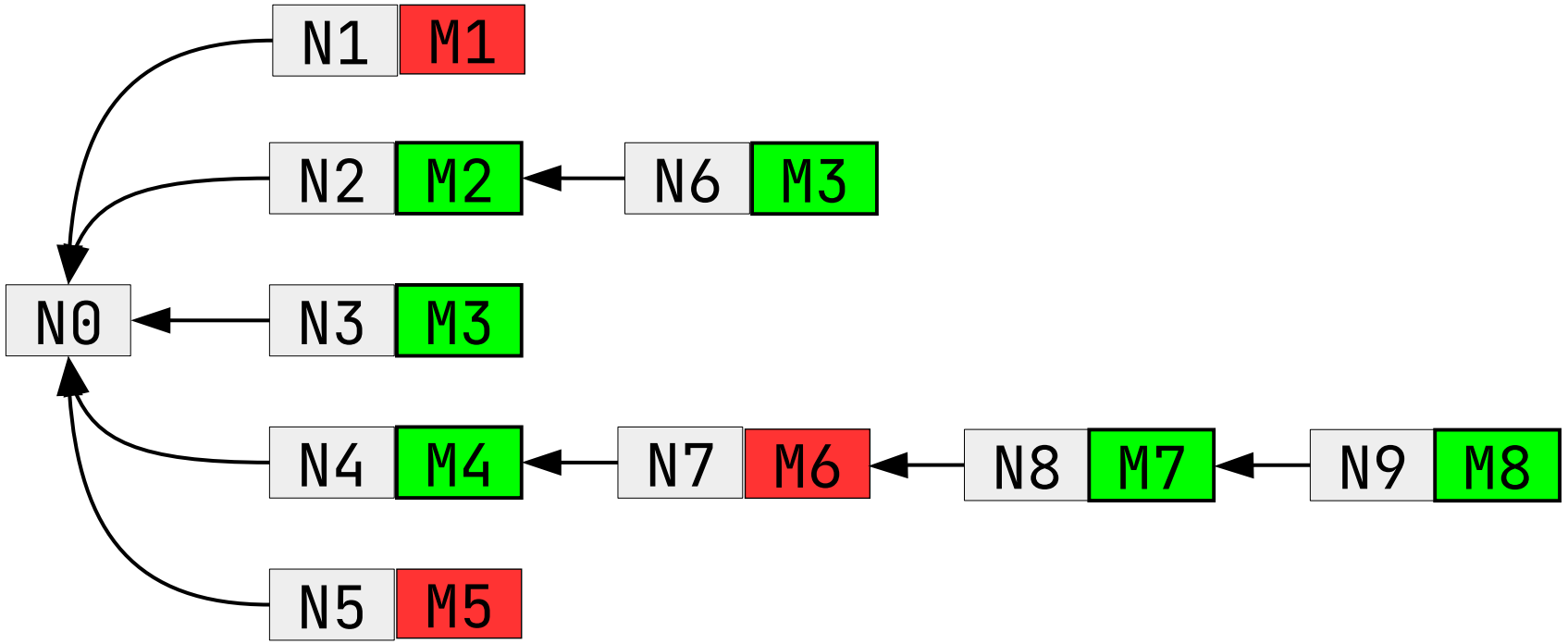
GC страдает

GC доволен

```
const lz78Nodes = new Array(lz78RecordsCount);
lz78Nodes[0] = null;

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Nodes[i] = {
        parent: lz78Nodes[parentId],
        method: methodId
    }
}
```

```
const lz78OwnMethods = new Uint32Array(...);
const lz78Parents = new Uint32Array(...);

for (let i = 1; i < lz78RecordsCount; i++) {
    const parentId = data.nextVarInt();
    const methodId = data.nextVarInt();

    lz78Parents[i] = parentId;
    lz78OwnMethods[i] = methodId;

}
```

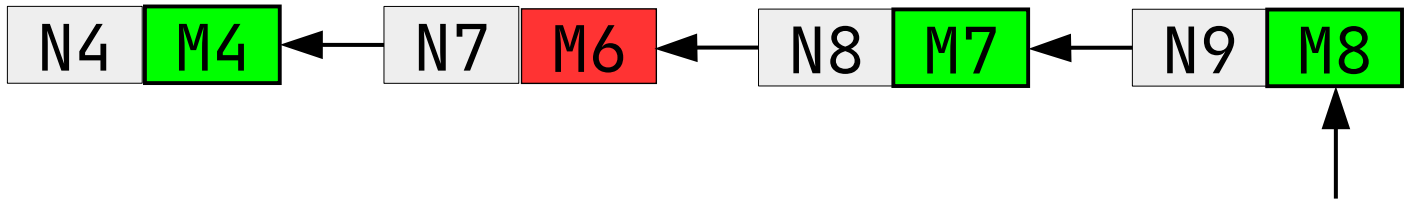
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

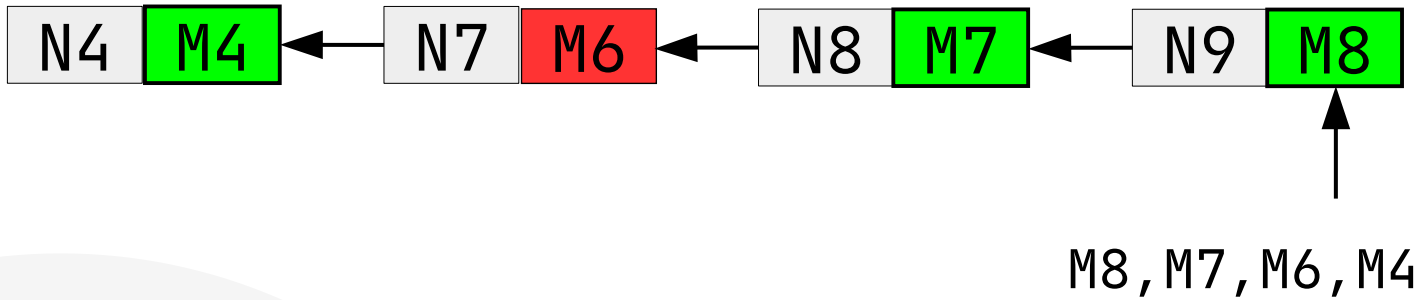
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

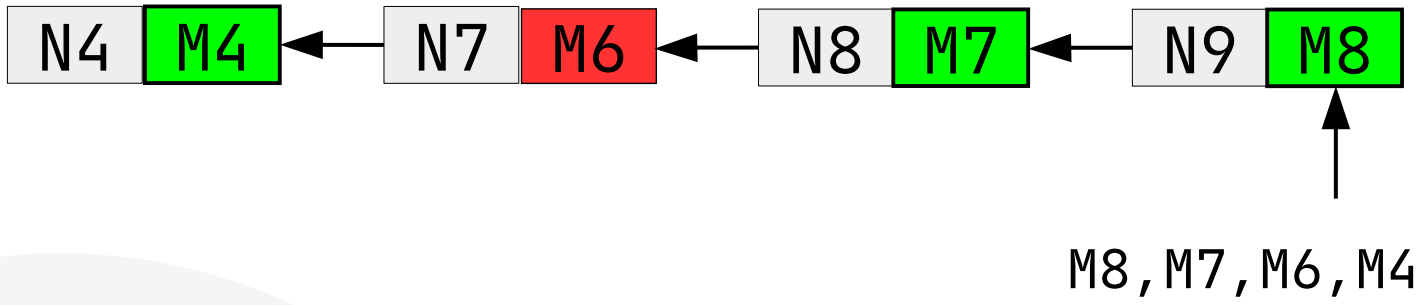
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

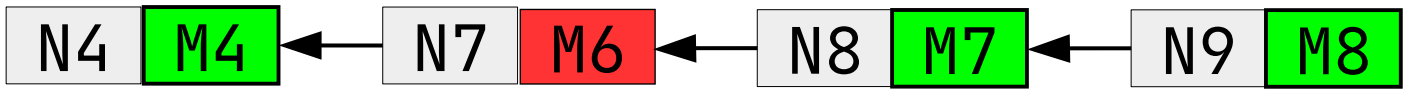
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

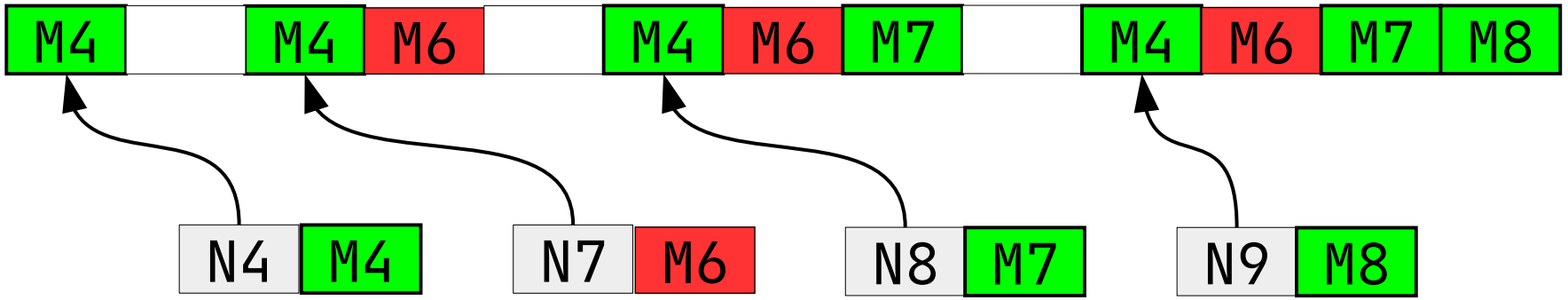
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

Распаковка словаря

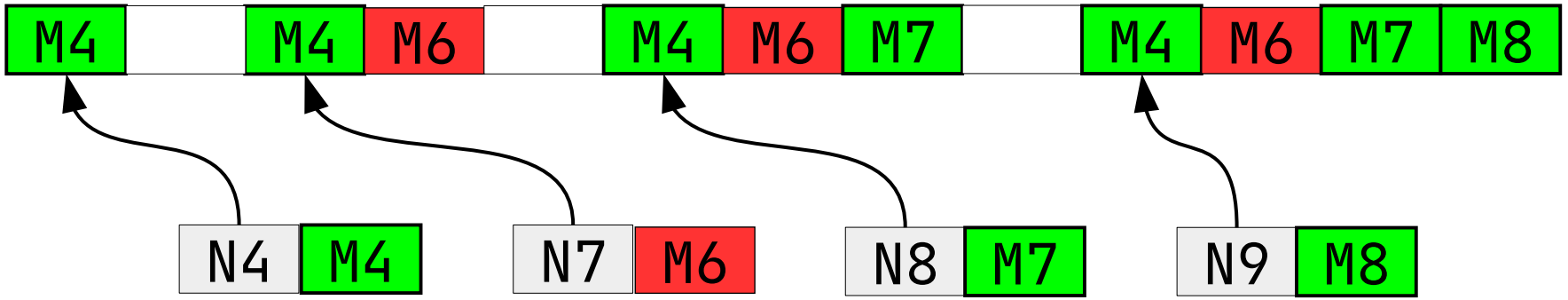


Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

Распаковка словаря

Так это же почти JFR!



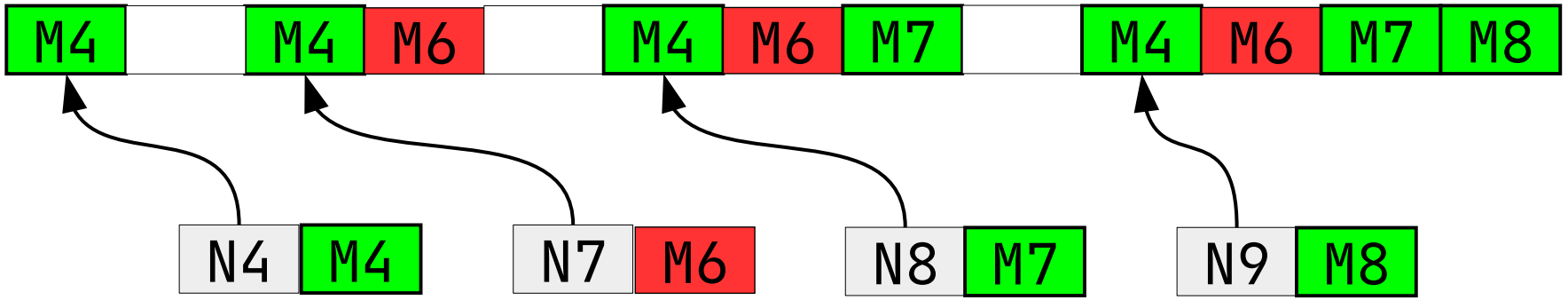
Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M4,M6,M7"
9:  "M4,M6,M7,M8"
```

Распаковка словаря

Так это же почти JFR!

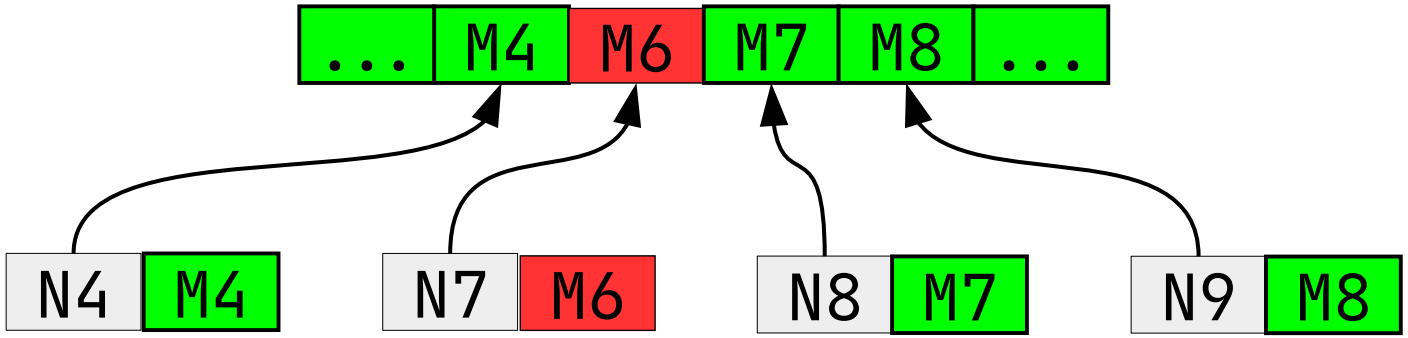
...4 GB...



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M4,M6,M7"
9:  "M4,M6,M7,M8"
```

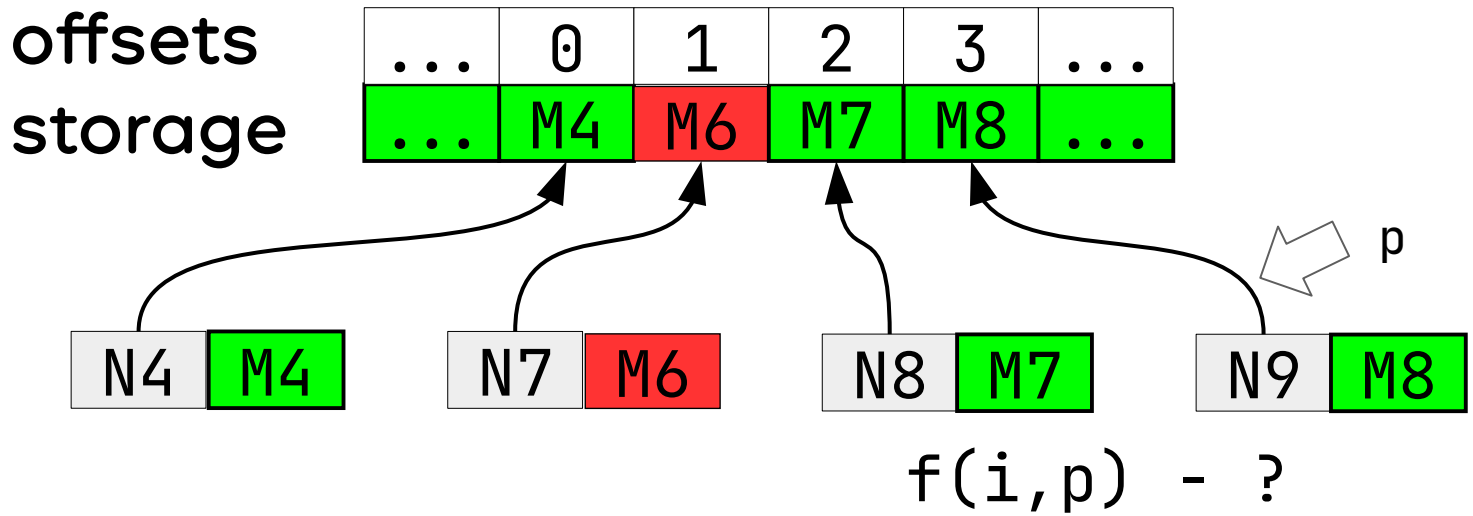
Распаковка словаря



Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

Распаковка словаря

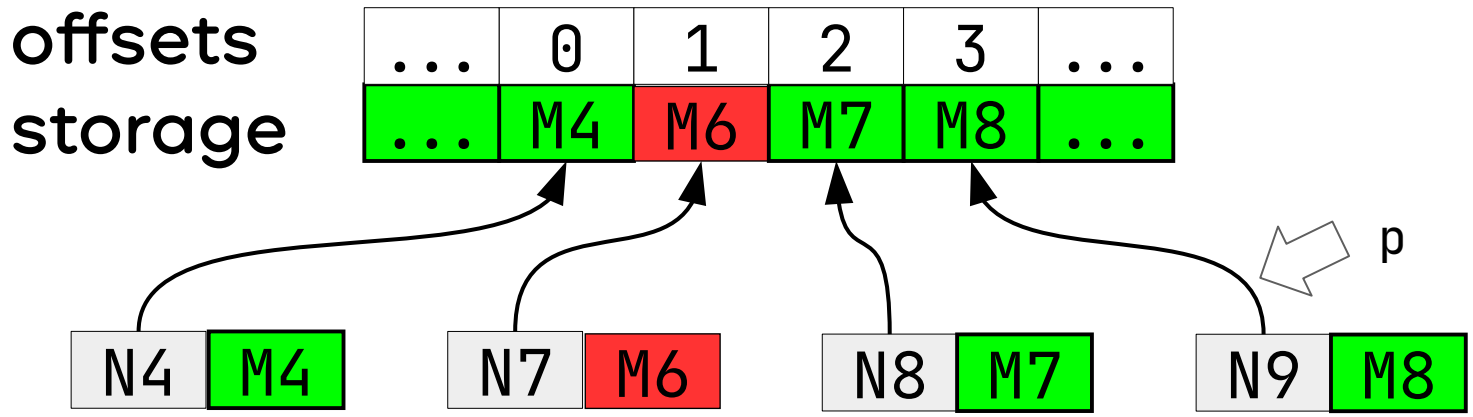


Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
     ↑ i
```

Распаковка словаря

```
f(i,p) = storage[p - offsets[p] + i]
```




```
f(0,p) = M4
f(1,p) = M6
f(2,p) = M7
f(3,p) = M8
```

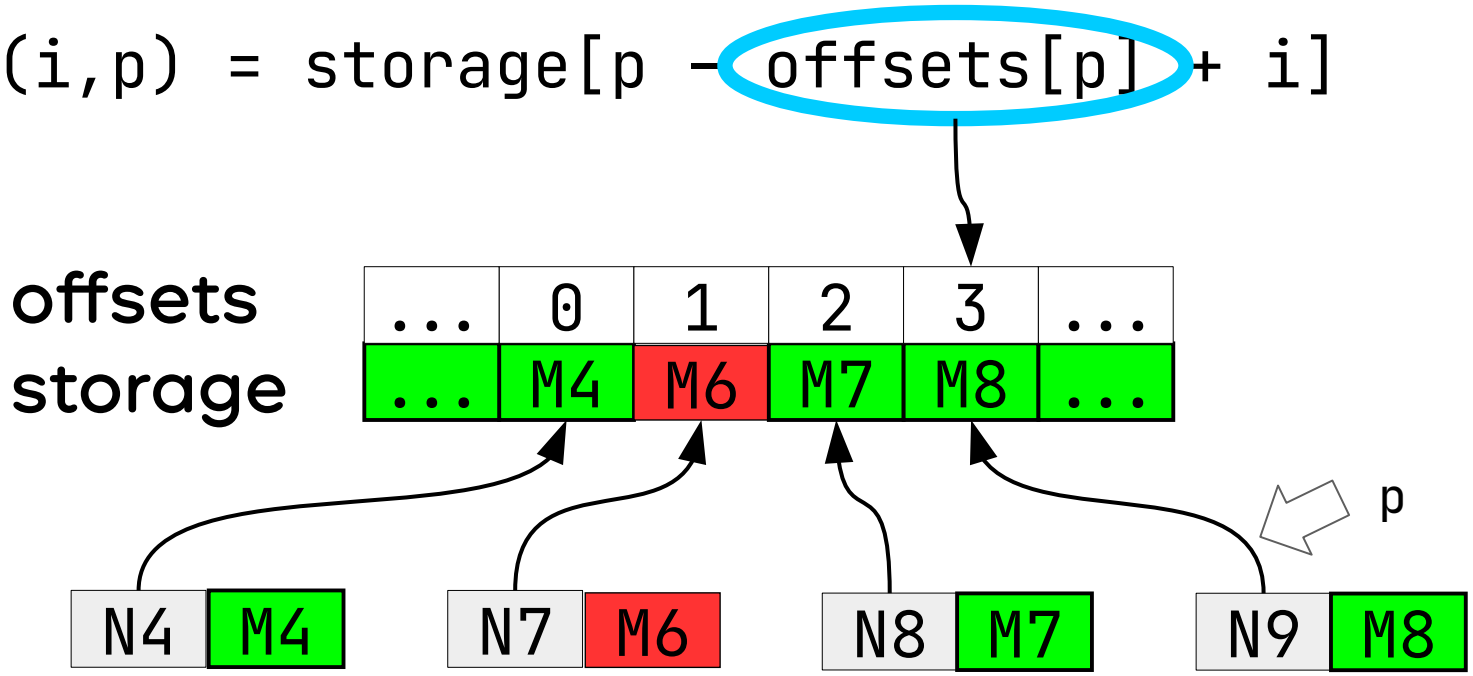
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

i

Распаковка словаря

```
f(i,p) = storage[p - offsets[p] + i]
```




```
f(0,p) = M4
f(1,p) = M6
f(2,p) = M7
f(3,p) = M8
```

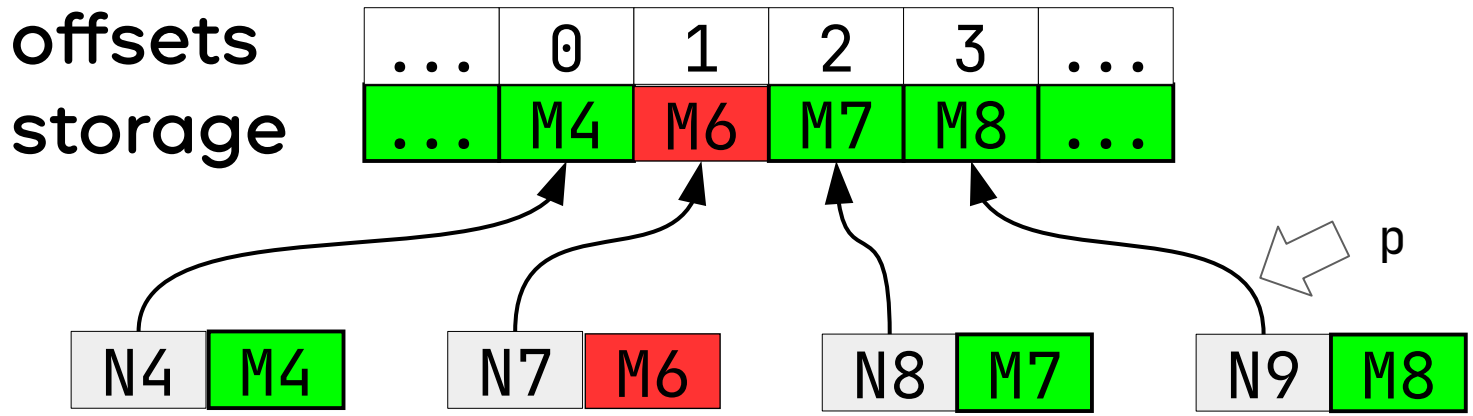
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

i

Распаковка словаря

```
f(i,p) = storage[p - 3 + i]
```




```
f(0,p) = storage[p - 3 + 0] = M4
f(1,p) = storage[p - 3 + 1] = M6
f(2,p) = storage[p - 3 + 2] = M7
f(3,p) = storage[p - 3 + 3] = M8
```

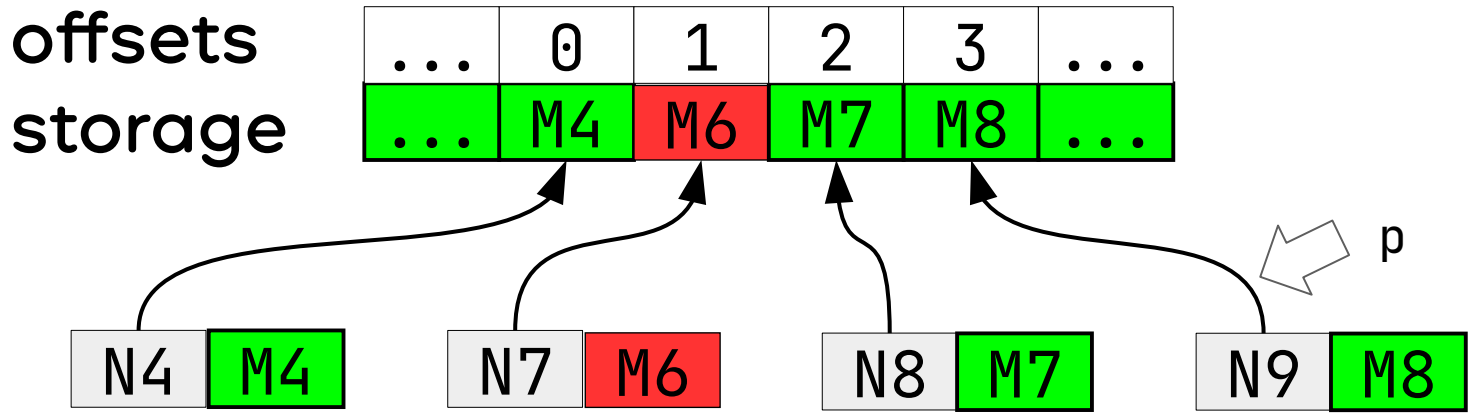
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

i

Распаковка словаря

80 МБ CPU
24 МБ Alloc




```
f(0,p) = storage[p - 3 + 0] = M4
f(1,p) = storage[p - 3 + 1] = M6
f(2,p) = storage[p - 3 + 2] = M7
f(3,p) = storage[p - 3 + 3] = M8
```

Словарь

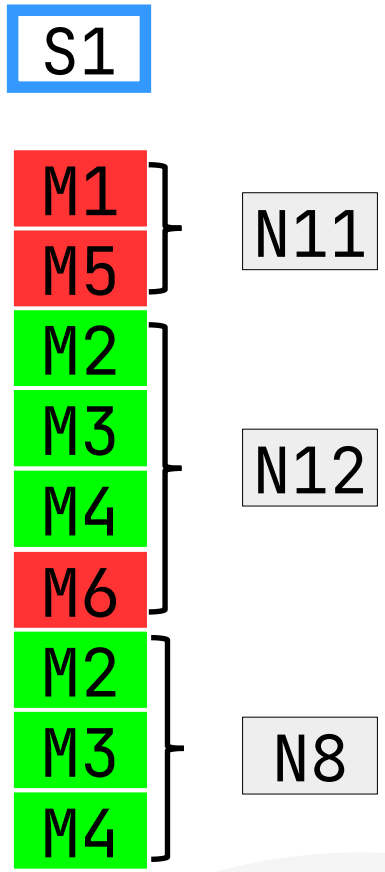
```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M4,M6,M7"
9: "M4,M6,M7,M8"
```

i

Распаковка

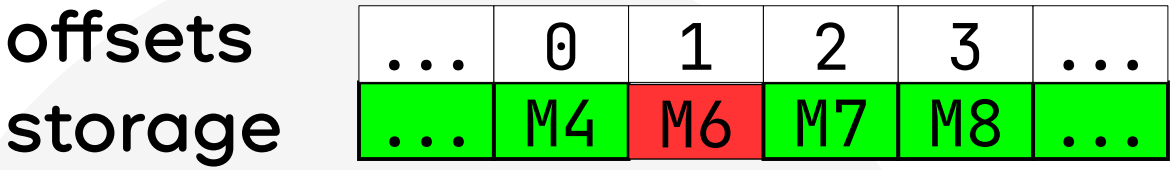
Сэмплов

Распаковка сэмплов



Словарь

```
0:  ""
1:  "M1"
2:  "M2"
3:  "M3"
4:  "M4"
5:  "M5"
6:  "M2,M3"
7:  "M4,M6"
8:  "M2,M3,M4"
9:  "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов

N11

N12

N8

offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Словарь

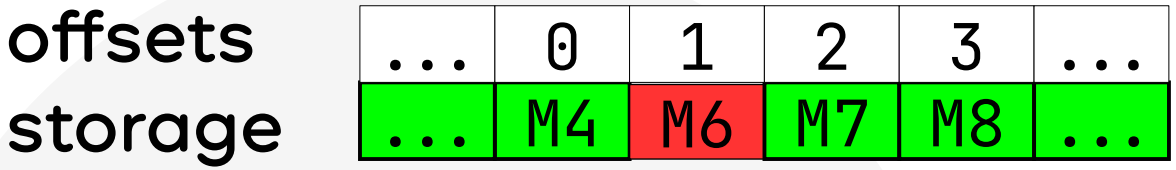
```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов

N11 N12 N8

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов

N11	N12	N8	N1	N6	N11	N13	...

Сэмплы

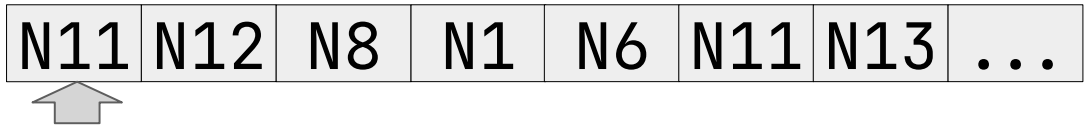
```
0:
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Распаковка сэмплов

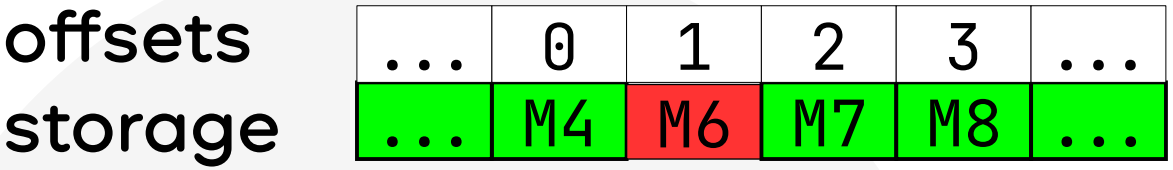


Сэмплы

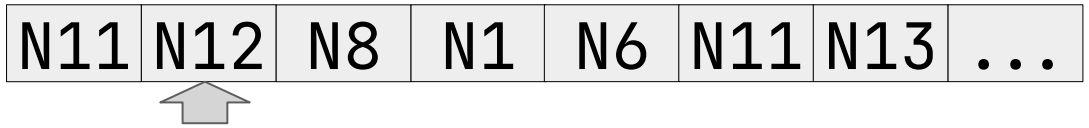
```
0: "N11"
```

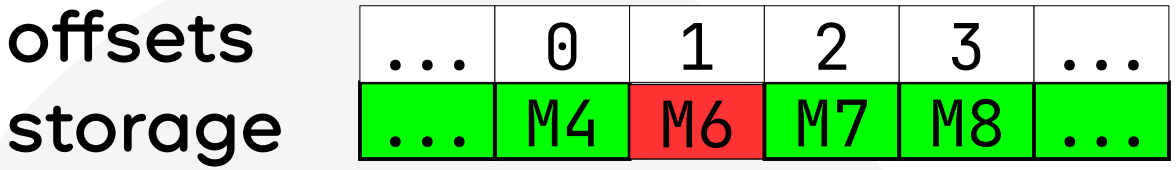
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов





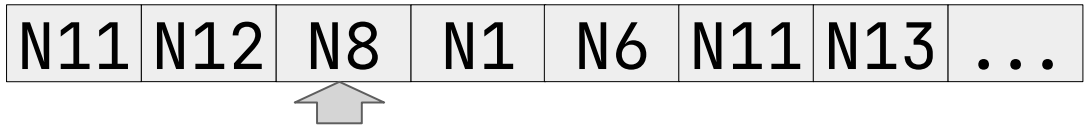
Сэмплы

```
0: "N11,N12"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов

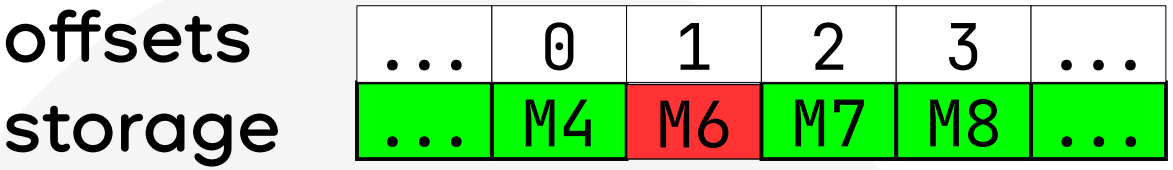


Сэмплы

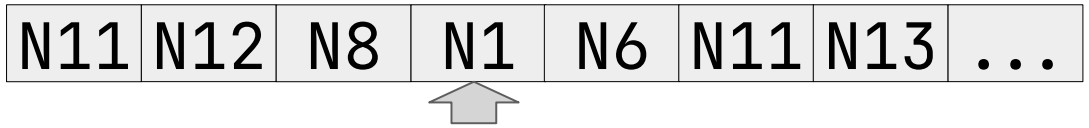
```
0: "N11,N12,N8"
```

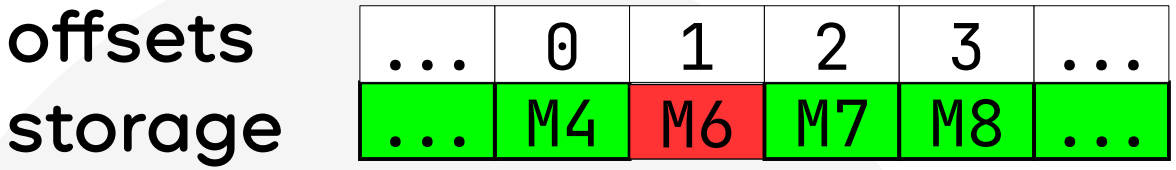
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов





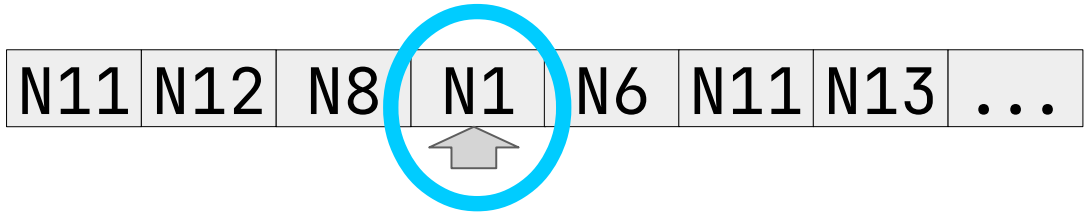
Сэмплы

```
0: "N11,N12,N8"
1: "N1"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов

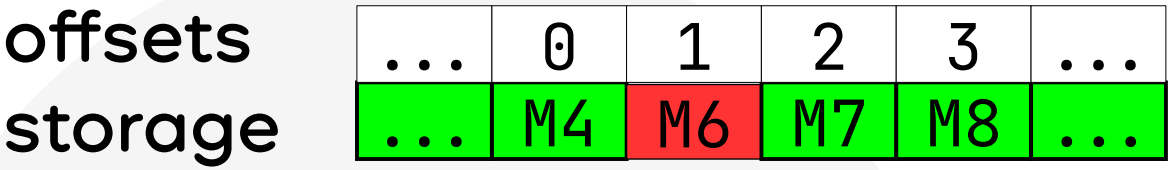


Сэмплы

```
0: "N11,N12,N8"
1: "N1"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов

N11	N12	N8	N1	N6	N11	N13	...

offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

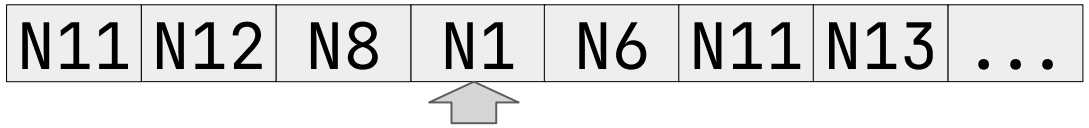
Сэмплы

```
0: "N11,N12,N8"
1: "N1"
```

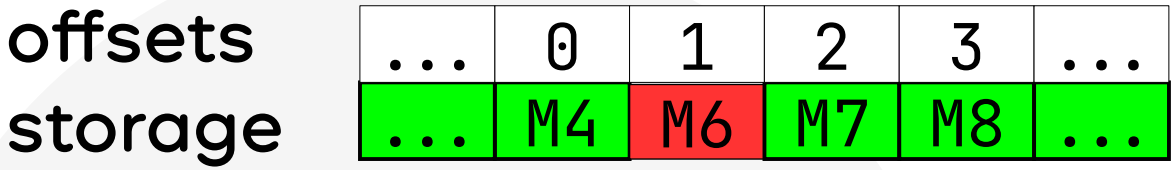
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов




```
java.lang.Thread.run
```



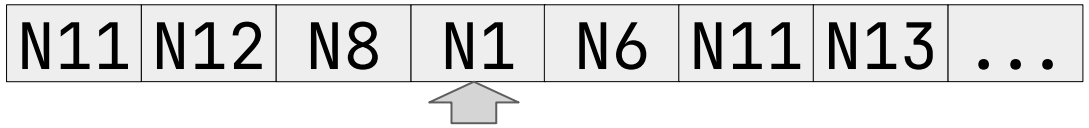
Сэмплы

```
0: "N11,N12,N8"
1: "N1"
```

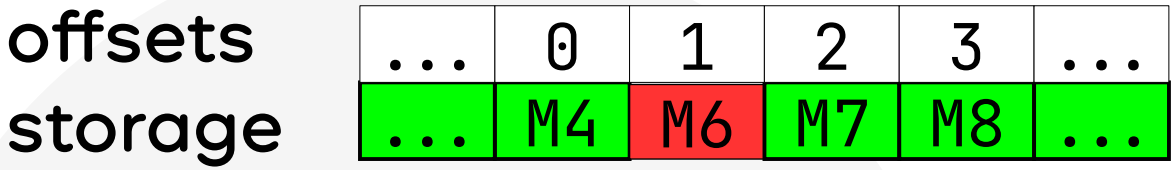
Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов




```
java.lang.Thread.run
```



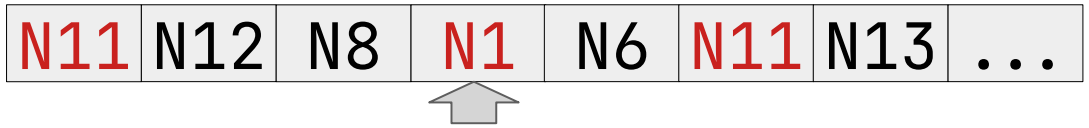
Сэмплы

```
0: "N11,N12,N8"
1: "N1"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов



java.lang.Thread.run

offsets

...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Сэмплы

```
0: "N11,N12,N8"
1: "N1"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов

N11	N12	N8	N1	N6	N11	N13	...

offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

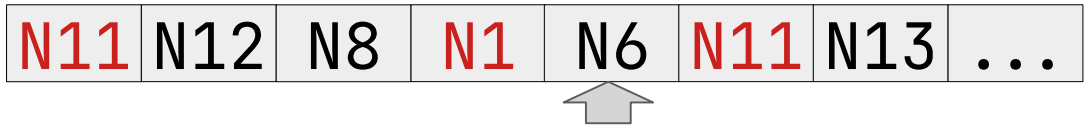
Сэмплы

```
0: "N11,N12,N8"
1: "N1"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов

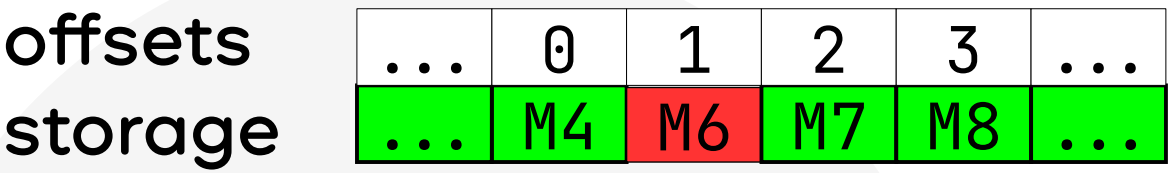


Сэмплы

```
0: "N11,N12,N8"
1: "N1,N6"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



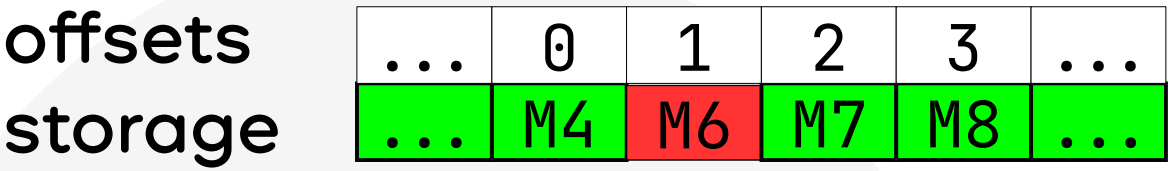
Распаковка сэмплов

Сэмплы

```
0: "N11,N12,N8"
1: "N1,N6"
2: "N11"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



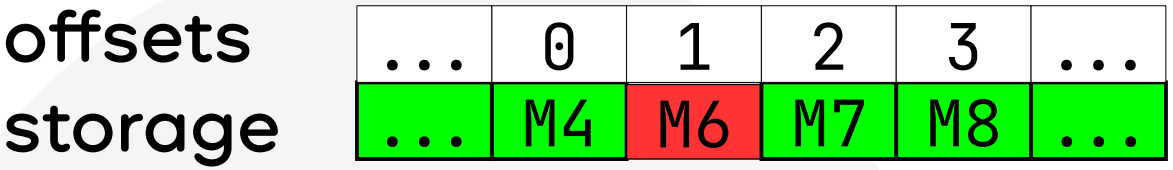
Распаковка сэмплов

Сэмплы

```
0: "N11,N12,N8"
1: "N1,N6"
2: "N11,N13"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов

N11	N12	N8	N1	N6	N11	N13	...

offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Сэмплы

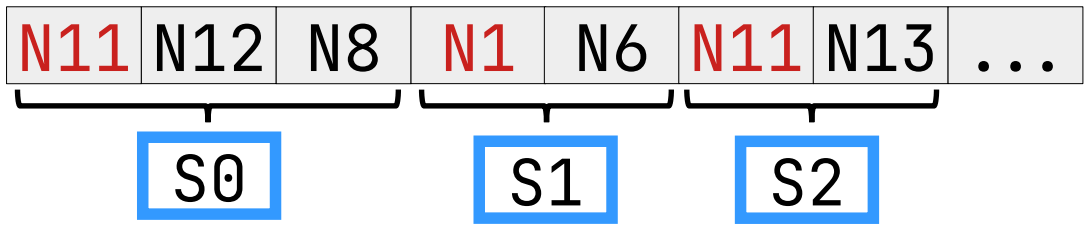
```
0: "N11,N12,N8"
1: "N1,N6"
2: "N11,N13"
```

Аллоцировать нельзя, Карл!

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

Распаковка сэмплов

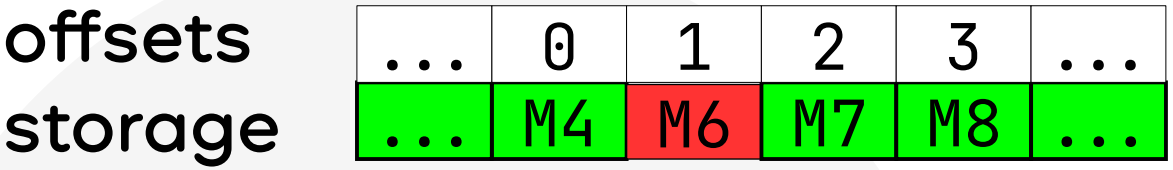


Сэмплы

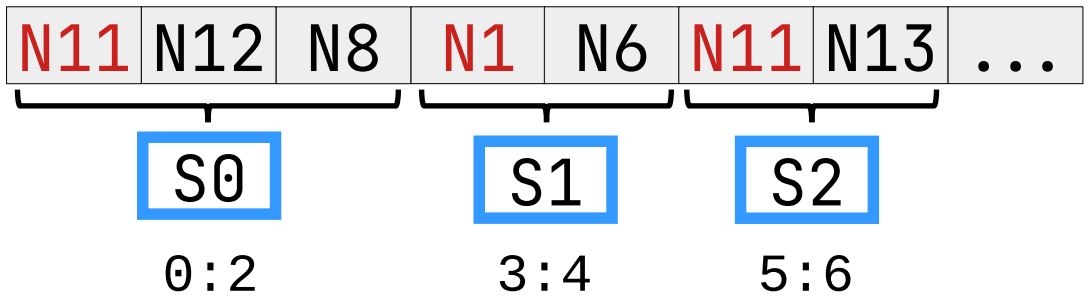
```
0: "N11,N12,N8"
1: "N1,N6"
2: "N11,N13"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов

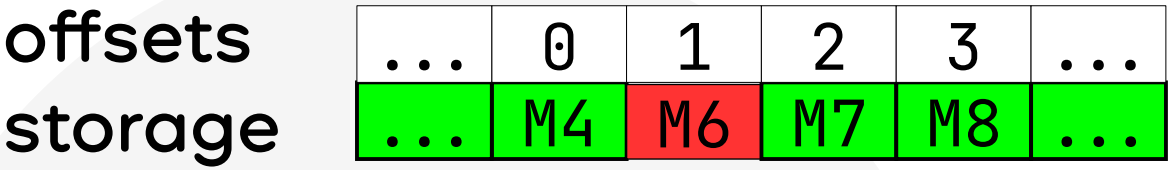


Сэмплы

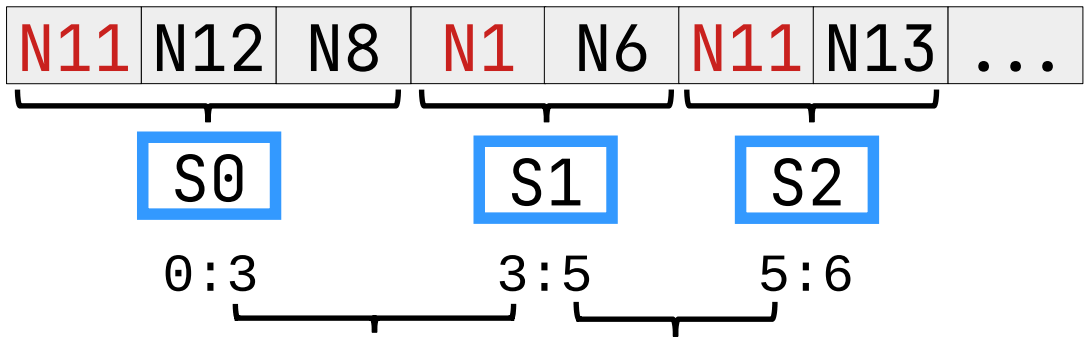
```
0: "N11,N12,N8"
1: "N1,N6"
2: "N11,N13"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов

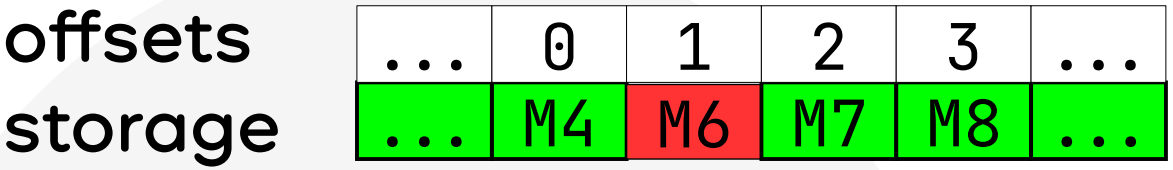


Сэмплы

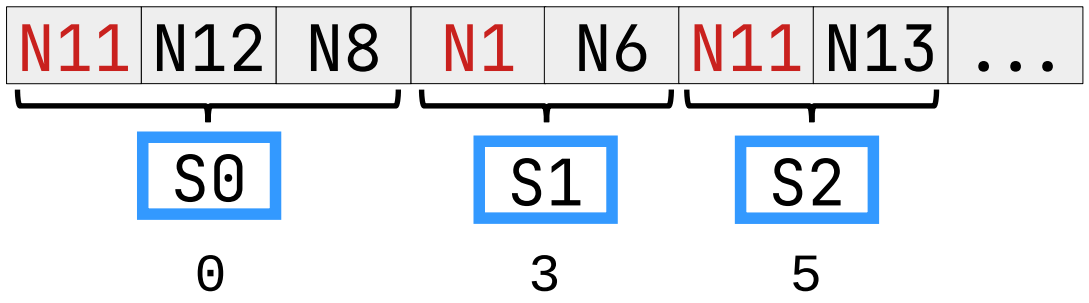
```
0: "N11,N12,N8"
1: "N1,N6"
2: "N11,N13"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов

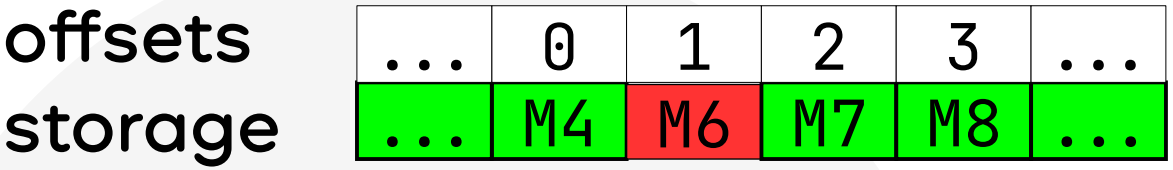


Сэмплы

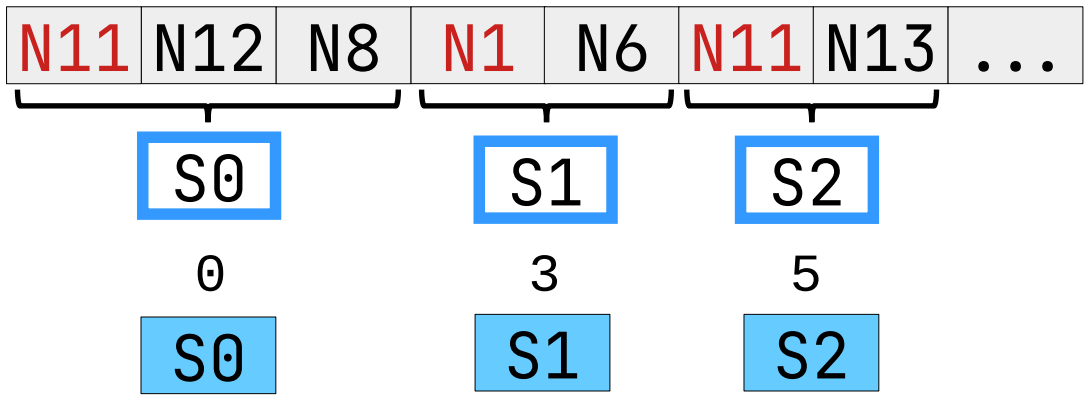
```
0: "N11,N12,N8"
1: "N1,N6"
2: "N11,N13"
```

Словарь

```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```



Распаковка сэмплов

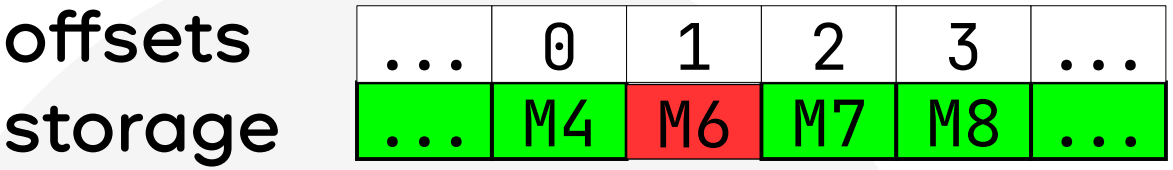


Сэмплы

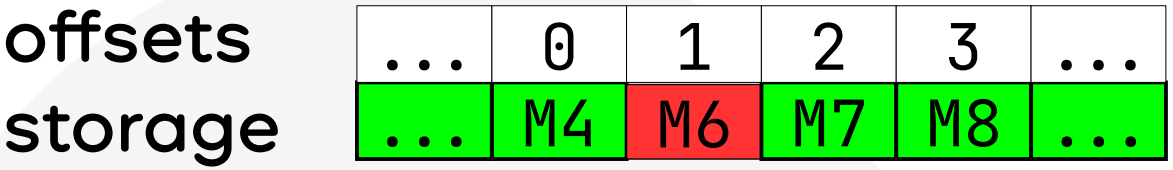
```
0: "N11,N12,N8"
1: "N1,N6"
2: "N11,N13"
```

Словарь

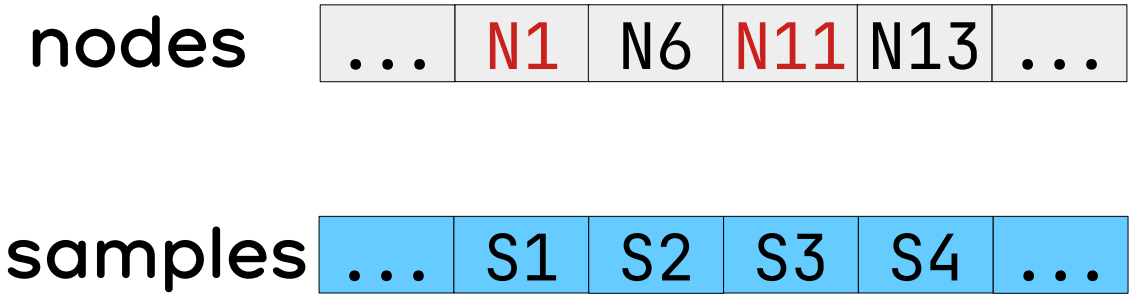
```
0: ""
1: "M1"
2: "M2"
3: "M3"
4: "M4"
5: "M5"
6: "M2,M3"
7: "M4,M6"
8: "M2,M3,M4"
9: "M7"
10:"M8"
11:"M1,M5"
12:"M2,M3,M4,M6"
13:"M2,M3,M4,M9"
```

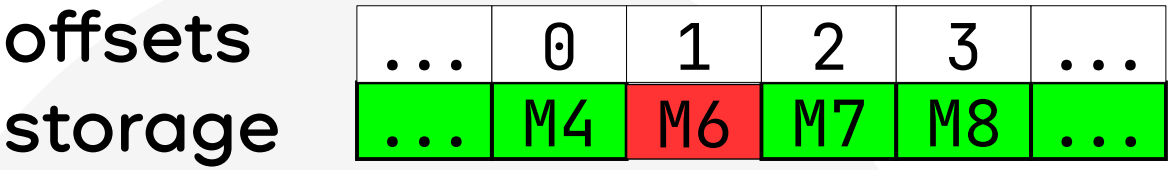


Распаковка сэмплов

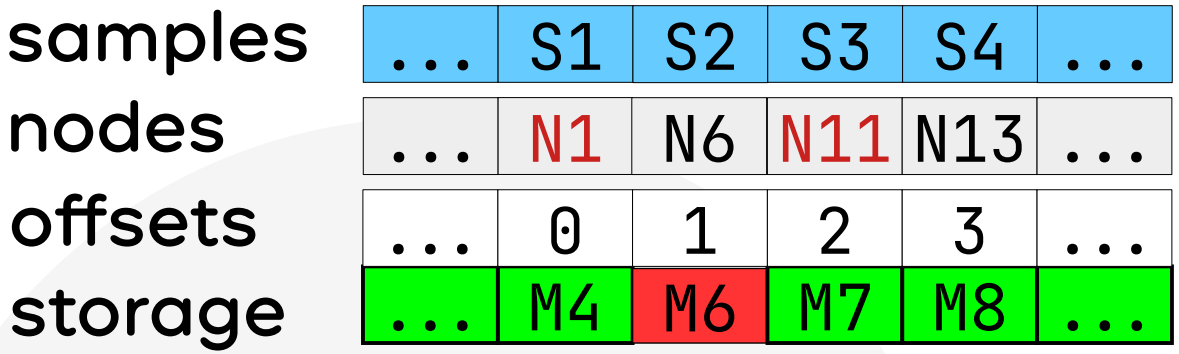


Распаковка сэмплов





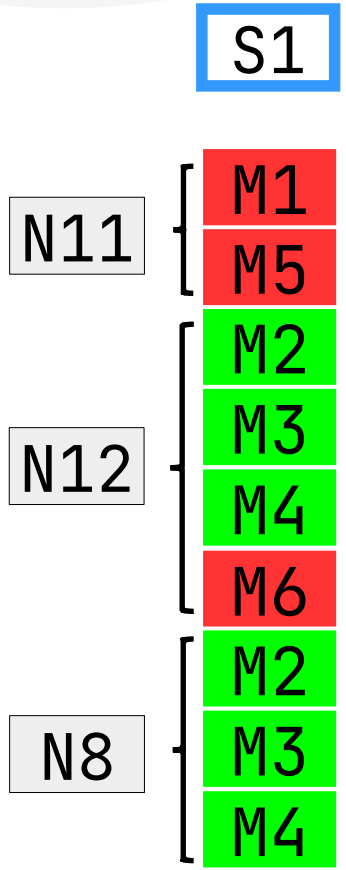
Распаковка сэмплов



Распаковка

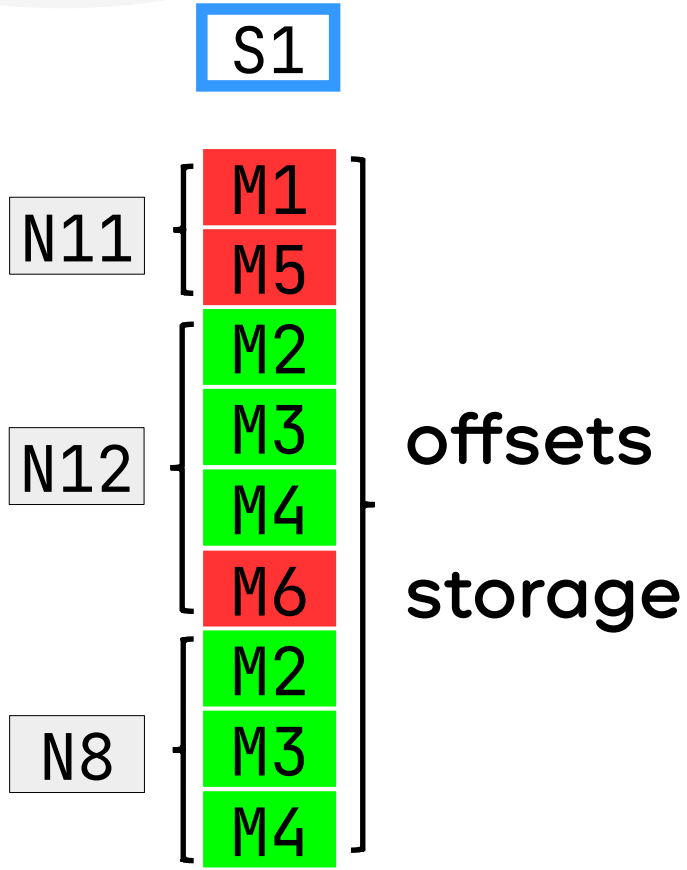
Закрепляем

Распаковка: закрепляем



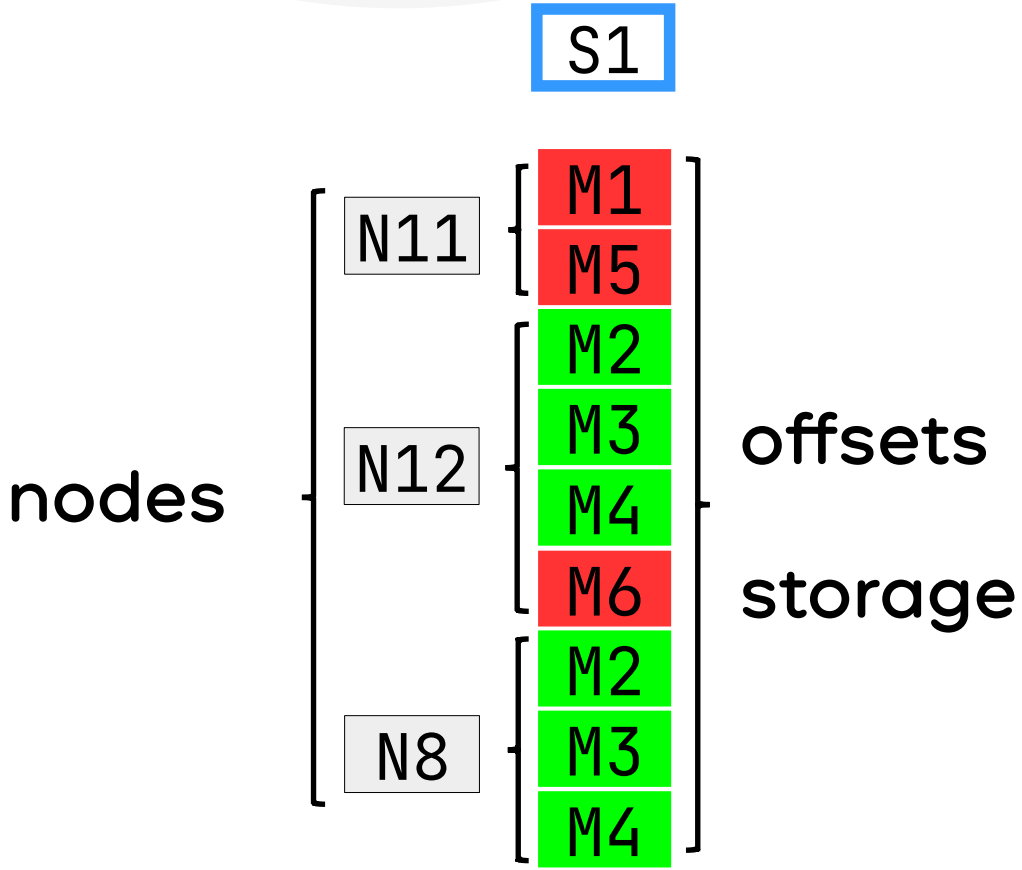
samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Распаковка: закрепляем



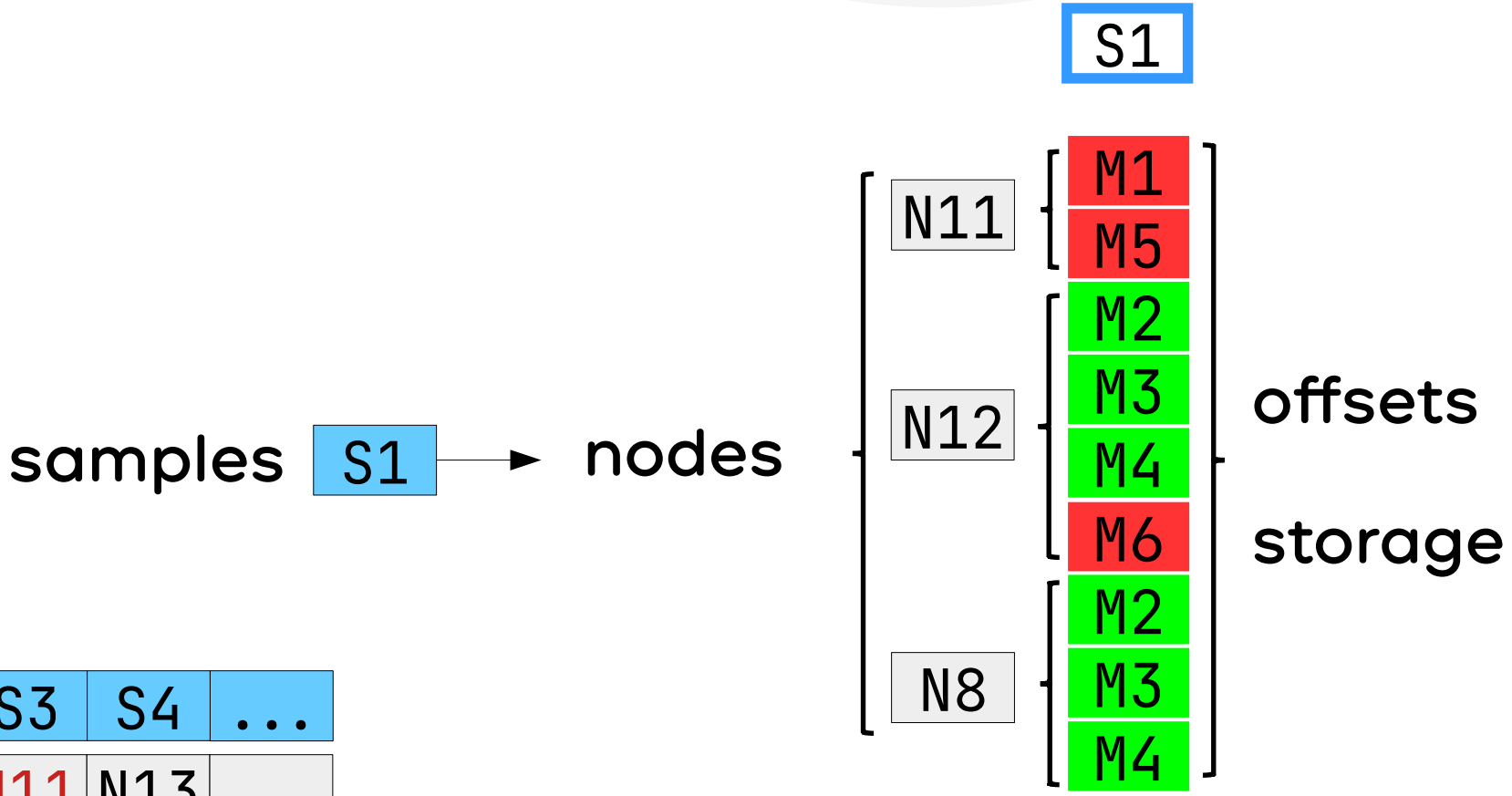
samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Распаковка: закрепляем



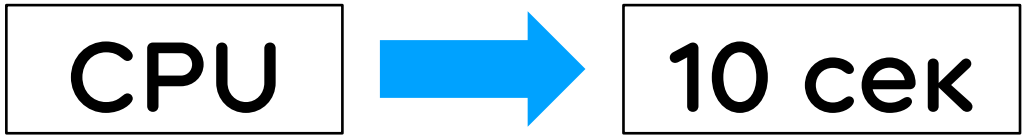
samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

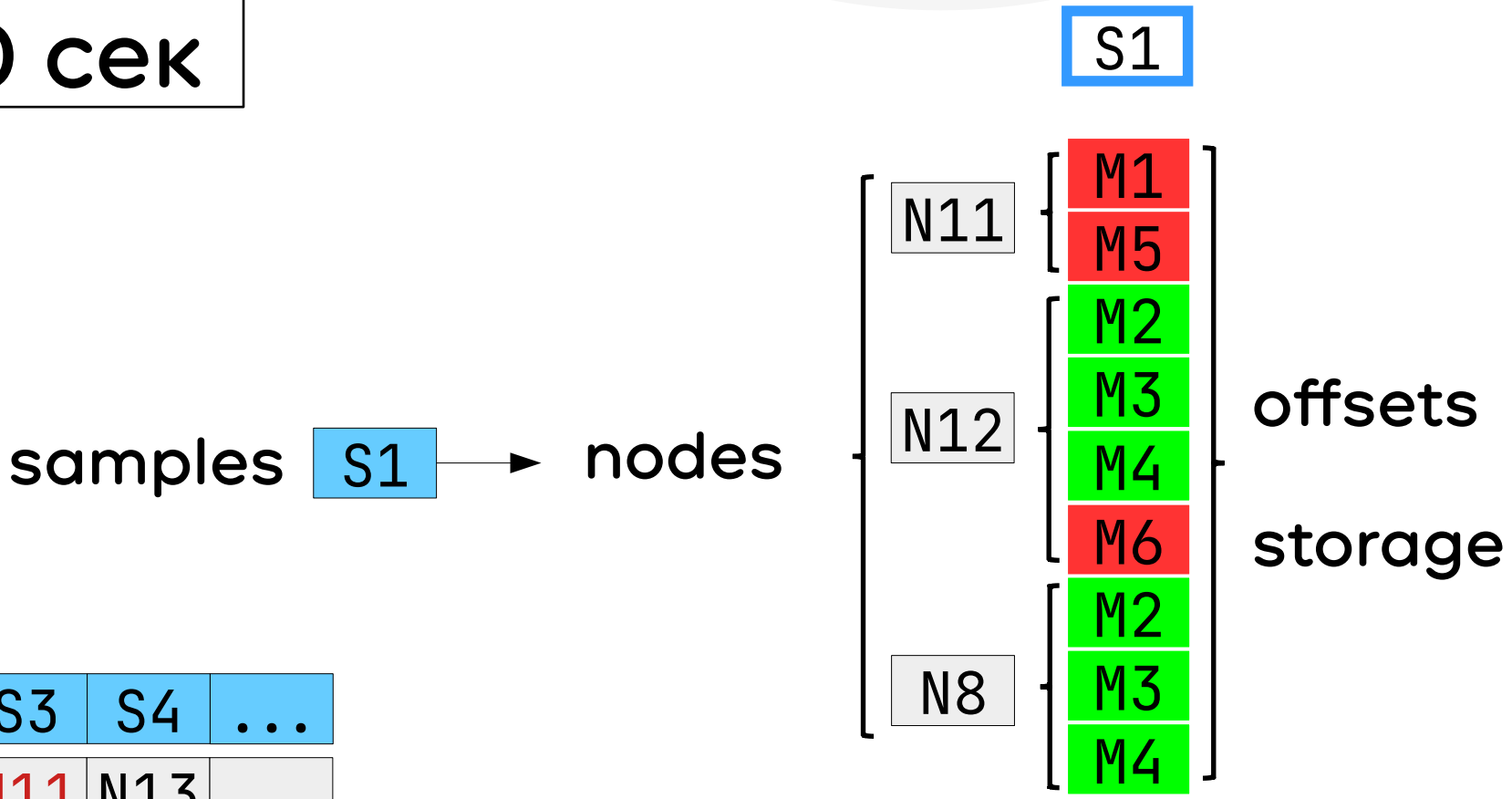
Распаковка: закрепляем



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

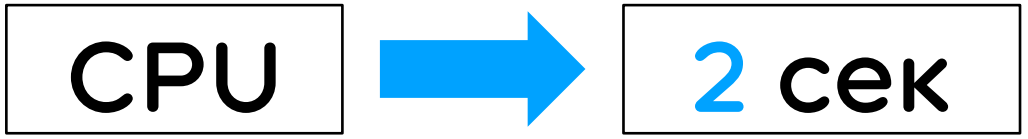
Распаковка: закрепляем

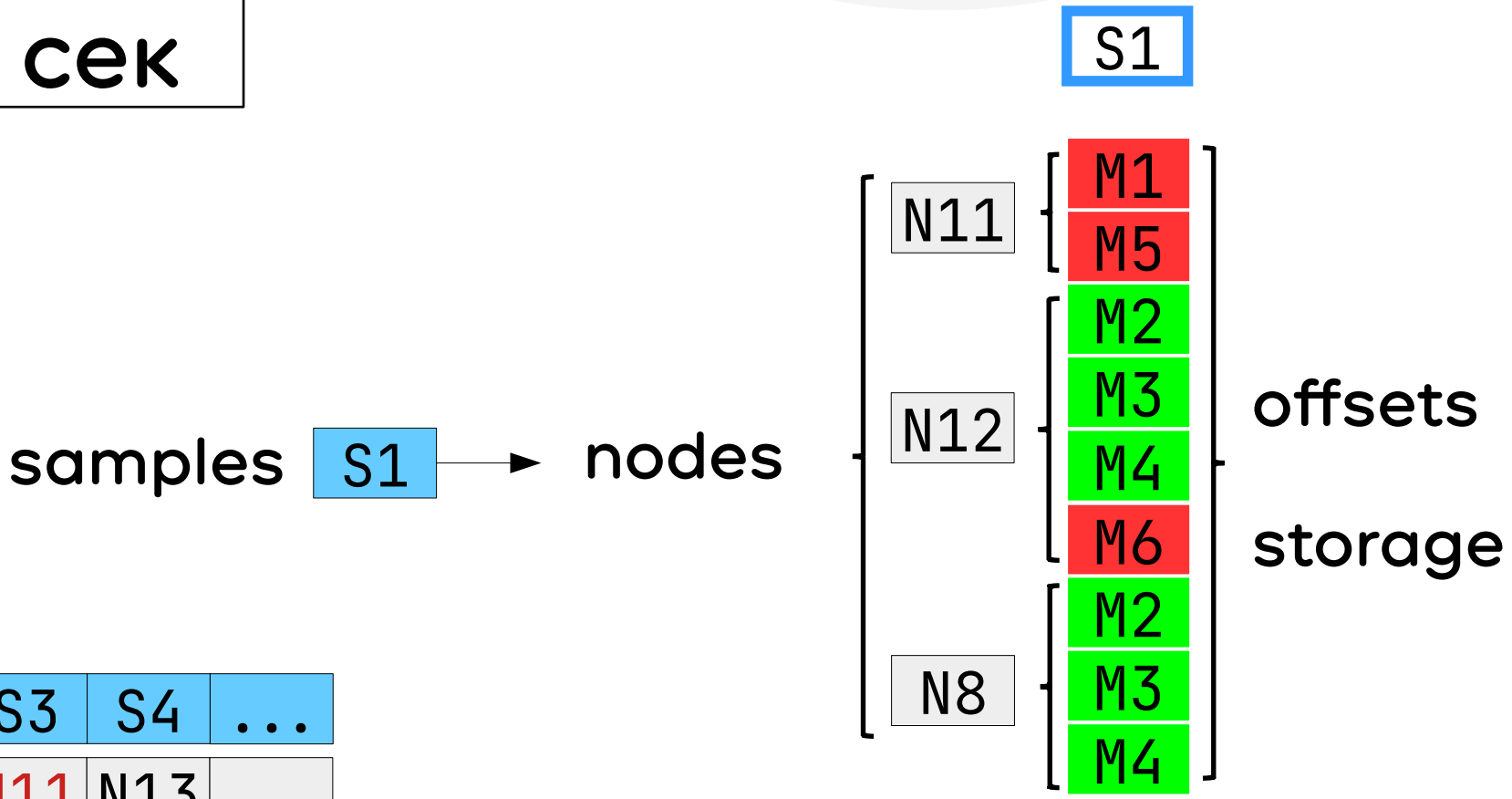




samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

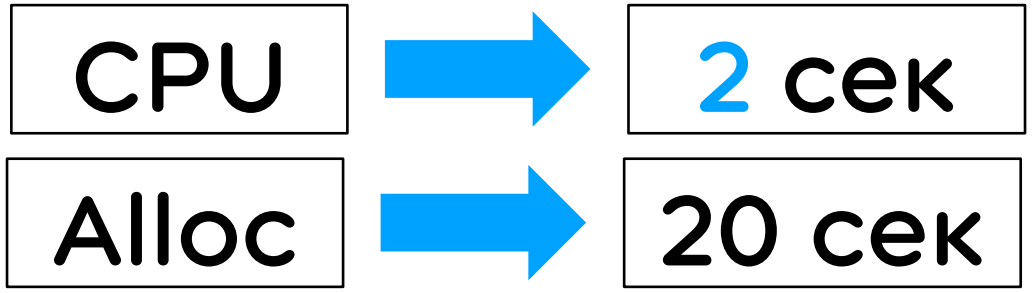
Распаковка: закрепляем

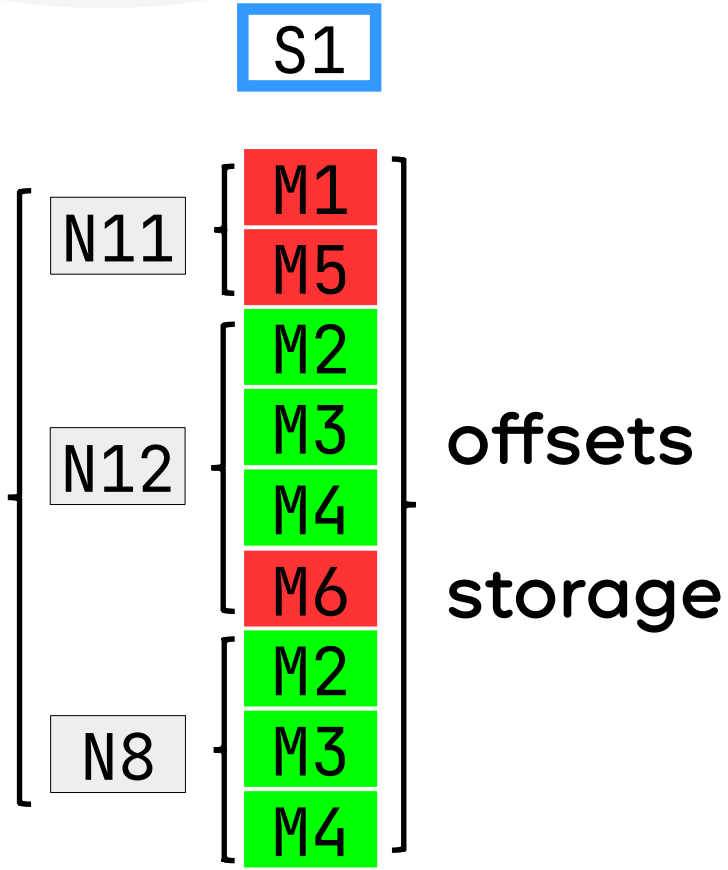




samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

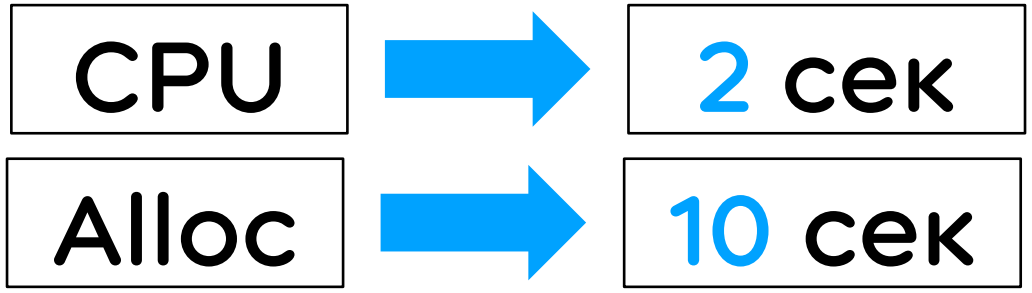
Распаковка: закрепляем

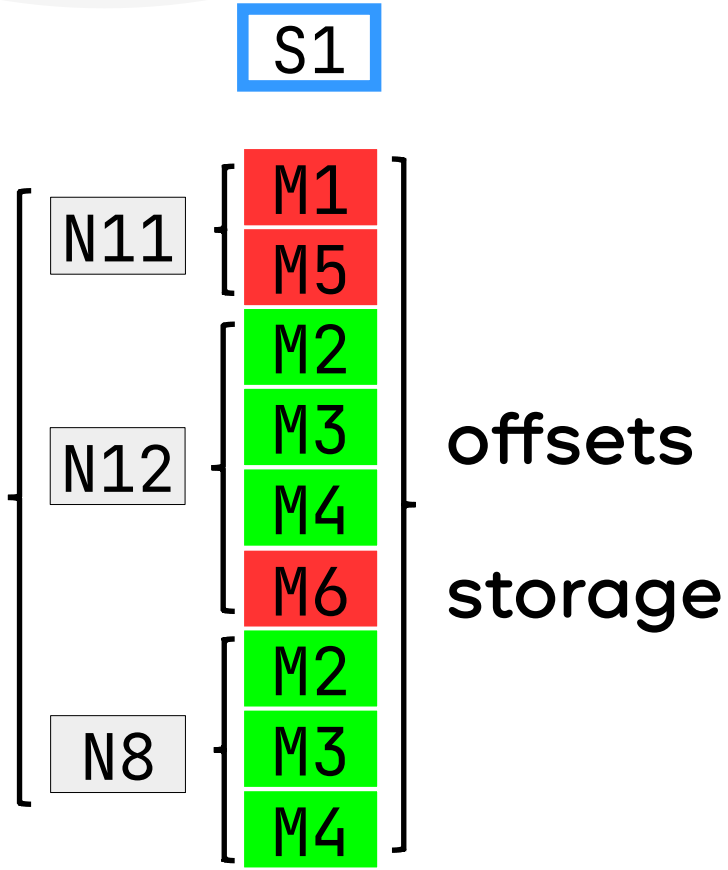




samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Распаковка: закрепляем



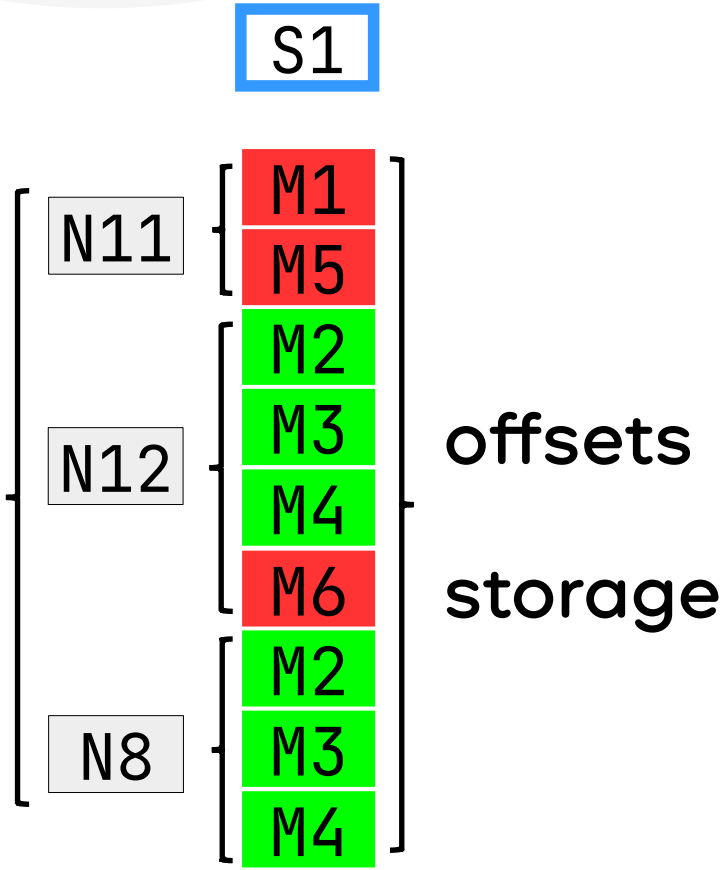


samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Распаковка: закрепляем

CPU	2 сек
Alloc	10 сек

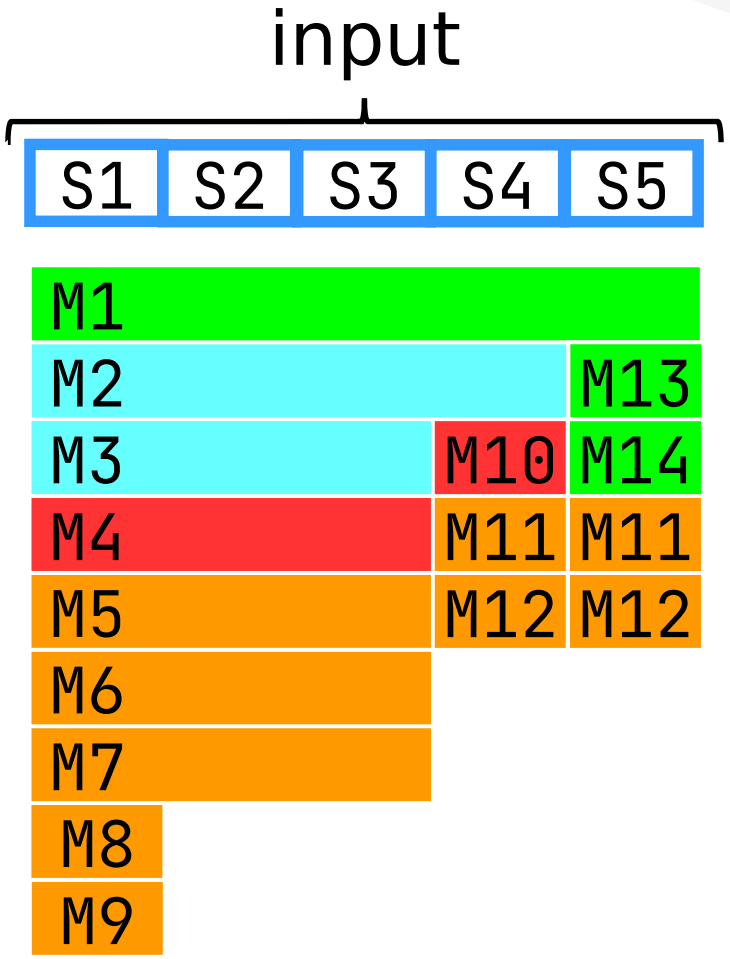
samples S1 → nodes



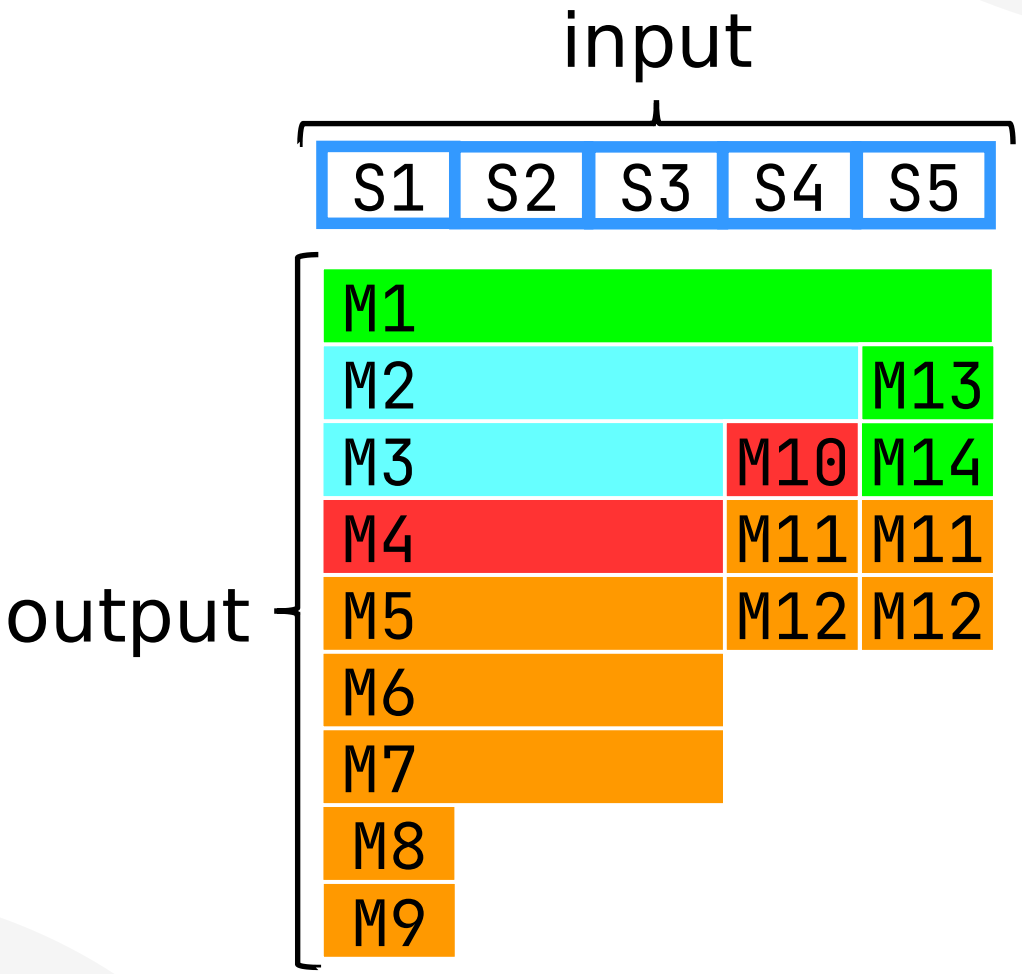
samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

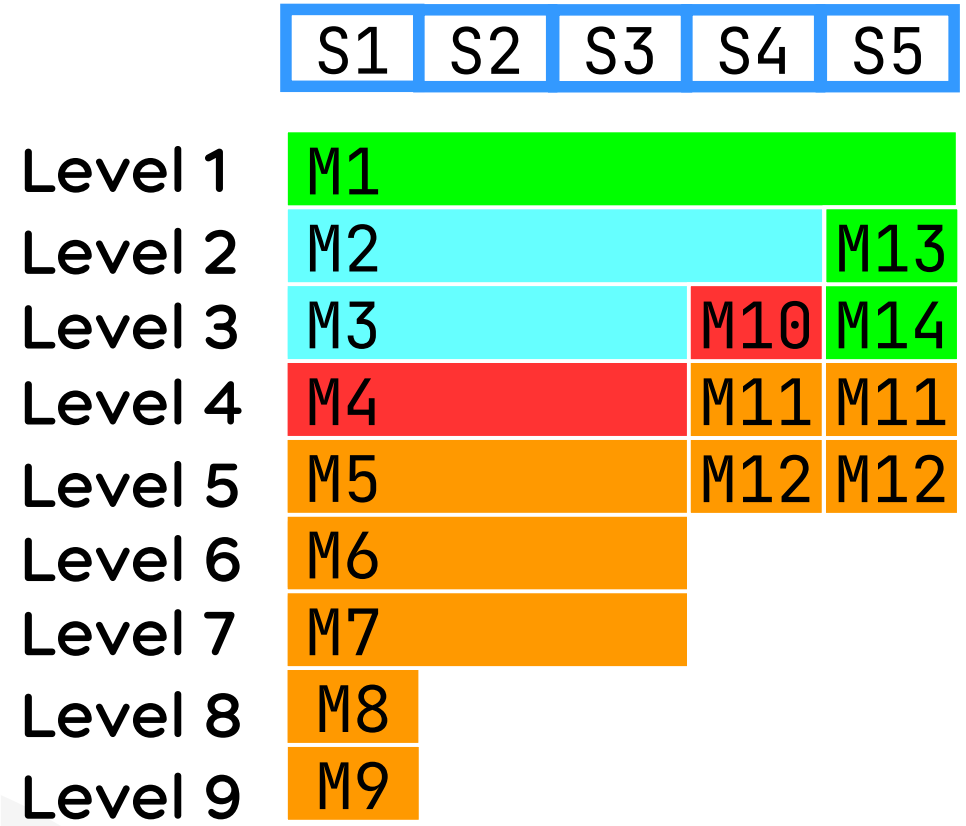
Что рисуем?


input
S1
S2
S3
S4
S5
M1
M2
M13
M3
M10
M14
M4
M11
M11
M5
M12
M12
M6
M7
M8
M9

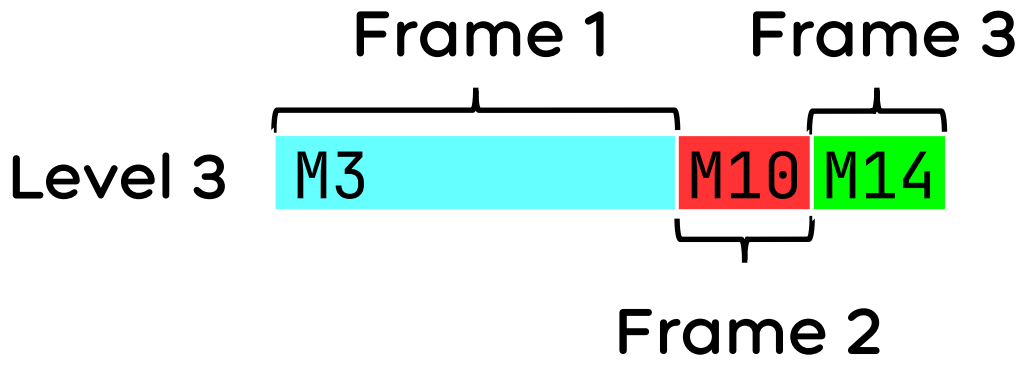
Что рисуем?


input
S1
S2
S3
S4
S5
output
M1
M2
M13
M3
M10
M14
M4
M11
M11
M5
M12
M12
M6
M7
M8
M9

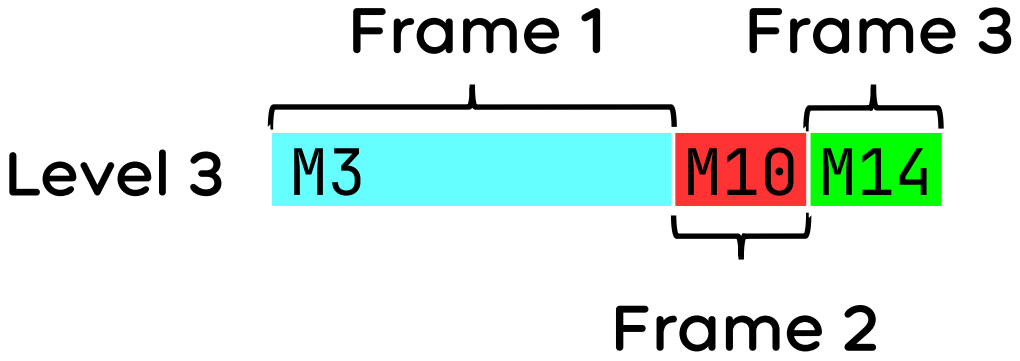
Что рисуем?


S1
S2
S3
S4
S5
Level 1
Level 2
Level 3
Level 4
Level 5
Level 6
Level 7
Level 8
Level 9
M1
M2
M13
M3
M10
M14
M4
M11
M11
M5
M12
M12
M6
M7
M8
M9

Что рисуем?



Что рисуем?

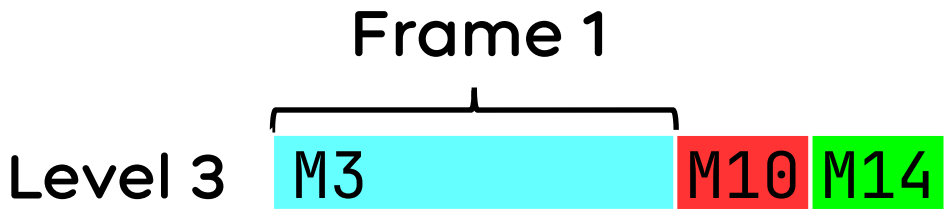


Что рисуем?

FlameGraph: `Level[]`

Level: `Frame[]`

Frame: `{x, width, text, color}`

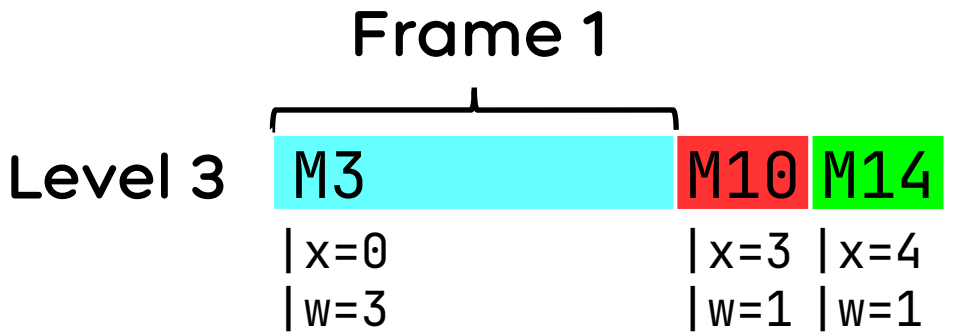


Что рисуем?

FlameGraph: `Level[]`

Level: `Frame[]`

Frame: `{x, width, text, color}`



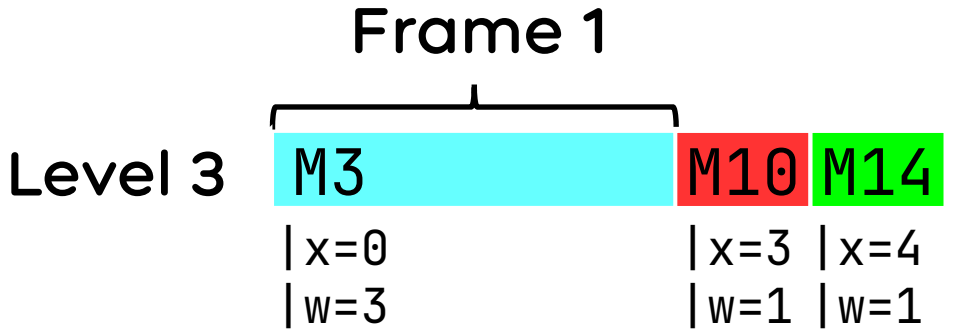
Что рисуем?

FlameGraph: `Level[]`

Level: `Frame[]`

Frame: `{x, width, text, color}`

} Результат работы



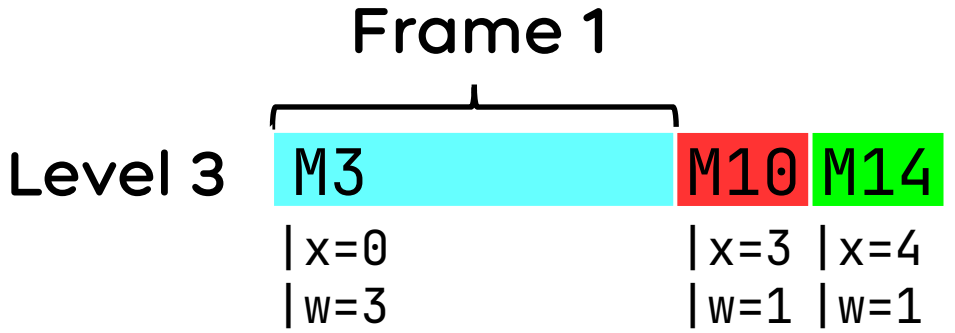
Что рисуем?

FlameGraph: `Level[]`

Level: `Frame[]`

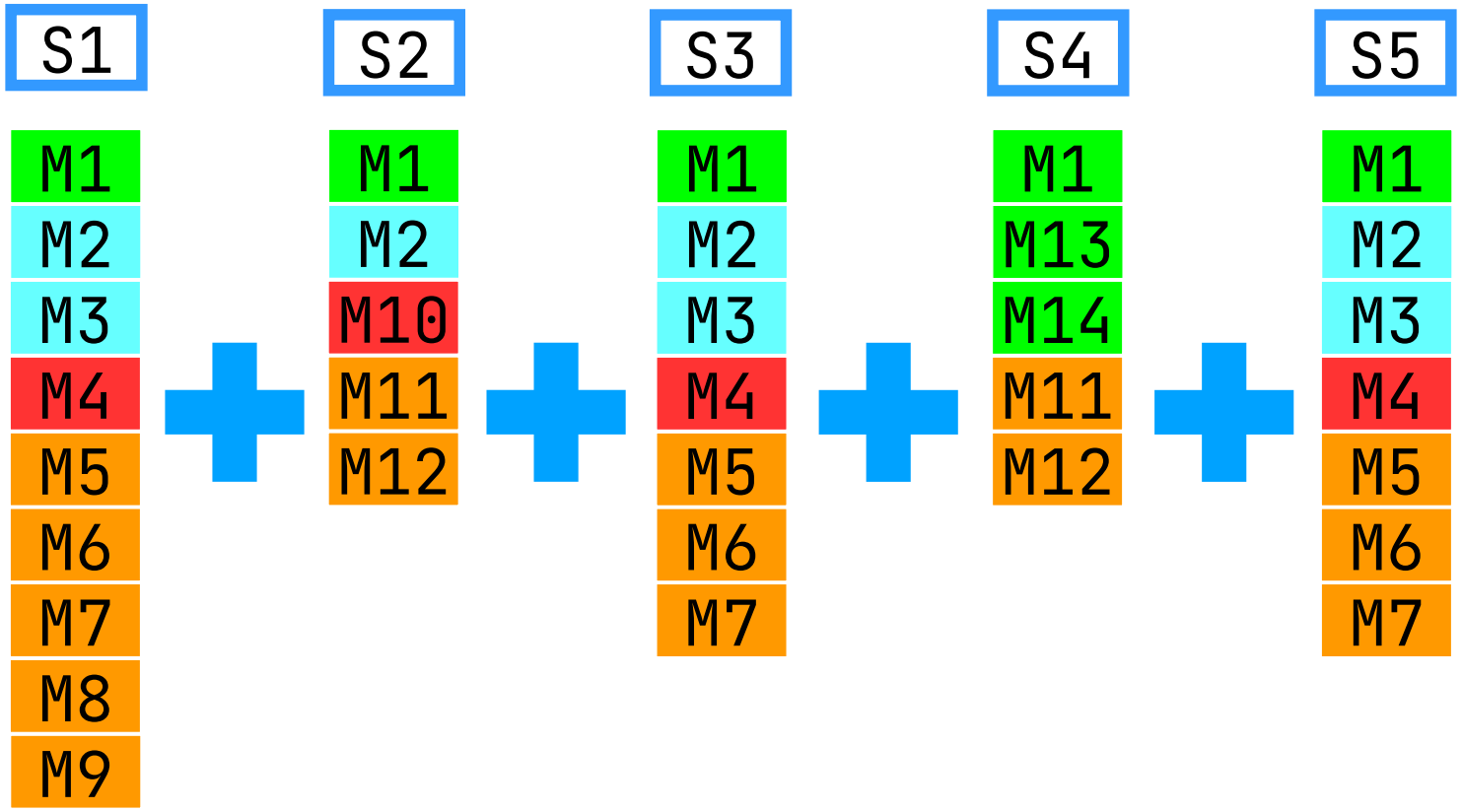
Frame: `{x, width, text, color}`

Результат работы

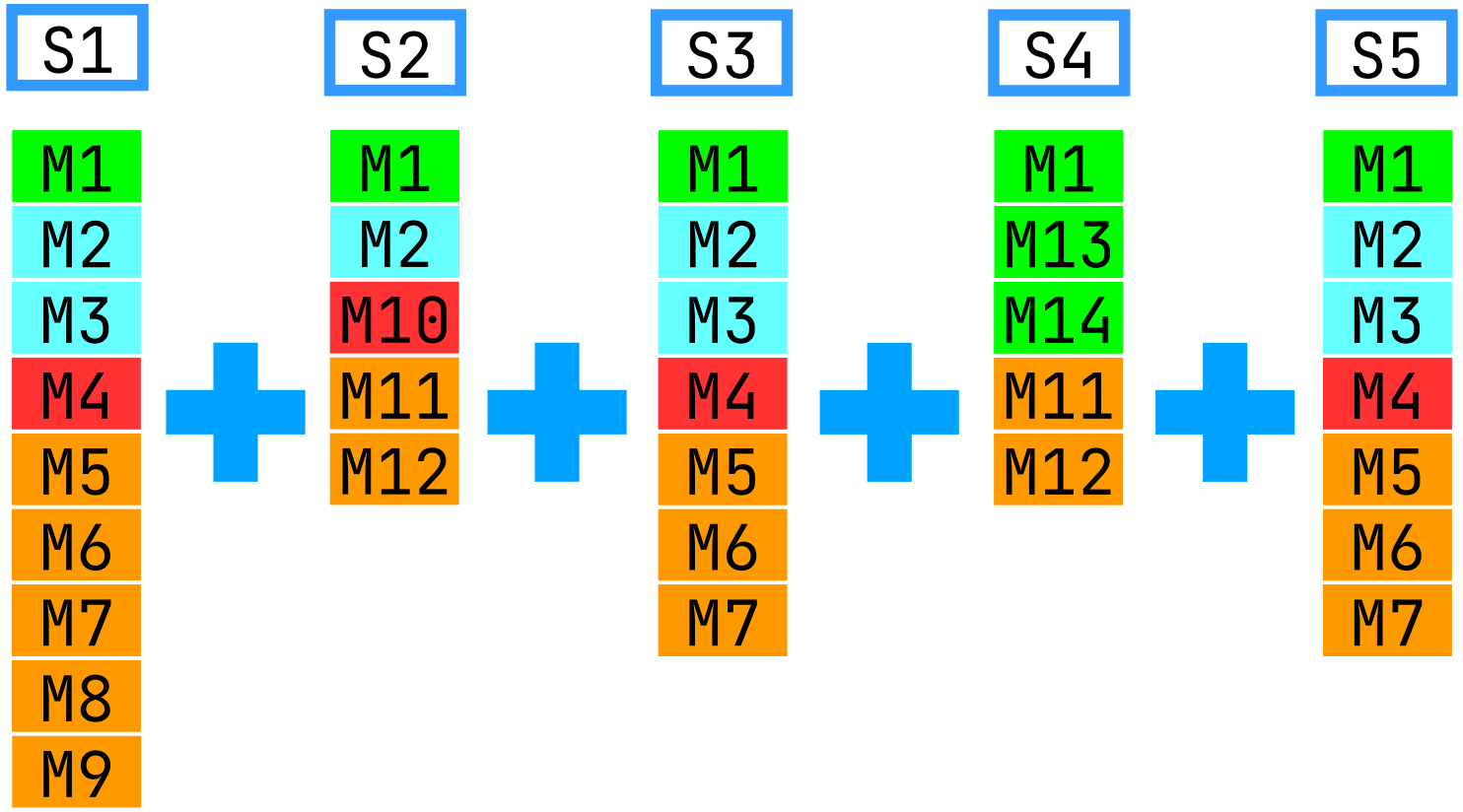


Как рисуем?

Как рисуем?


S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M1
M2
M10
M11
M12
S3
M1
M2
M3
M4
M5
M6
M7
S4
M1
M13
M14
M11
M12
S5
M1
M2
M3
M4
M5
M6
M7

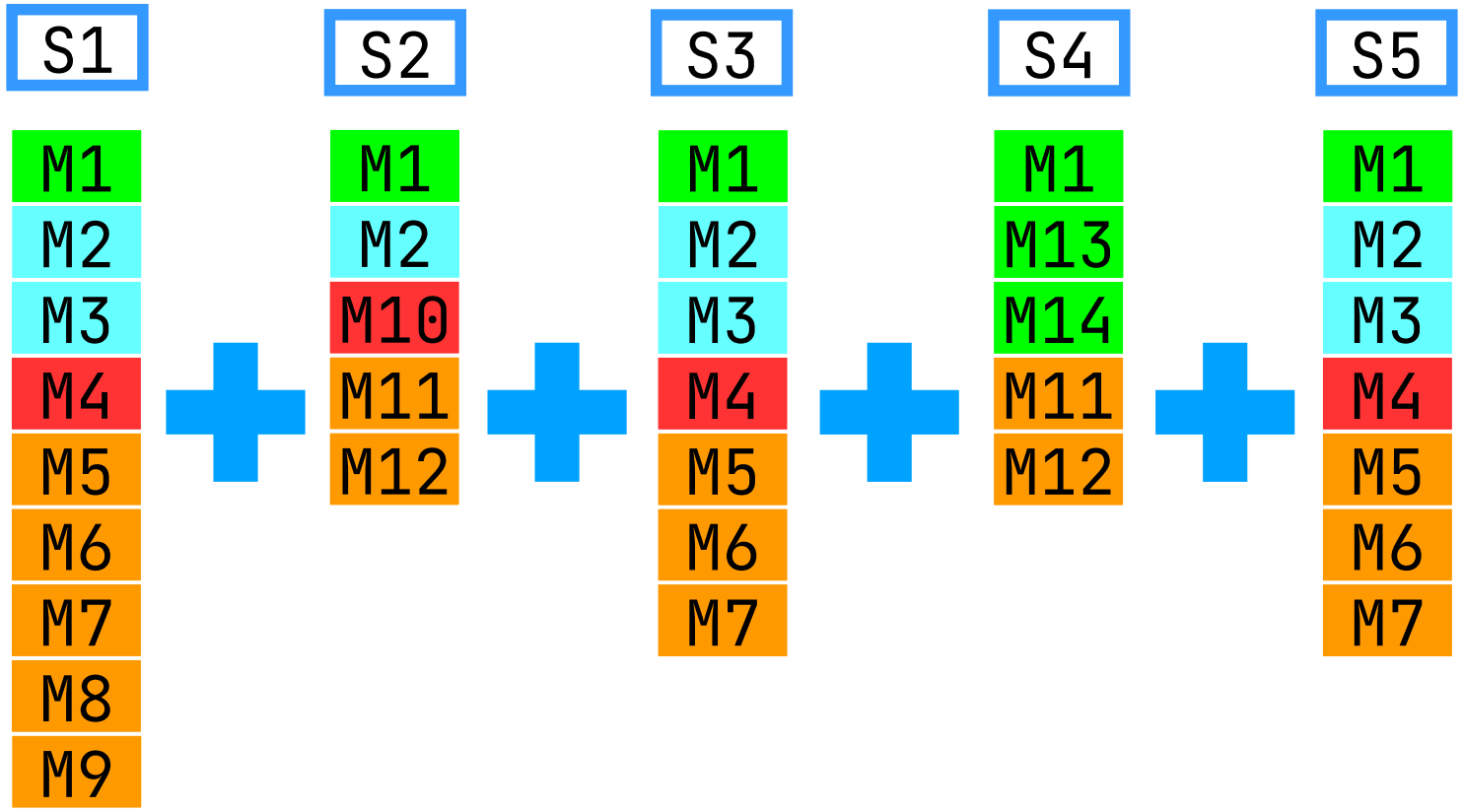
Как рисуем?


S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M1
M2
M10
M11
M12
S3
M1
M2
M3
M4
M5
M6
M7
S4
M1
M13
M14
M11
M12
S5
M1
M2
M3
M4
M5
M6
M7

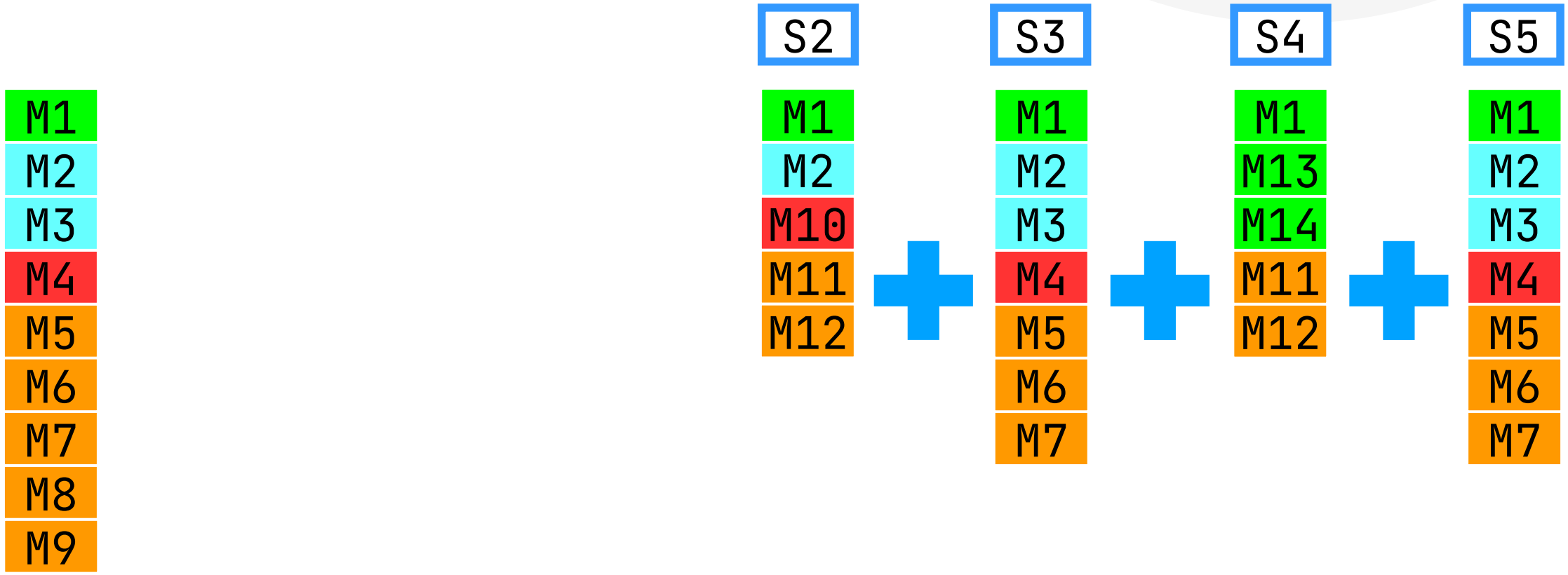
Как рисуем?

S1	S2	S3	S4	S5
M1	M1	M1	M1	M1
M2	M2	M2	M13	M2
M3	M10	M3	M14	M3
M4	M11	M4	M11	M4
M5	M12	M5	M12	M5
M6		M6		M6
M7		M7		M7
M8				
M9				

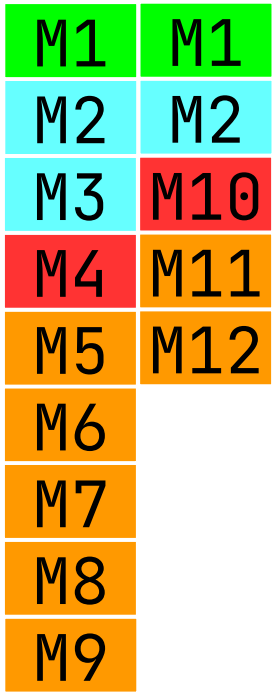
A. В глубину

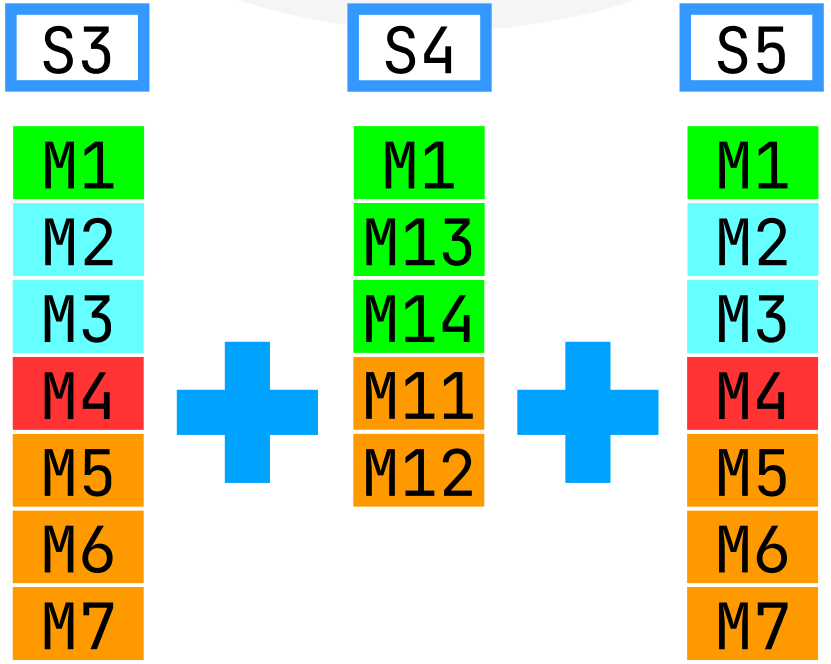

S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M1
M2
M10
M11
M12
S3
M1
M2
M3
M4
M5
M6
M7
S4
M1
M13
M14
M11
M12
S5
M1
M2
M3
M4
M5
M6
M7

A. В глубину

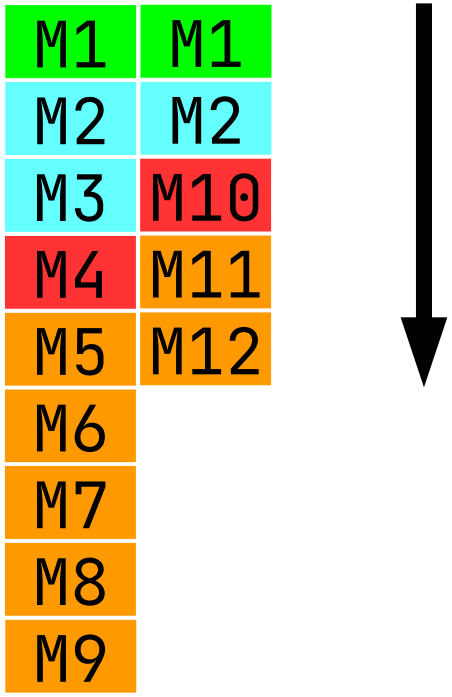

M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M1
M2
M10
M11
M12
S3
M1
M2
M3
M4
M5
M6
M7
S4
M1
M13
M14
M11
M12
S5
M1
M2
M3
M4
M5
M6
M7

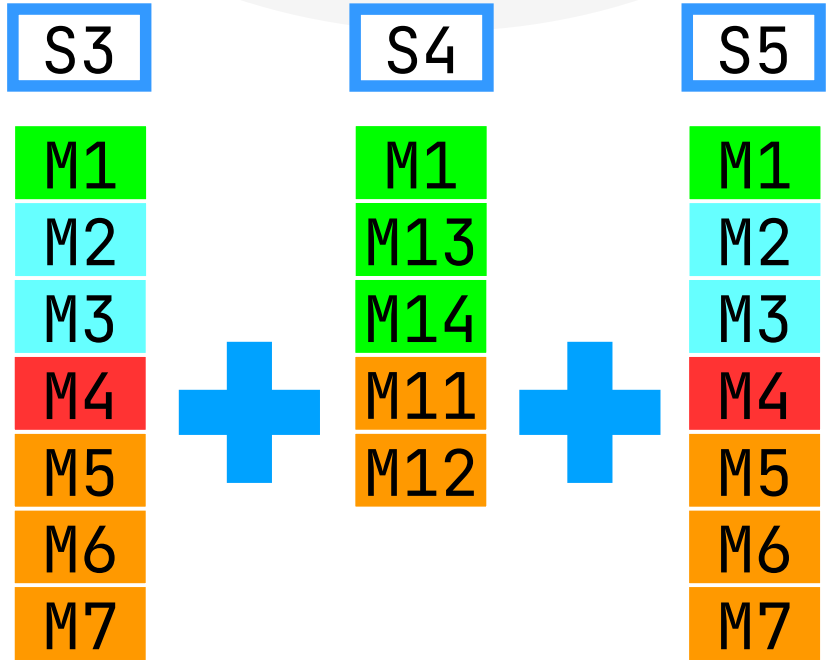
A. В глубину


M1
M2
M3
M4
M5
M6
M7
M8
M9
M1
M2
M10
M11
M12


S3
S4
S5
M1
M2
M3
M4
M5
M6
M7
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

A. В глубину


M1
M2
M3
M4
M5
M6
M7
M8
M9
M1
M2
M10
M11
M12


S3
S4
S5
M1
M2
M3
M4
M5
M6
M7
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

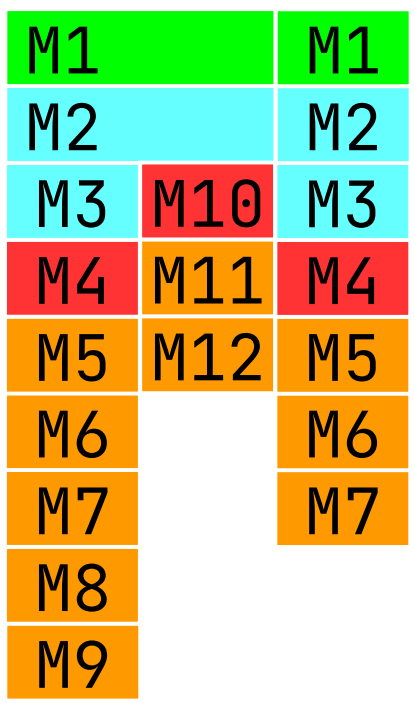
A. В глубину

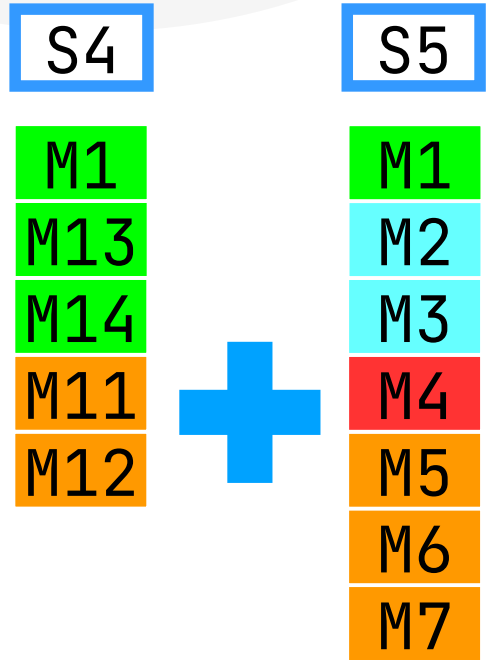

M1
M2
M2
M3
M10
M4
M11
M5
M12
M6
M7
M8
M9
S3
S4
S5
M1
M2
M3
M4
M5
M6
M7
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

A. В глубину

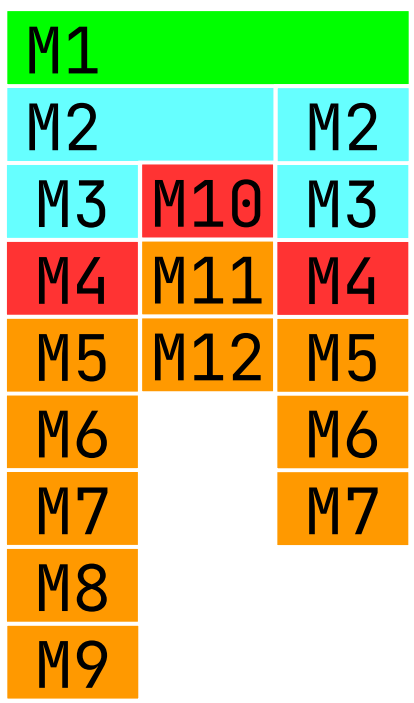

M1
M2
M3
M10
M4
M11
M5
M12
M6
M7
M8
M9
S3
S4
S5
M1
M2
M3
M4
M5
M6
M7
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

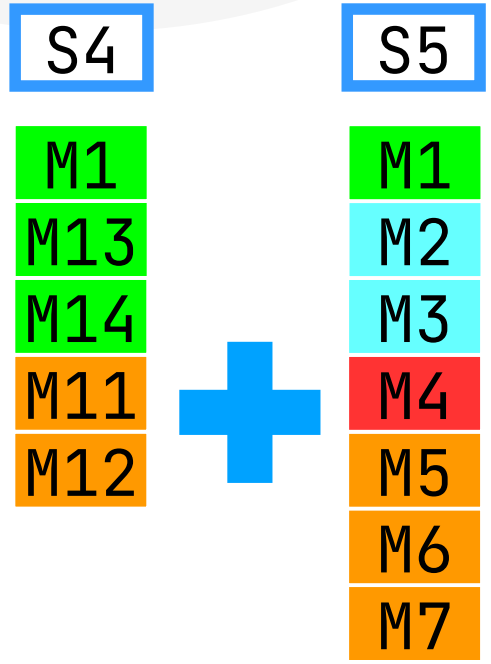
A. В глубину


M1
M2
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M1
M2
M3
M4
M5
M6
M7

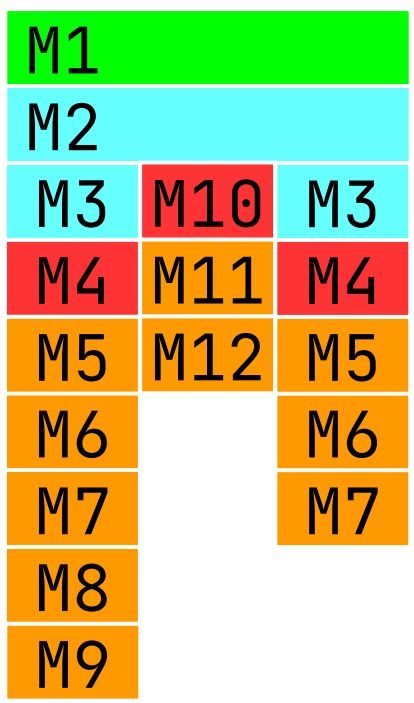

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

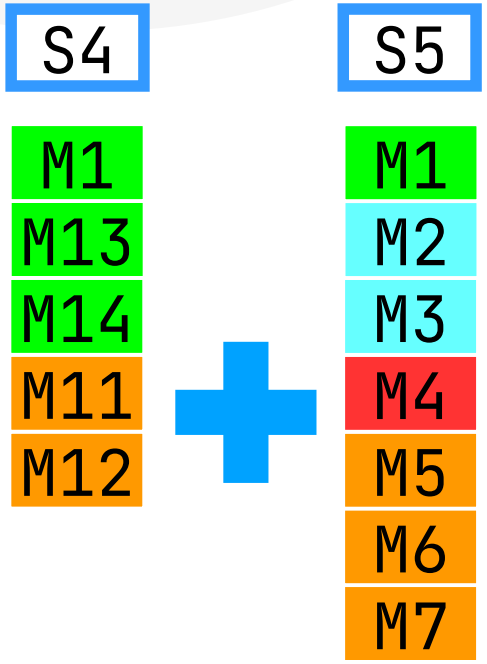
A. В глубину


M1
M2
M2
M3
M10
M3
M4
M11
M4
M5
M12
M5
M6
M6
M7
M7
M8
M9

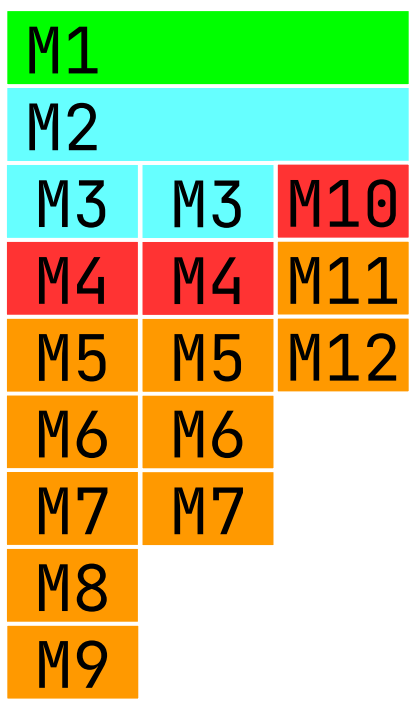

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

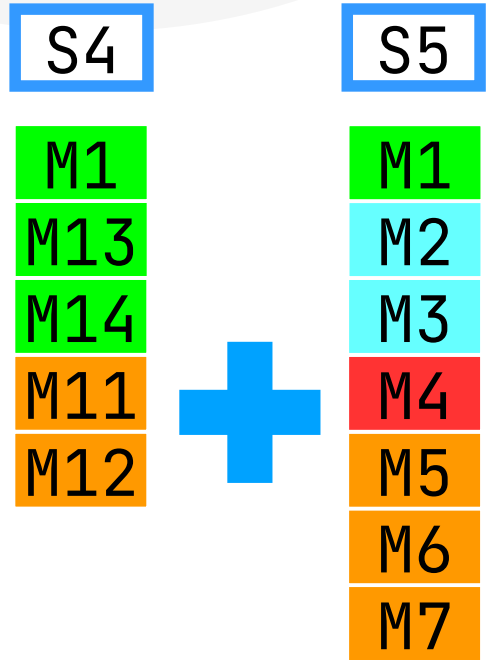
A. В глубину


M1
M2
M3
M10
M3
M4
M11
M4
M5
M12
M5
M6
M6
M7
M7
M8
M9

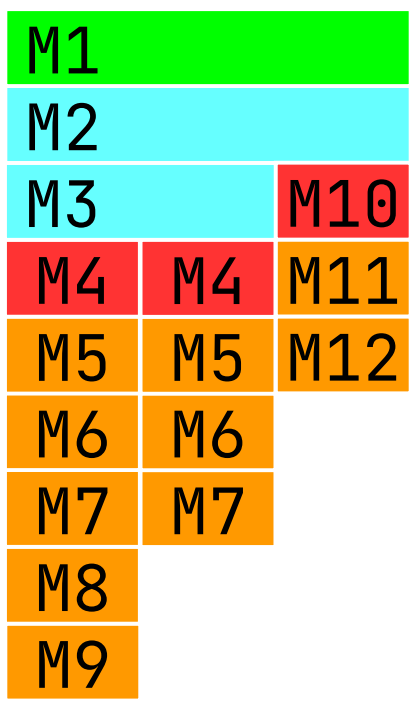

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

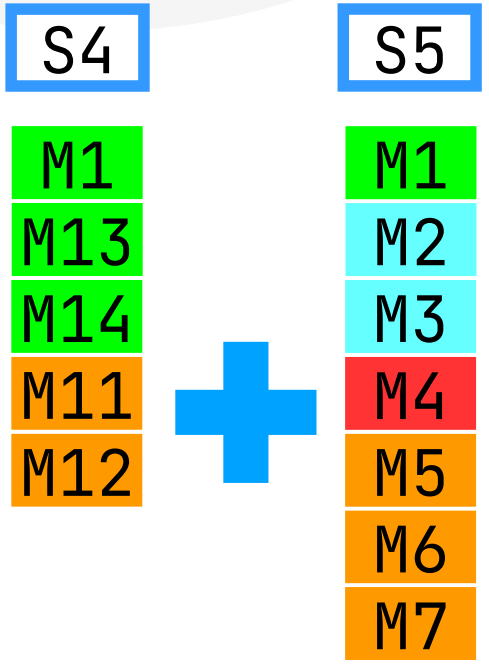
A. В глубину


M1
M2
M3
M3
M10
M4
M4
M11
M5
M5
M12
M6
M6
M7
M7
M8
M9

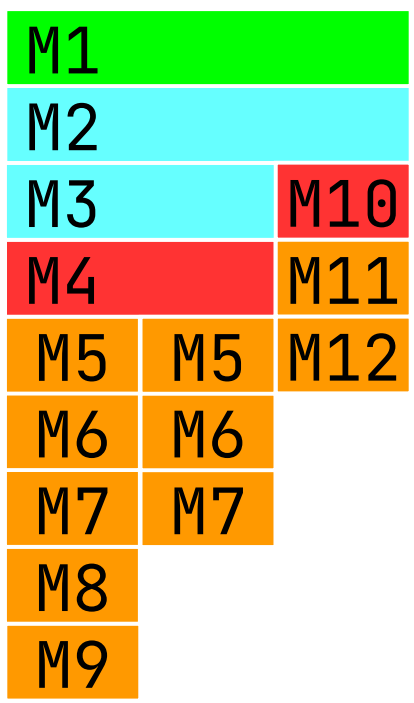

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

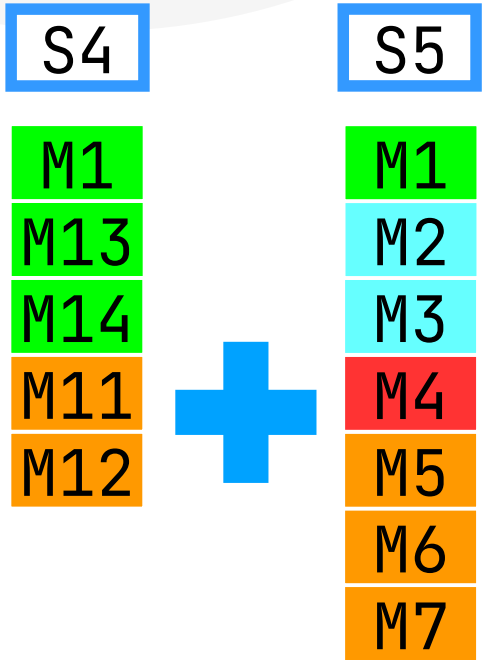
A. В глубину


M1
M2
M3
M10
M4
M4
M11
M5
M5
M12
M6
M6
M7
M7
M8
M9

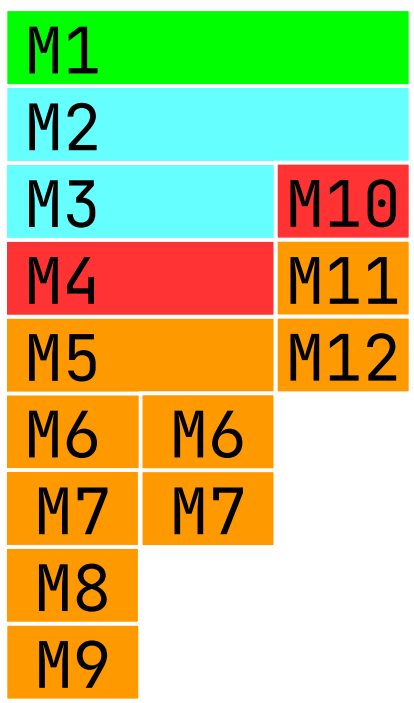

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

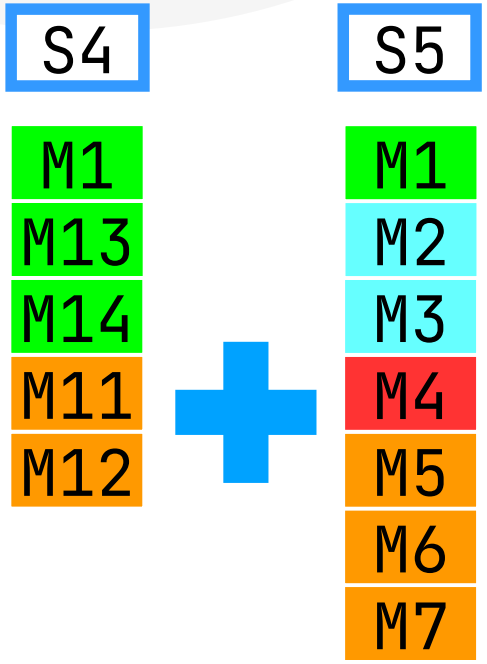
A. В глубину


M1
M2
M3
M10
M4
M11
M5
M5
M12
M6
M6
M7
M7
M8
M9

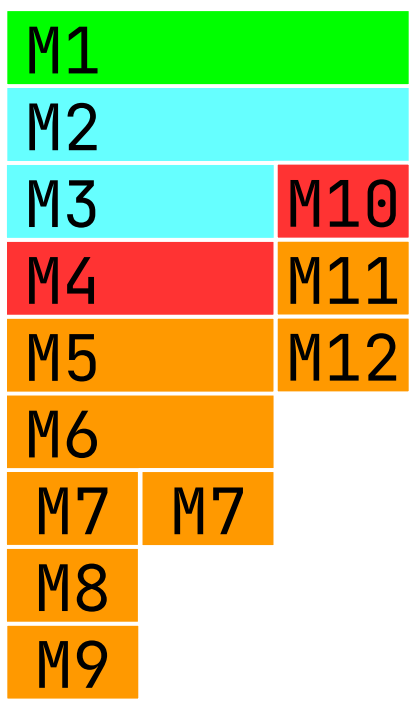

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

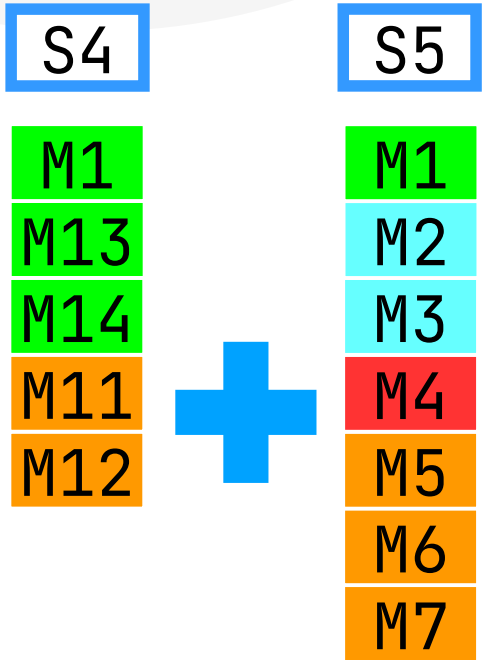
A. В глубину


M1
M2
M3
M10
M4
M11
M5
M12
M6
M6
M7
M7
M8
M9

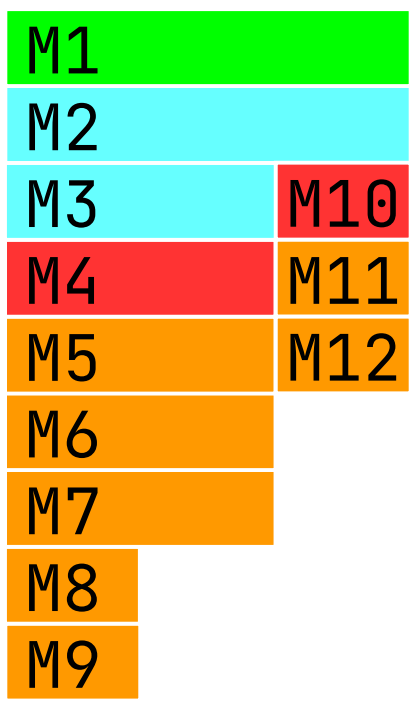

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

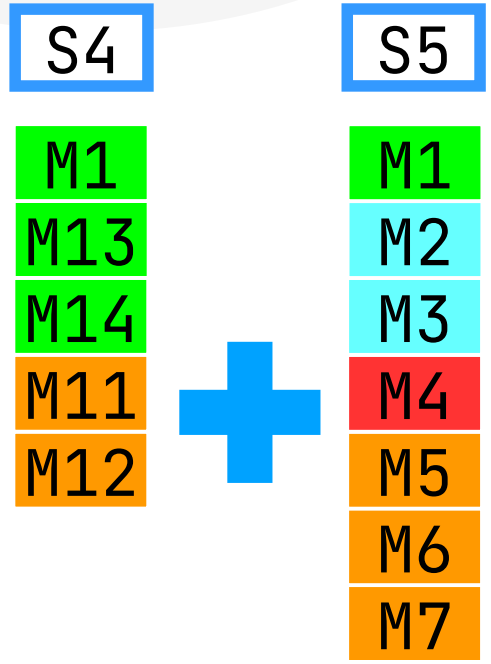
A. В глубину


M1
M2
M3
M10
M4
M11
M5
M12
M6
M7
M7
M8
M9

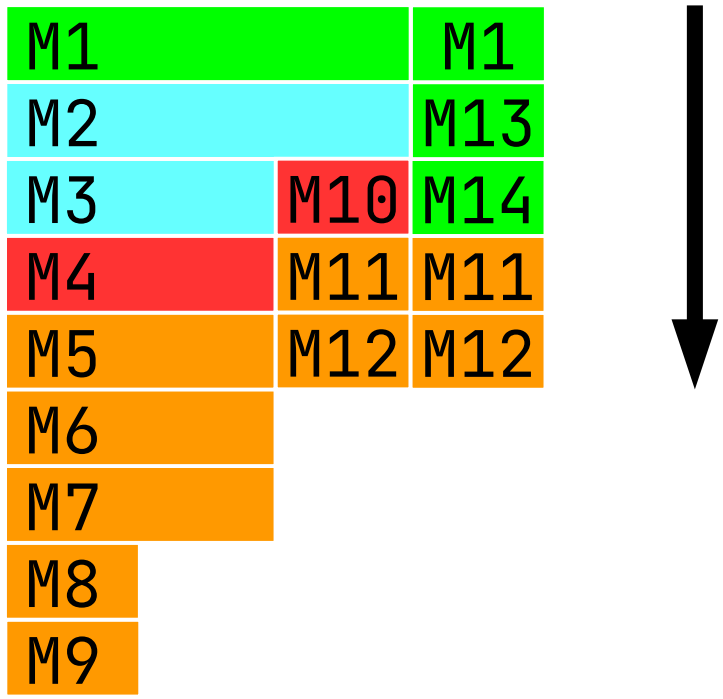

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

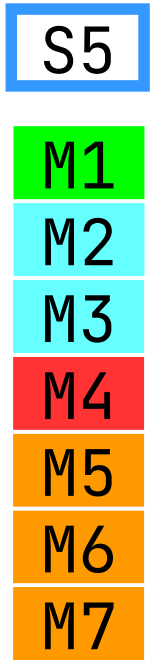
A. В глубину


M1
M2
M3
M10
M4
M11
M5
M12
M6
M7
M8
M9

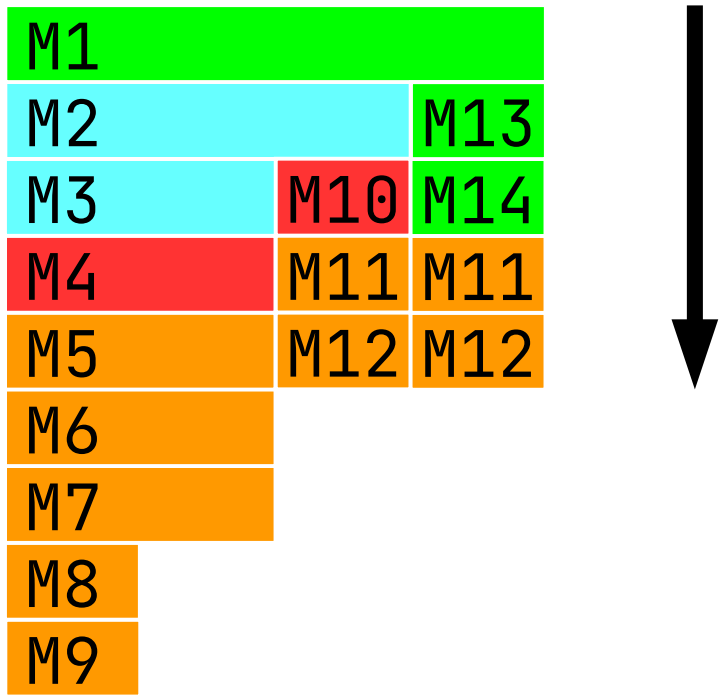

S4
S5
M1
M13
M14
M11
M12
M1
M2
M3
M4
M5
M6
M7

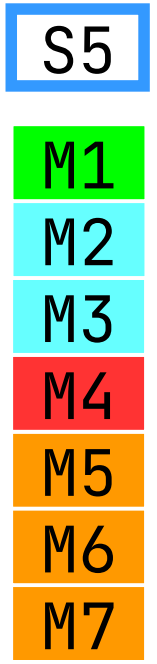
A. В глубину


M1
M1
M2
M13
M3
M10
M14
M4
M11
M11
M5
M12
M12
M6
M7
M8
M9

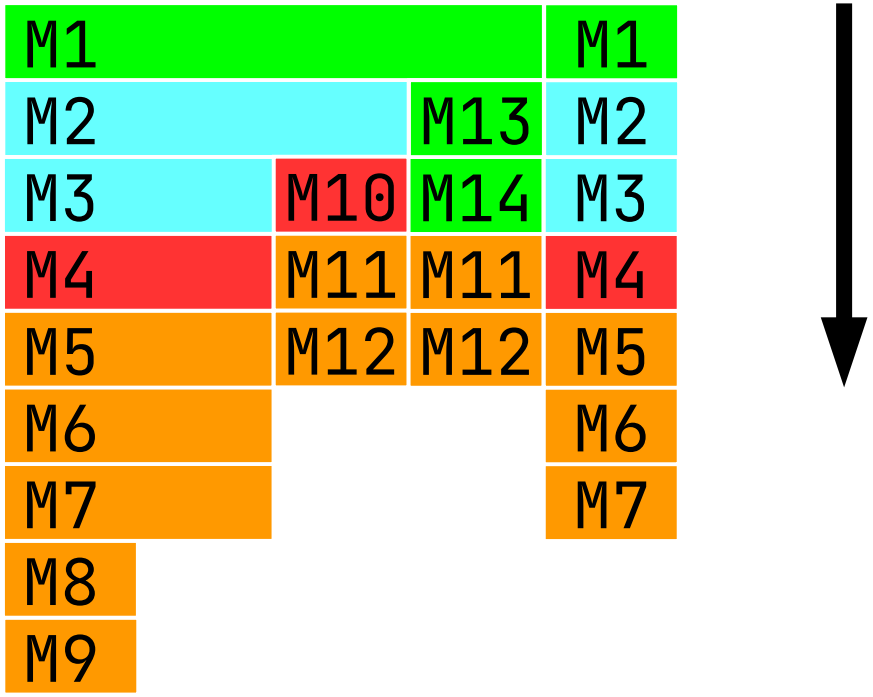

S5
M1
M2
M3
M4
M5
M6
M7

A. В глубину

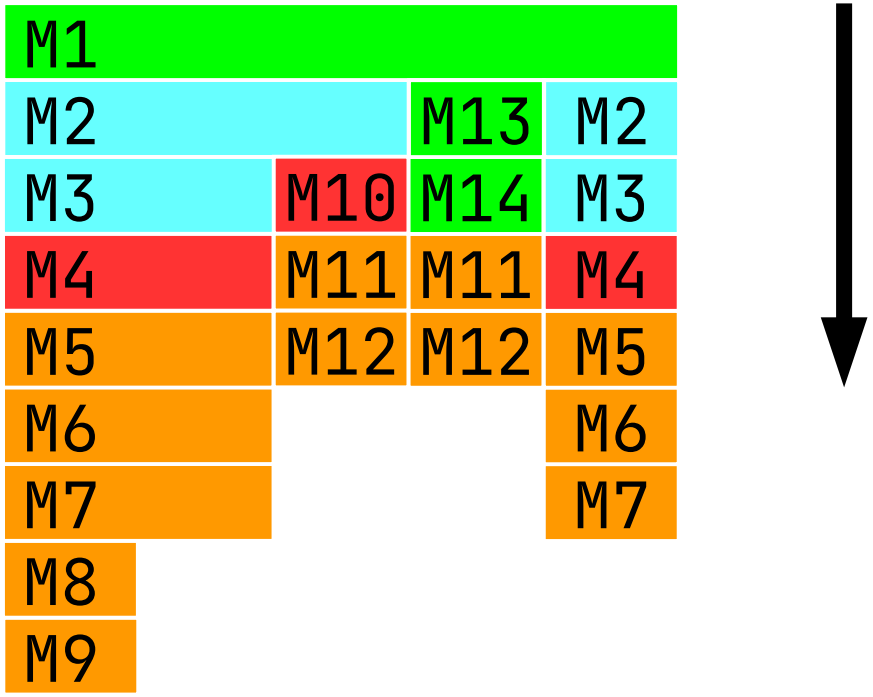

M1
M2
M13
M3
M10
M14
M4
M11
M11
M5
M12
M12
M6
M7
M8
M9


S5
M1
M2
M3
M4
M5
M6
M7

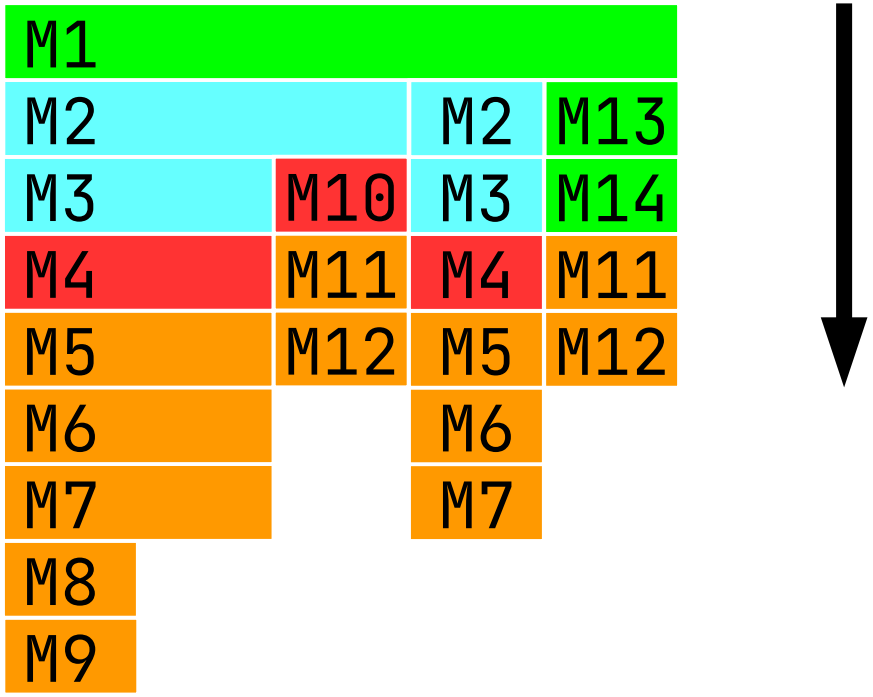
A. В глубину


M1
M1
M2
M13
M2
M3
M10
M14
M3
M4
M11
M11
M4
M5
M12
M12
M5
M6
M6
M7
M7
M8
M9

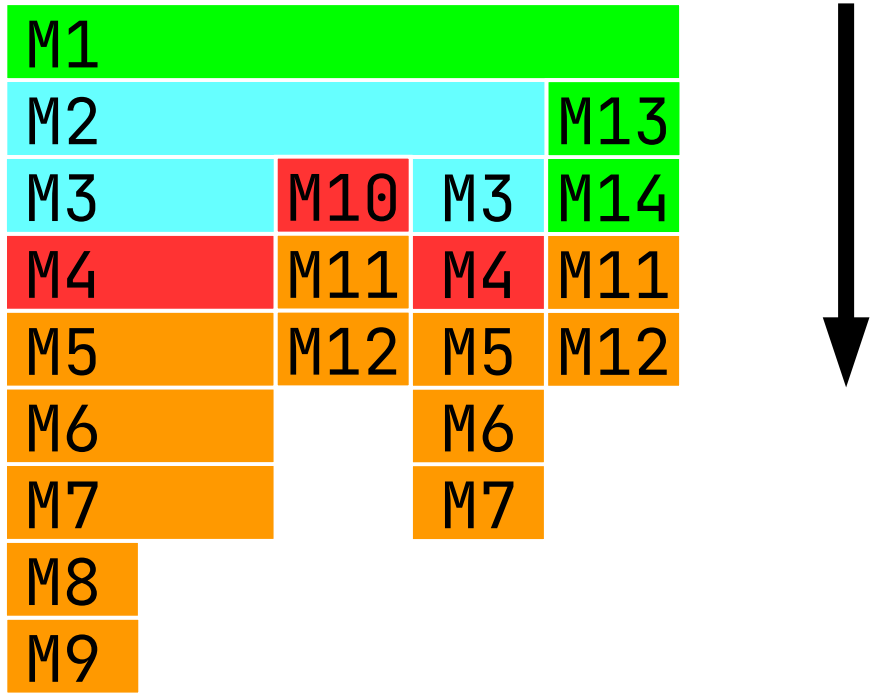
A. В глубину


M1
M2
M13
M2
M3
M10
M14
M3
M4
M11
M11
M4
M5
M12
M12
M5
M6
M6
M7
M7
M8
M9

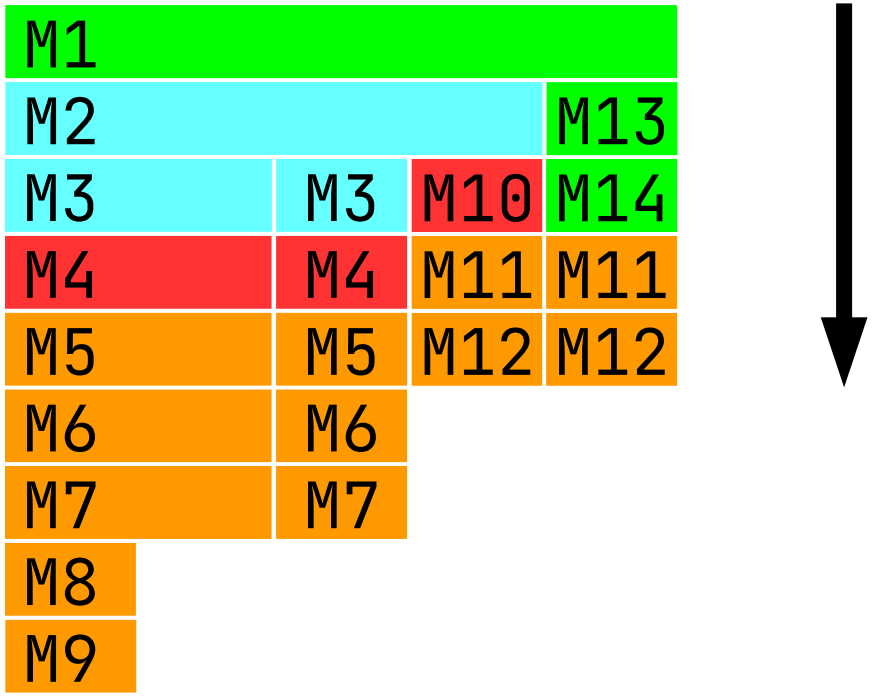
A. В глубину


M1
M2
M2
M13
M3
M10
M3
M14
M4
M11
M4
M11
M5
M12
M5
M12
M6
M6
M7
M7
M8
M9

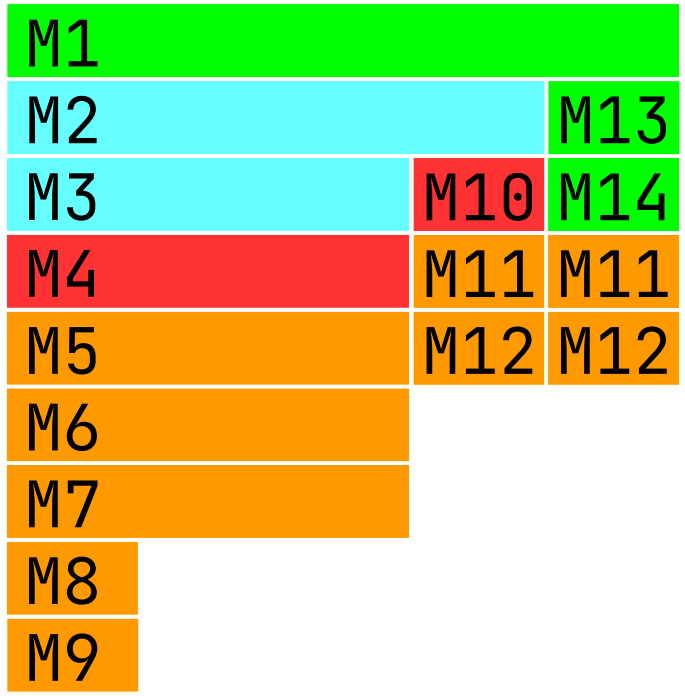
A. В глубину



A. В глубину



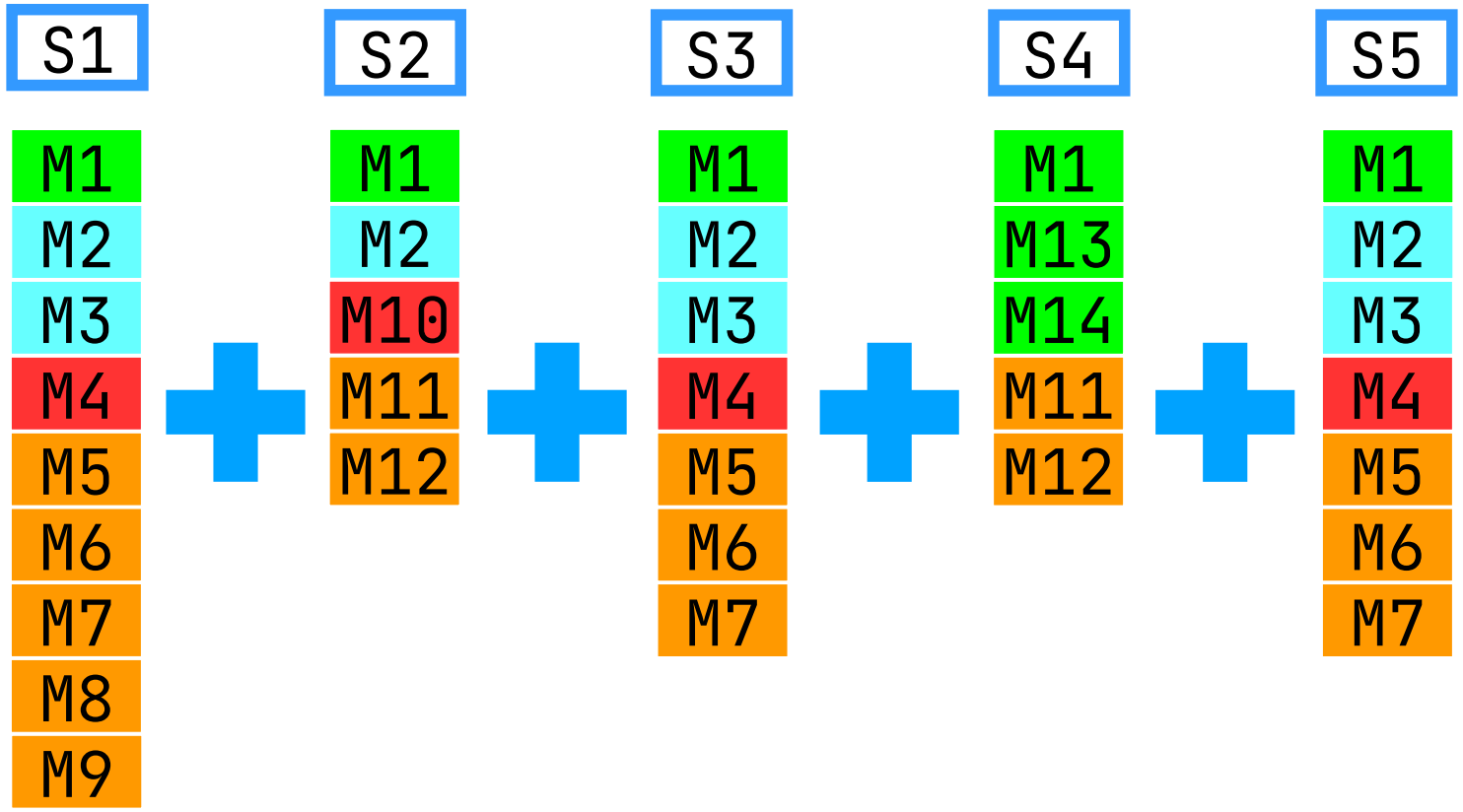
A. В глубину



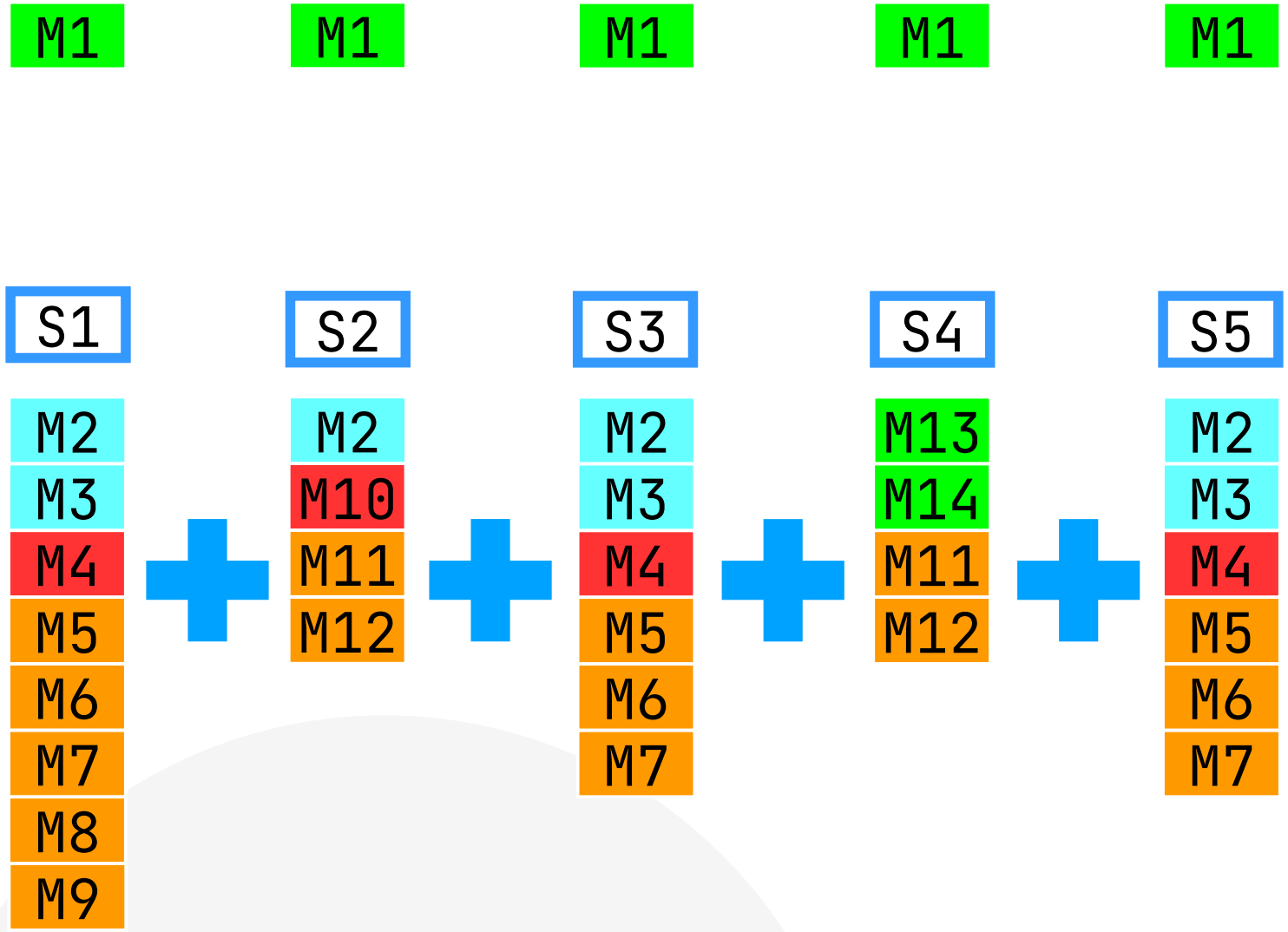
A. В глубину

M1		
M2		M13
M3	M10	M14
M4	M11	M11
M5	M12	M12
M6		
M7		
M8		
M9		

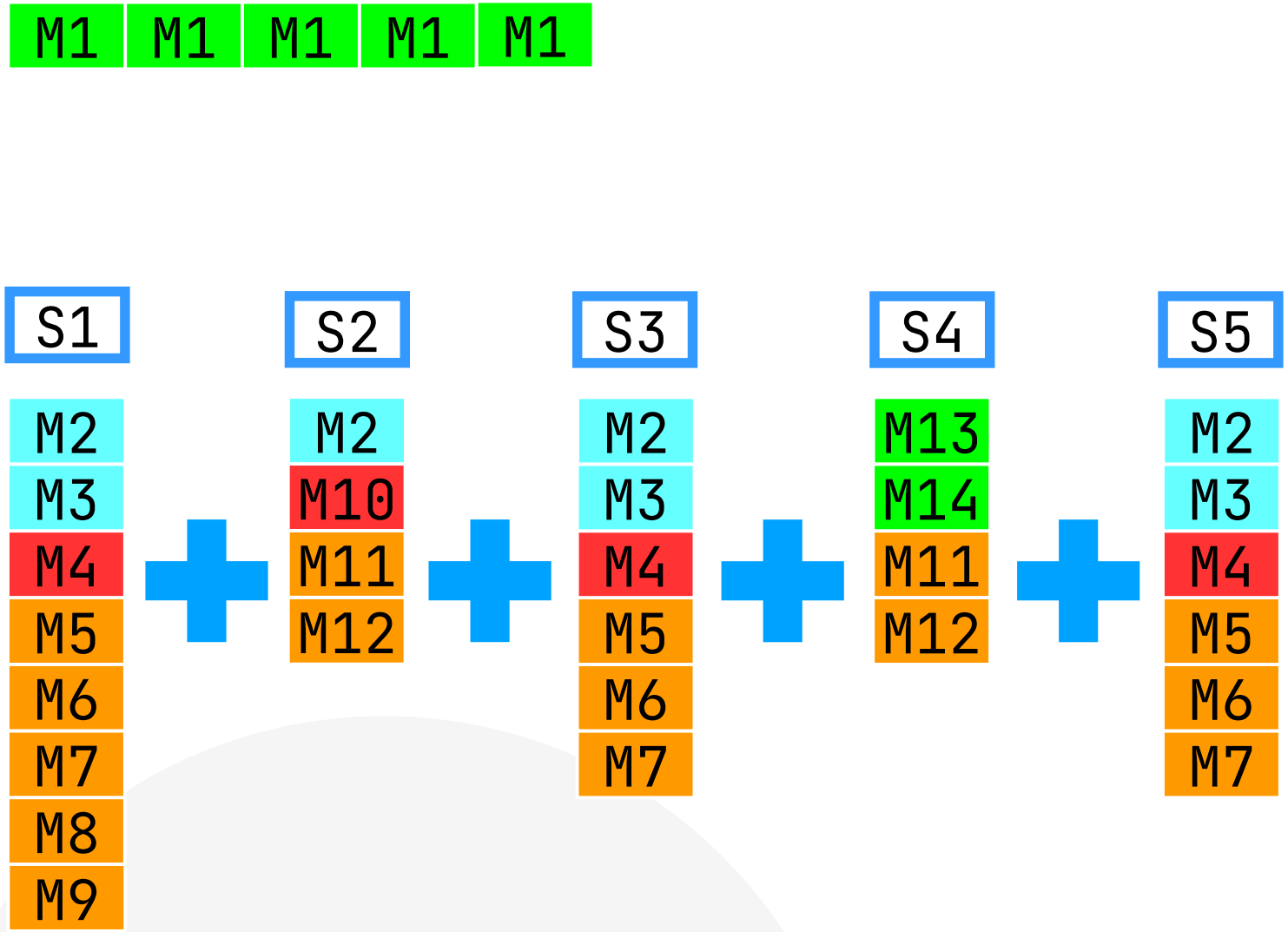
Б. В Ширину


S1
M1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M1
M2
M10
M11
M12
S3
M1
M2
M3
M4
M5
M6
M7
S4
M1
M13
M14
M11
M12
S5
M1
M2
M3
M4
M5
M6
M7

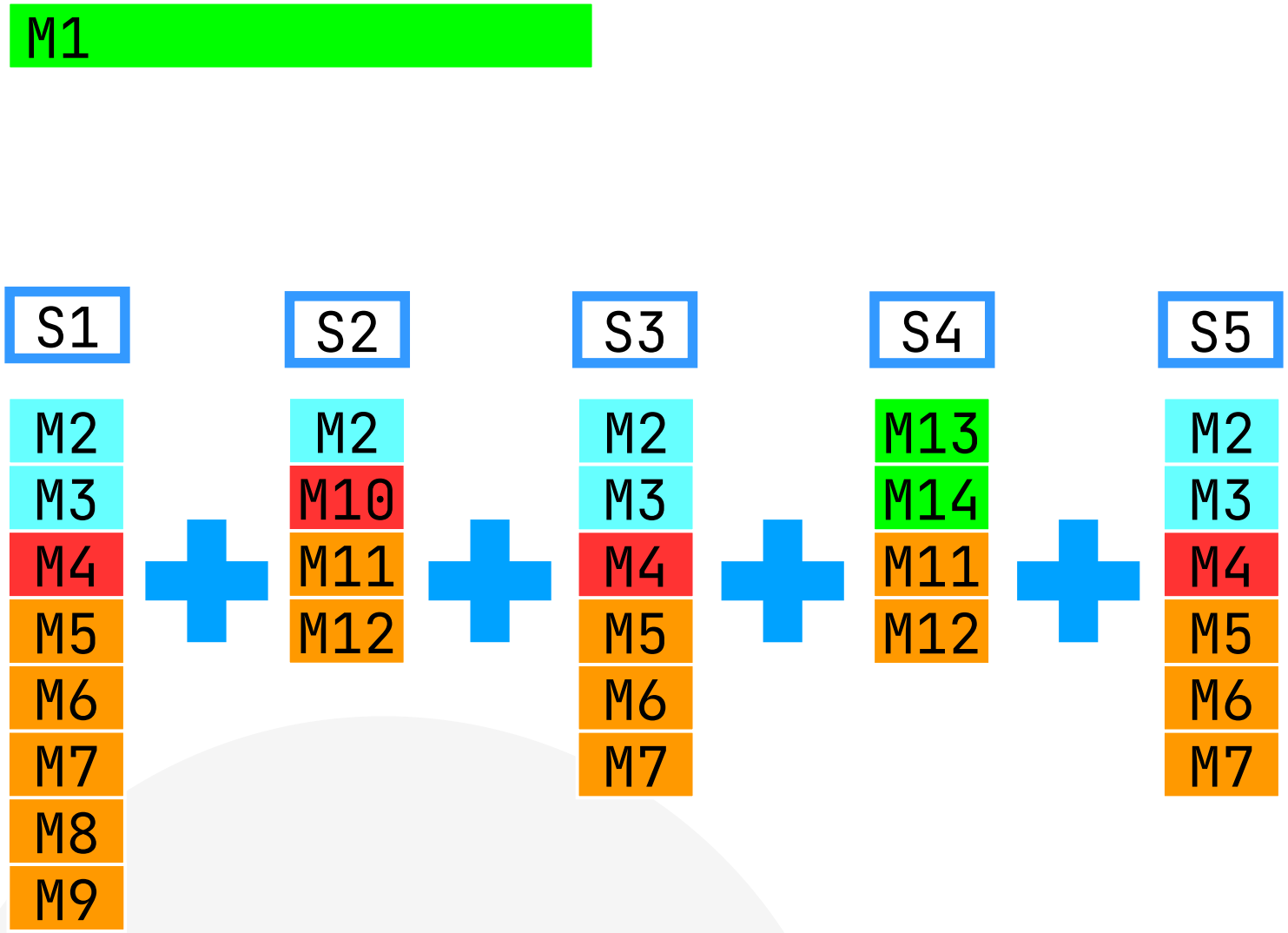
Б. В Ширину


M1
M1
M1
M1
M1
S1
S2
S3
S4
S5
M2
M3
M4
M5
M6
M7
M8
M9
M2
M10
M11
M12
M2
M3
M4
M5
M6
M7
M13
M14
M11
M12
M2
M3
M4
M5
M6
M7

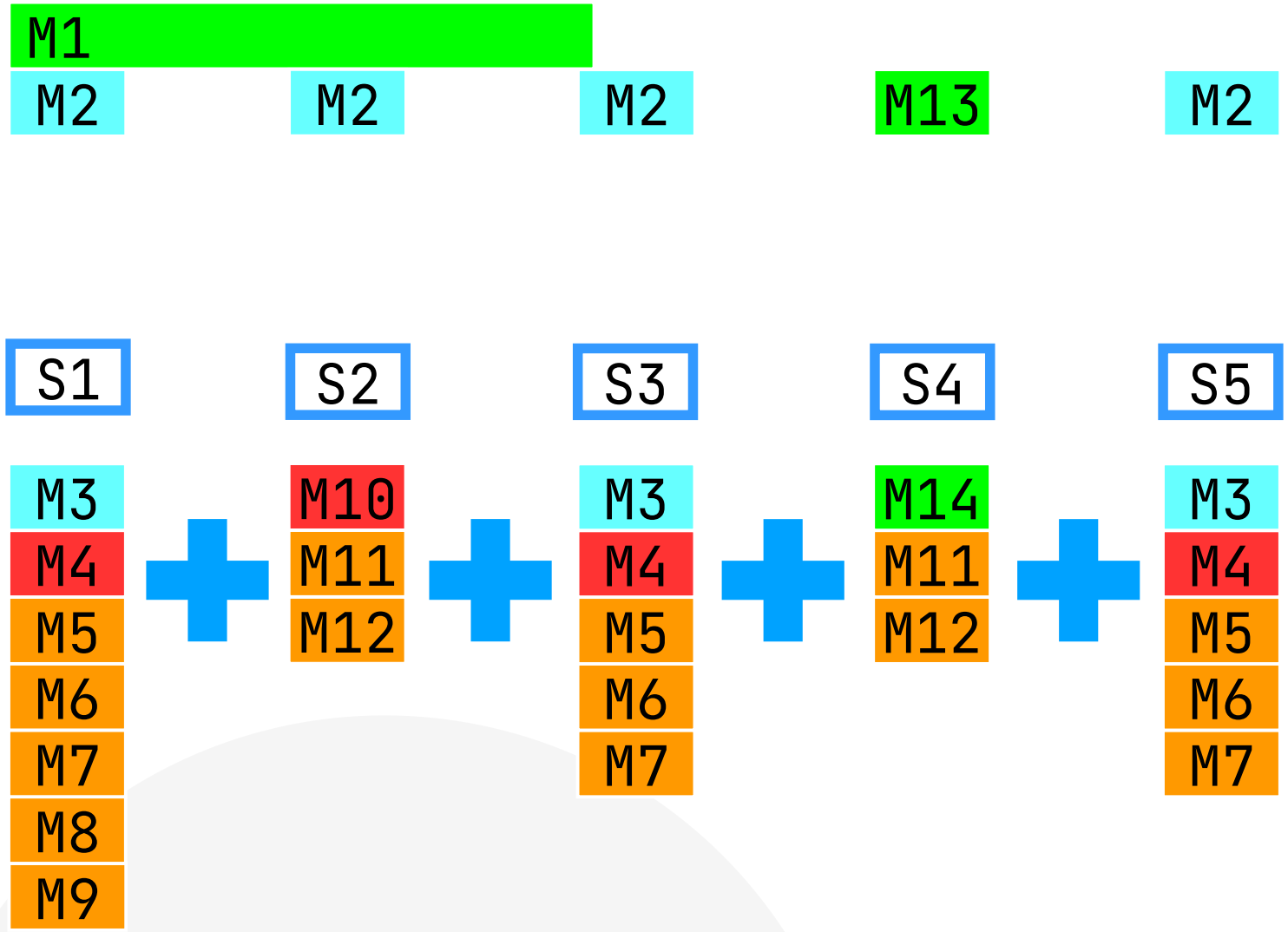
Б. В Ширину


M1 M1 M1 M1 M1
S1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M2
M10
M11
M12
S3
M2
M3
M4
M5
M6
M7
S4
M13
M14
M11
M12
S5
M2
M3
M4
M5
M6
M7

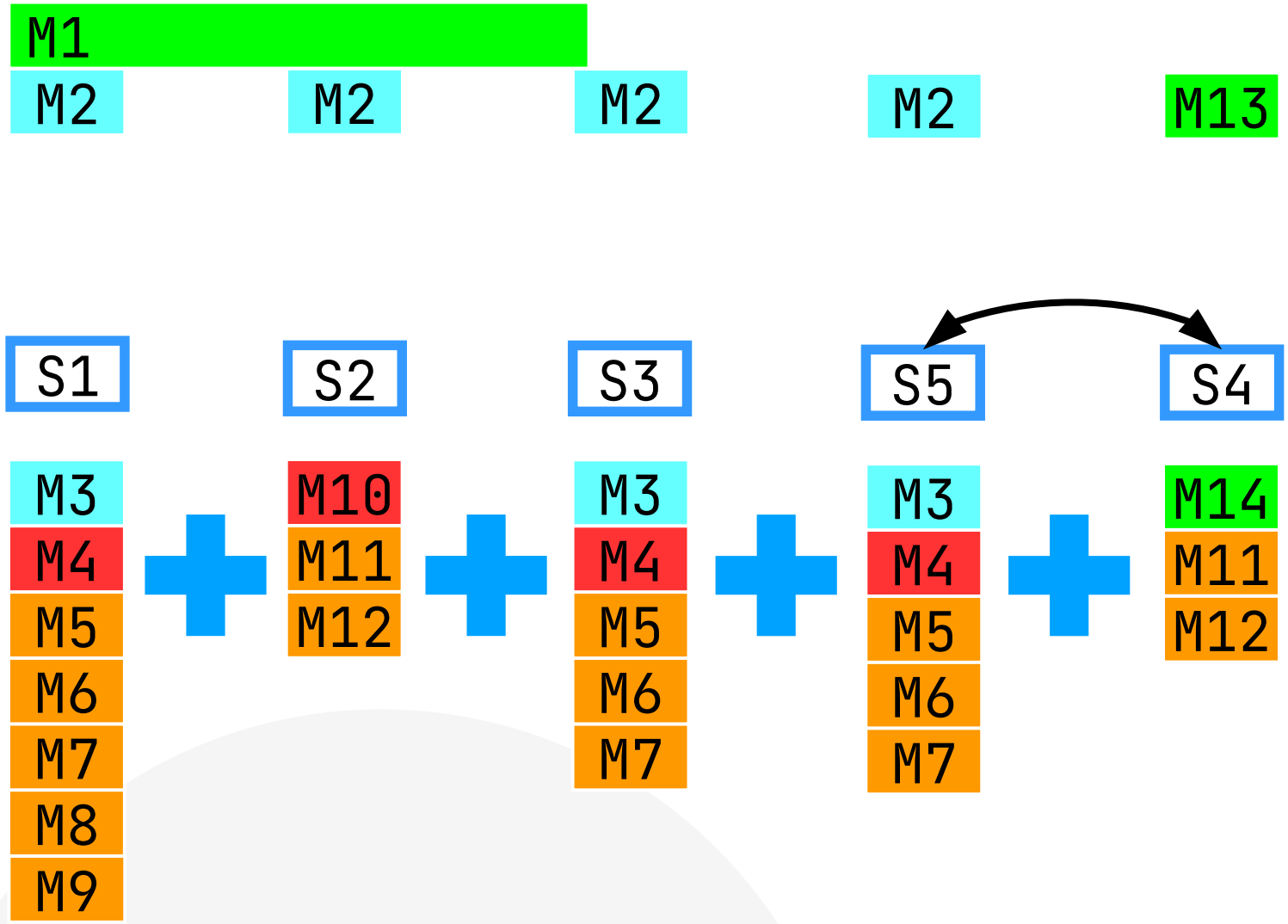
Б. В Ширину


M1
S1
M2
M3
M4
M5
M6
M7
M8
M9
S2
M2
M10
M11
M12
S3
M2
M3
M4
M5
M6
M7
S4
M13
M14
M11
M12
S5
M2
M3
M4
M5
M6
M7

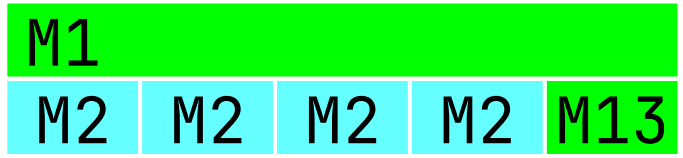
Б. В Ширину

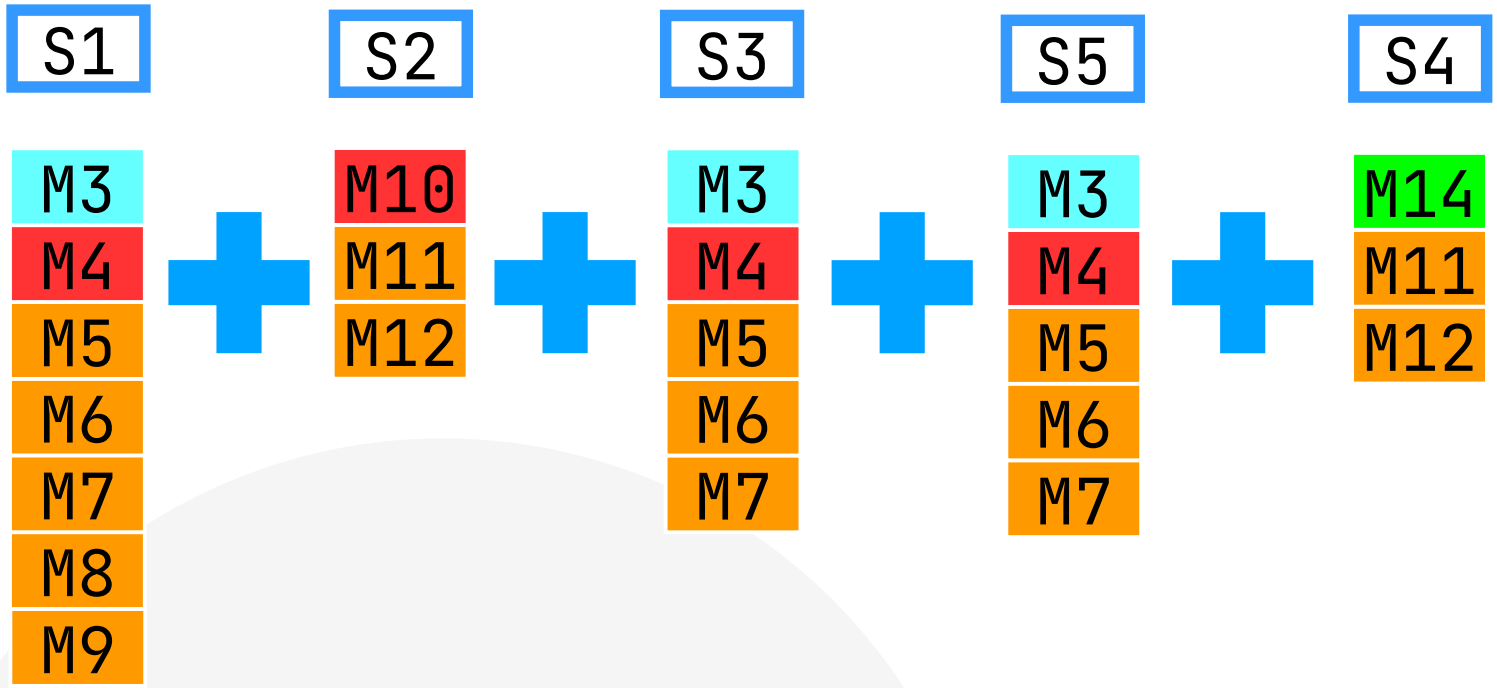

M1
M2
M2
M2
M13
M2
S1
S2
S3
S4
S5
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M3
M4
M5
M6
M7
M14
M11
M12
M3
M4
M5
M6
M7

Б. В Ширину

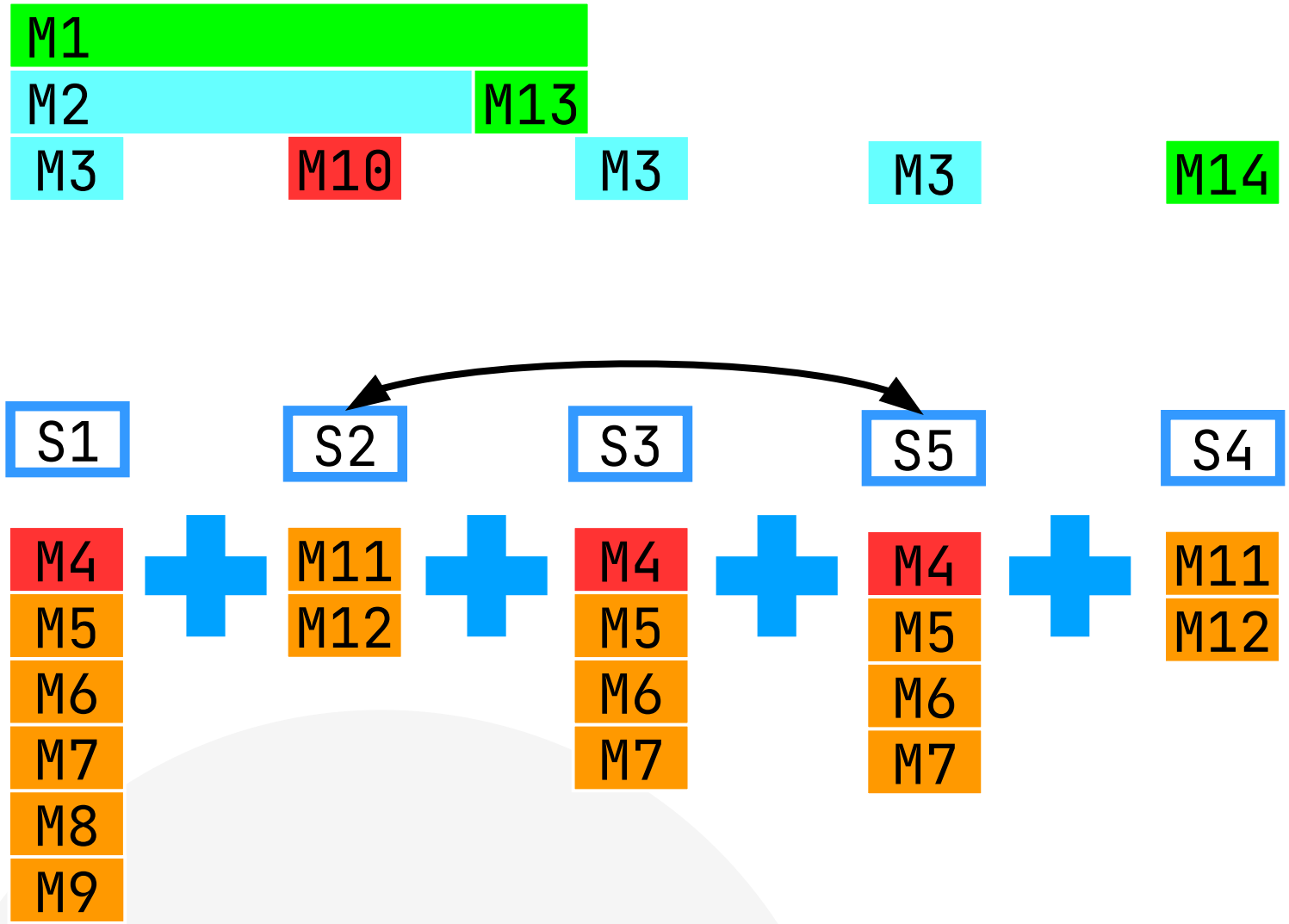

M1
M2
M2
M2
M2
M13
S1
S2
S3
S5
S4
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M3
M4
M5
M6
M7
M3
M4
M5
M6
M7
M14
M11
M12

Б. В Ширину

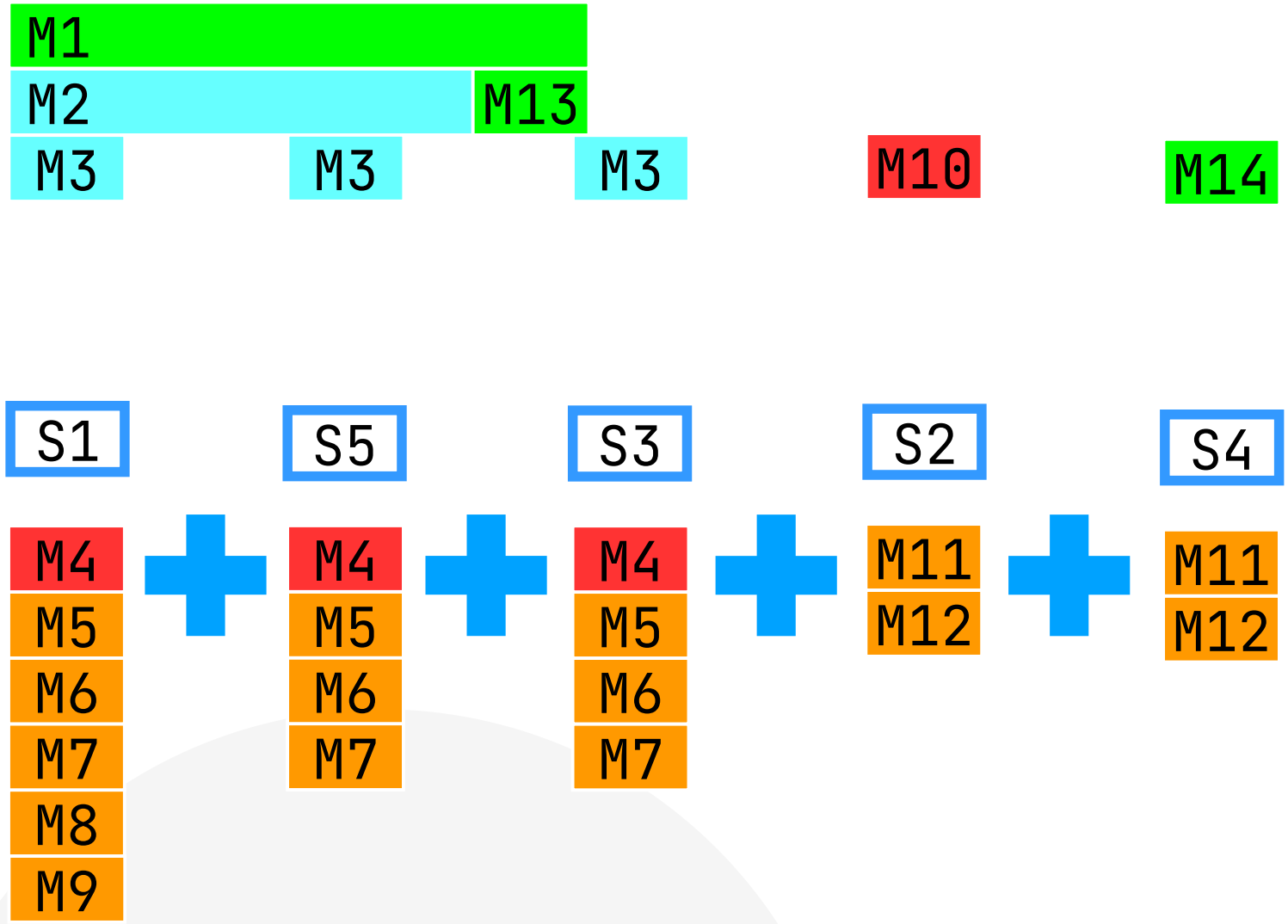

M1
M2
M2
M2
M2
M13


S1
S2
S3
S5
S4
M3
M4
M5
M6
M7
M8
M9
M10
M11
M12
M3
M4
M5
M6
M7
M3
M4
M5
M6
M7
M14
M11
M12

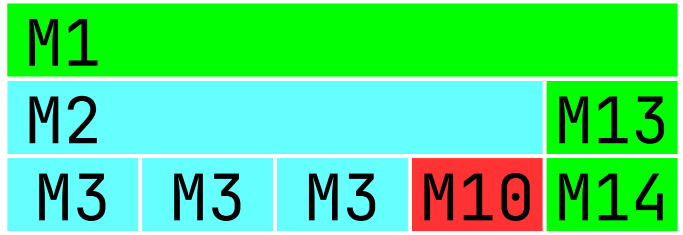
Б. В Ширину

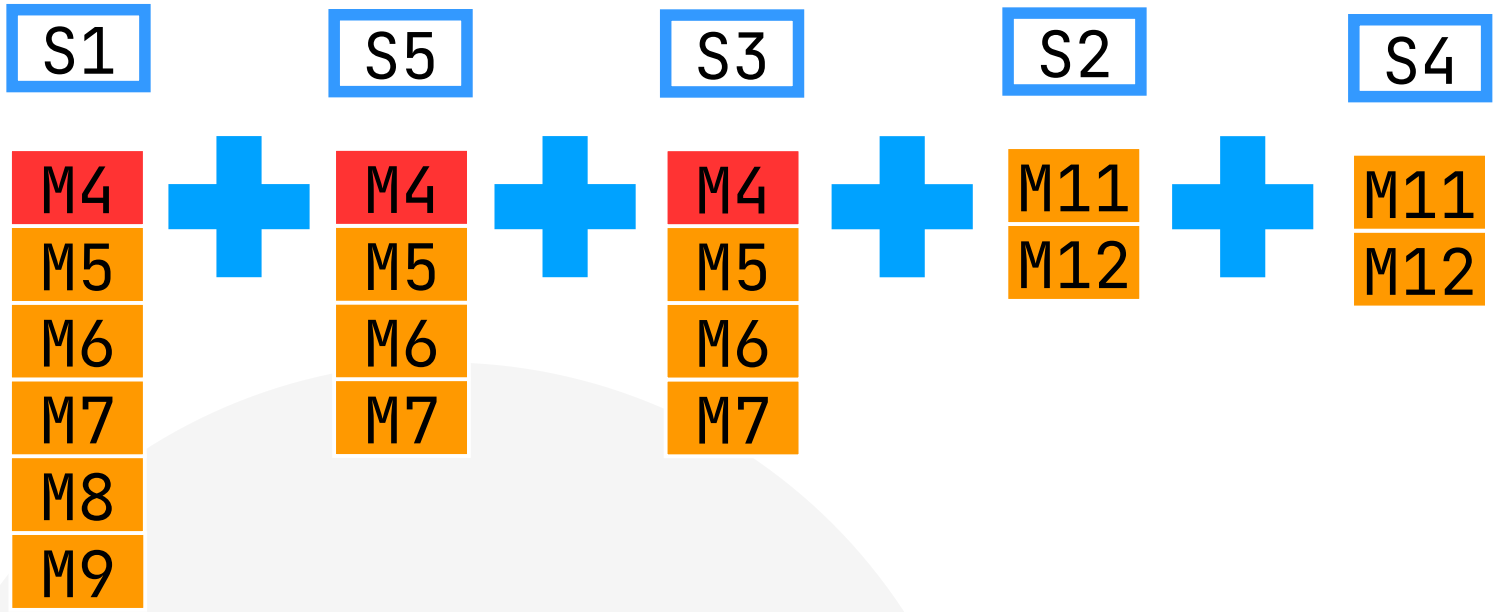

M1
M2
M13
M3
M10
M3
M3
M14
S1
S2
S3
S5
S4
M4
M5
M6
M7
M8
M9
M11
M12
M4
M5
M6
M7
M4
M5
M6
M7
M11
M12

Б. В Ширину


M1
M2
M13
M3
M3
M3
M10
M14
S1
S5
S3
S2
S4
M4
M5
M6
M7
M8
M9
M4
M5
M6
M7
M4
M5
M6
M7
M11
M12
M11
M12

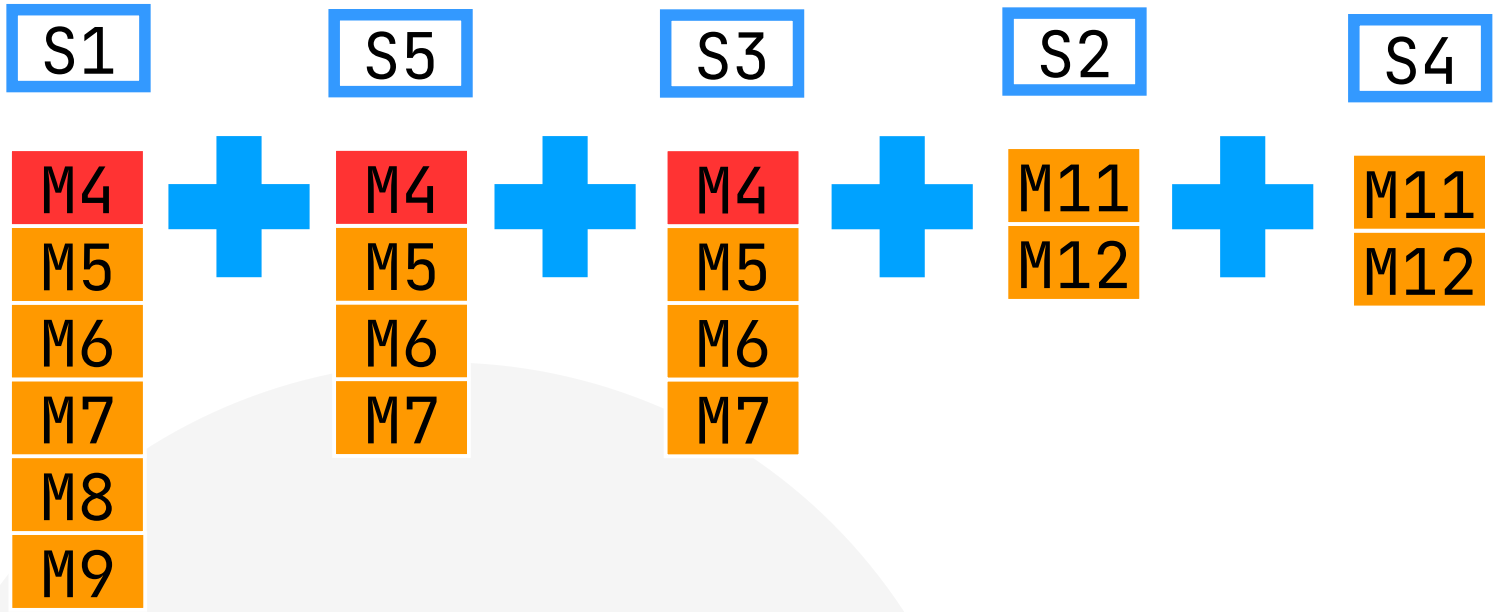
Б. В Ширину


M1
M2
M13
M3
M3
M3
M10
M14


S1
S5
S3
S2
S4
M4
M5
M6
M7
M8
M9
M4
M5
M6
M7
M4
M5
M6
M7
M11
M12
M11
M12

Б. В Ширину


M1
M2
M13
M3
M10
M14


S1
S5
S3
S2
S4
M4
M5
M6
M7
M8
M9
M4
M5
M6
M7
M4
M5
M6
M7
M11
M12
M11
M12

Б. В Ширину


M1
M2
M13
M3
M10
M14


M4
M5
M6
M7
M8
M9

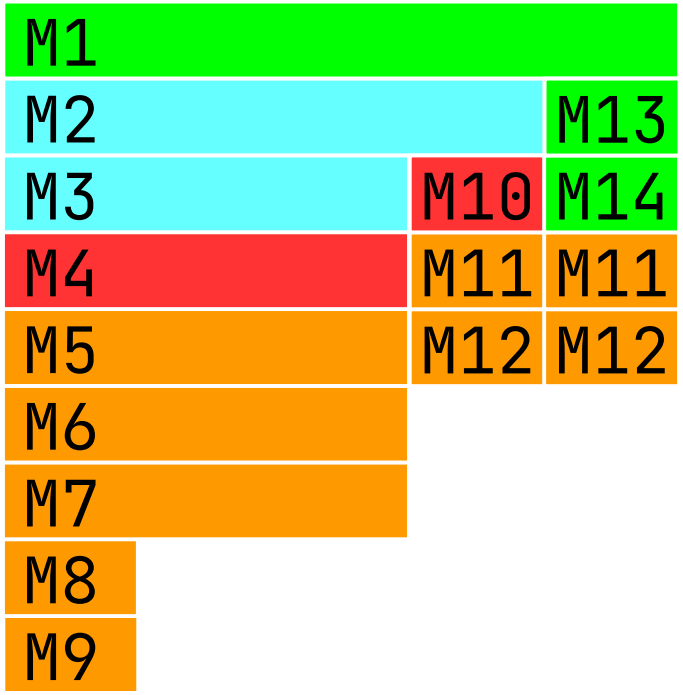

M4
M5
M6
M7


M4
M5
M6
M7

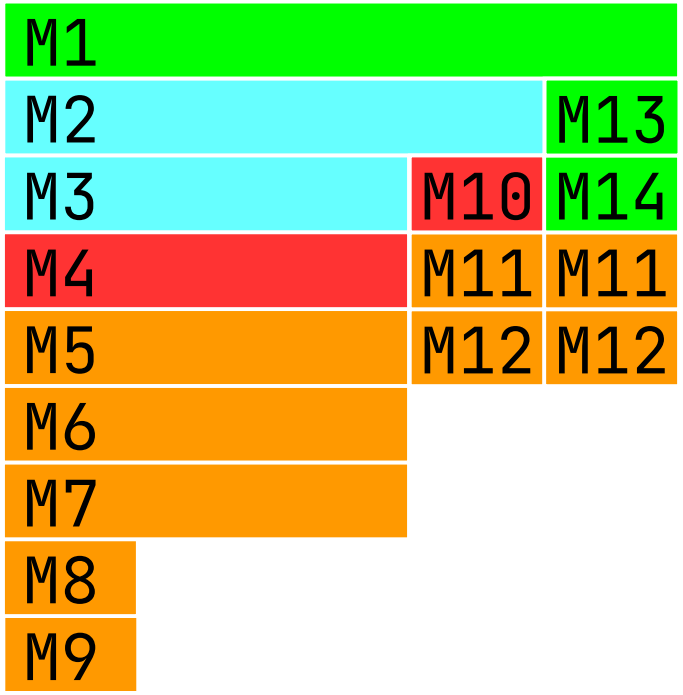

M11
M12


M11
M12

Б. В Ширину



Как рисуем?



в ширину vs в глубину

* в глубину (обычно) быстрее!

Как рисуем?

в ширину vs в глубину

* в глубину (обычно) быстрее, но докладчик растёт в ширину

Как рисуем?

- В глубину (обычно) быстрее

Как рисуем?

- В глубину (обычно) быстрее
- Но алгоритм потребует аллокаций

Как рисуем?

- В глубину (обычно) быстрее
- Но алгоритм потребует аллокаций
- Полная отрисовка > 10 секунд

Как рисуем?

- В глубину (обычно) быстрее
- Но алгоритм потребует аллокаций
- Полная отрисовка > 10 секунд
- Обход в ширину - итеративен!

Требования к отрисовке

CPU (6 млн сэмплов):

- < 1 секунды на первый уровень
- < 5 секунд на первые 20 уровней

Требования к отрисовке

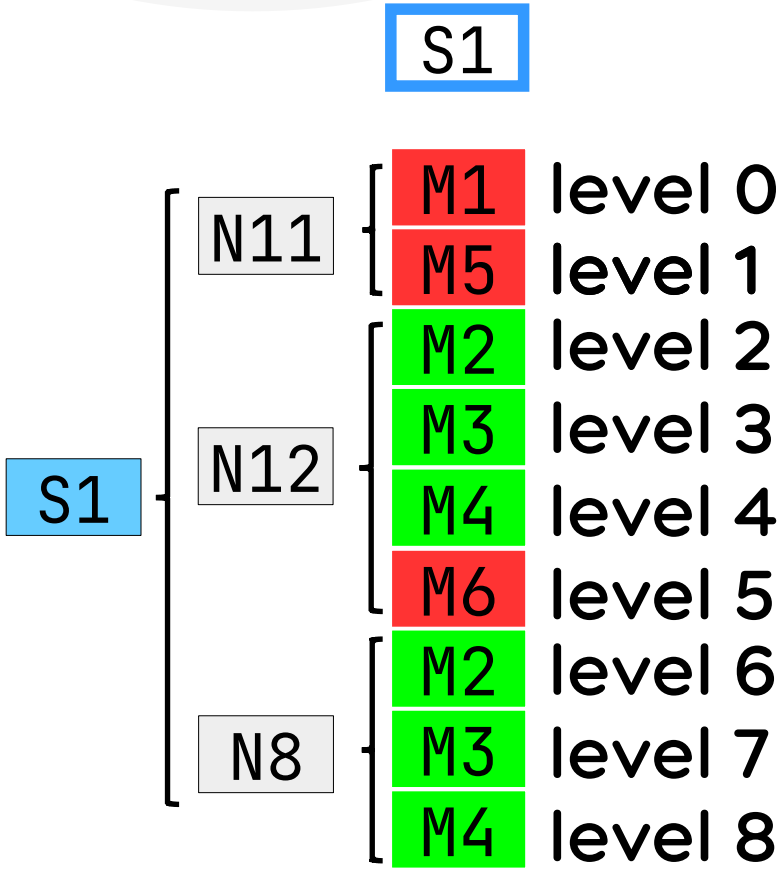
CPU (6 млн сэмплов):

- < 1 секунды на первый уровень
- < 5 секунд на первые 20 уровней

- Alloc (78 млн сэмплов):
- < 5 секунды на первый уровень
- < 10 секунд на первые 20 уровней

Рисуем

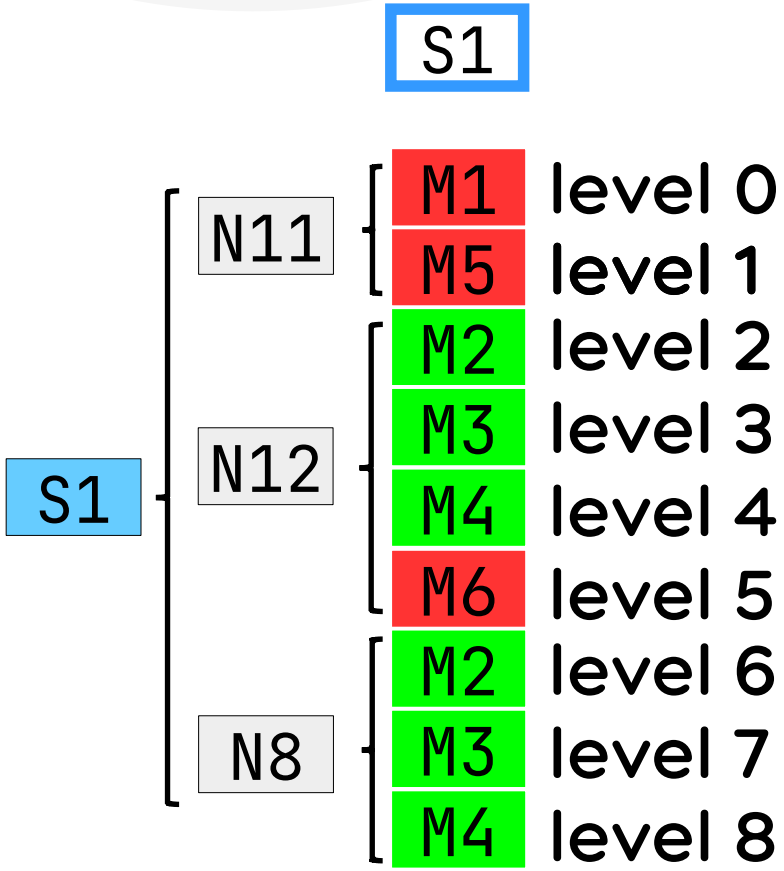
Frame: `{x, width, text, color}`



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

Frame: {x, width, text, color}

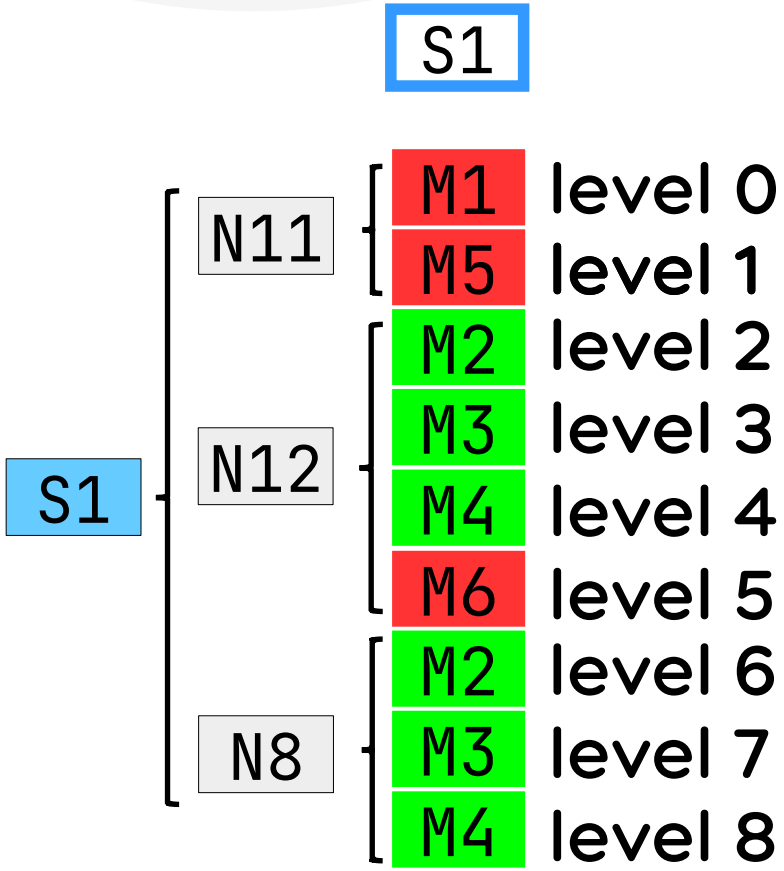


samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

Frame: {x, width, text, color}

f(index,node) = method



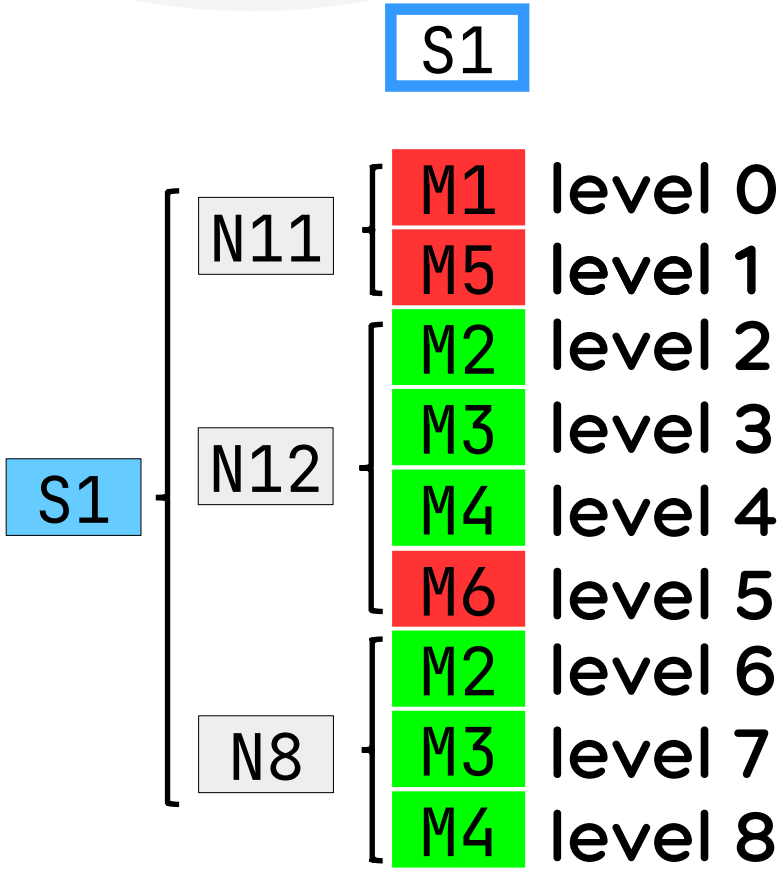
samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

Frame: {x, width, text, color}

f(index,node) = method

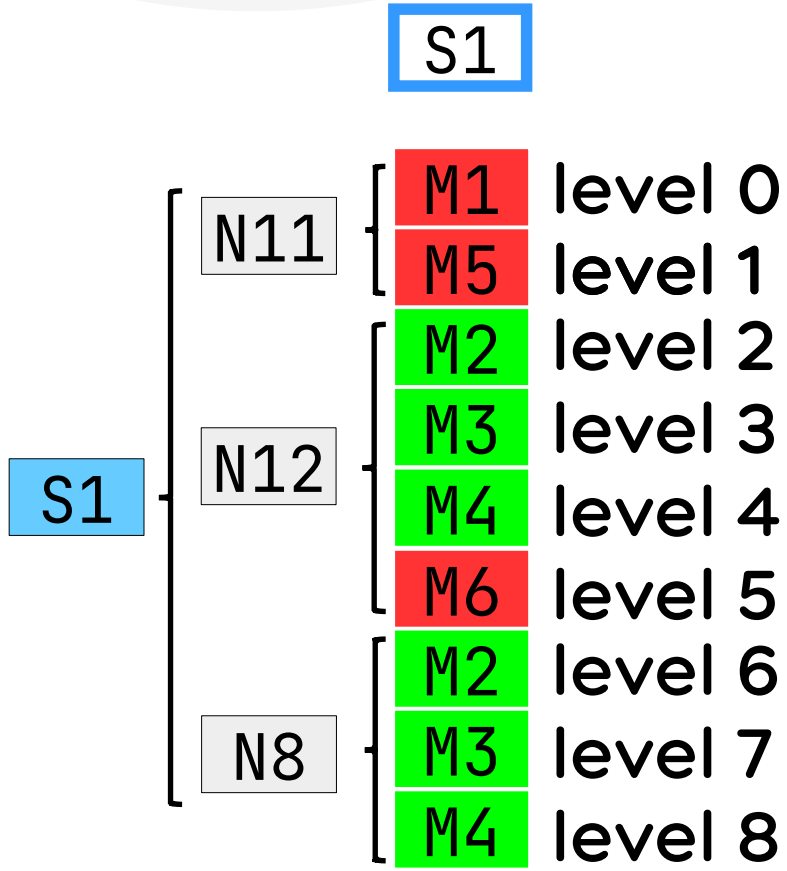
g(level,sample) = method



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

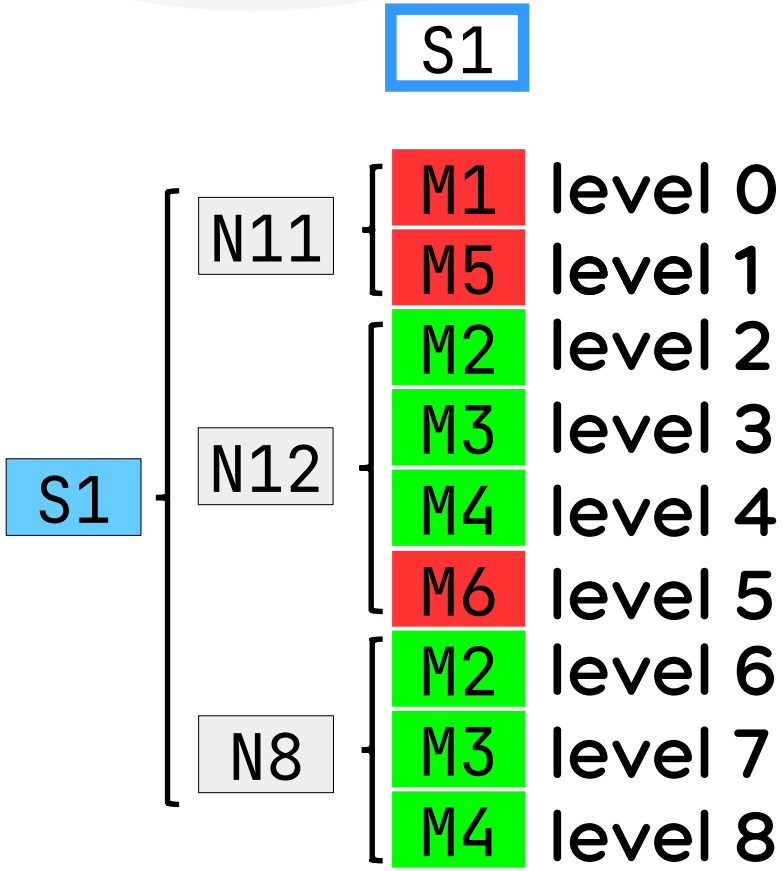
```
f(index,node) - O(1)

g(level,sample) = method
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

```
f(index,node) - O(1)
g(level,sample) - O(nodesCount(sample))
```

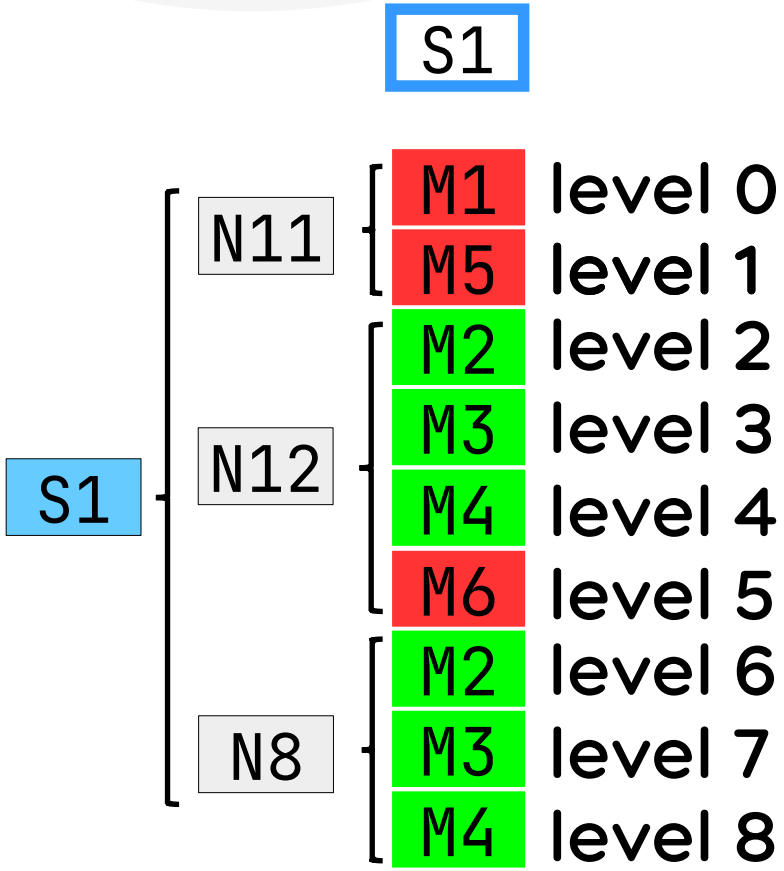


samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

f(index,node) - O(1)

g(level,sample) - ~~O(nodesCount(sample))~~

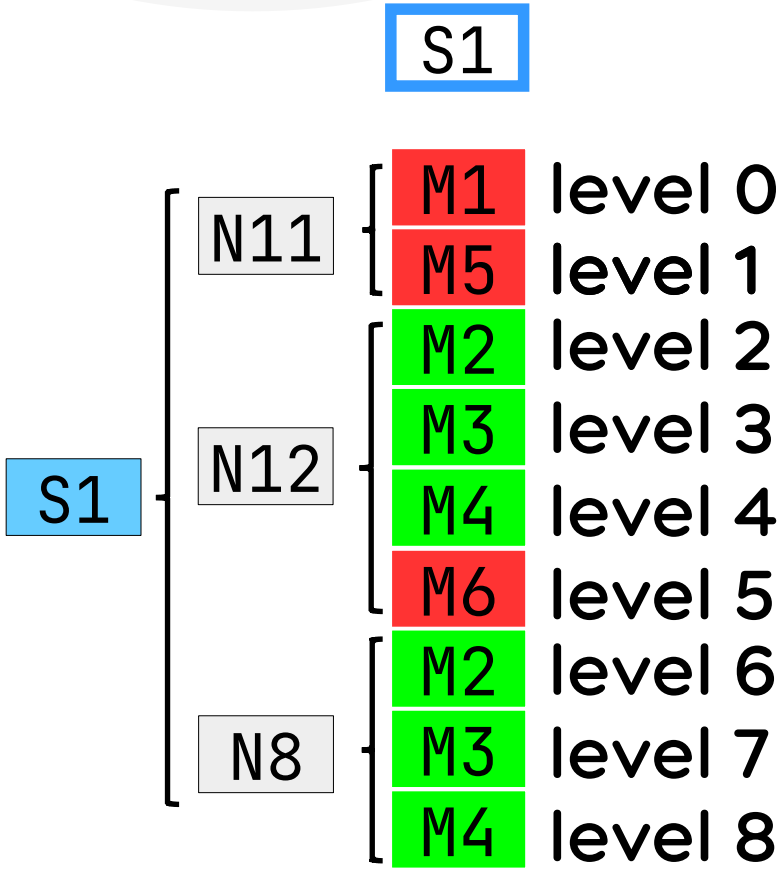


samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

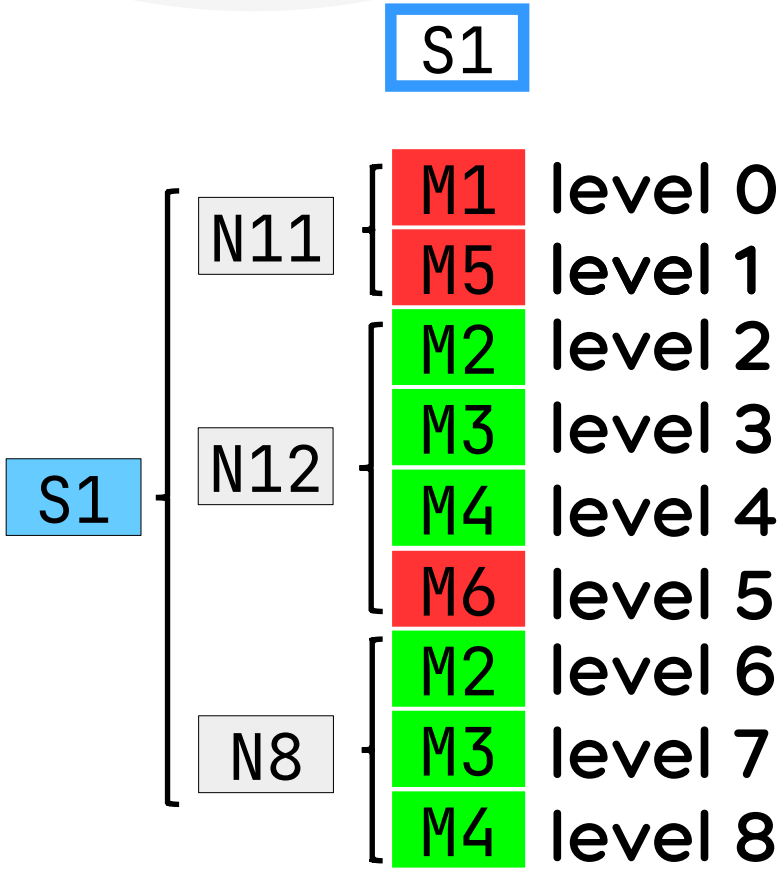
```
f(index,node) - O(1)

g(state,level,sample) - O(1)
```

samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...



Рисуем

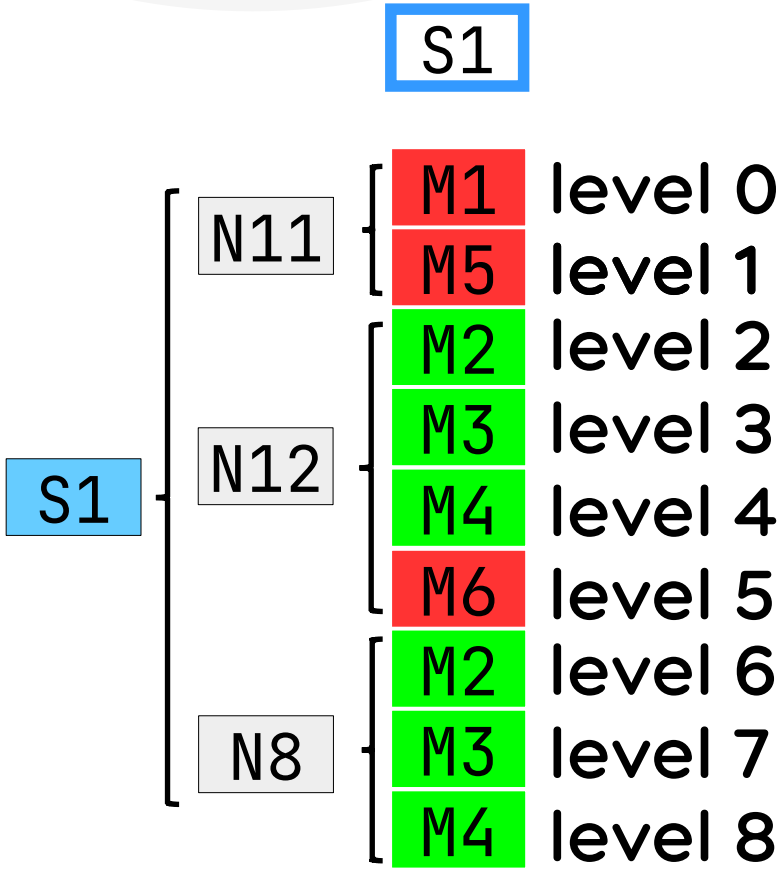
```
f(index,node) - O(1)

g(state,level,sample) = f(i(…),n(…))
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

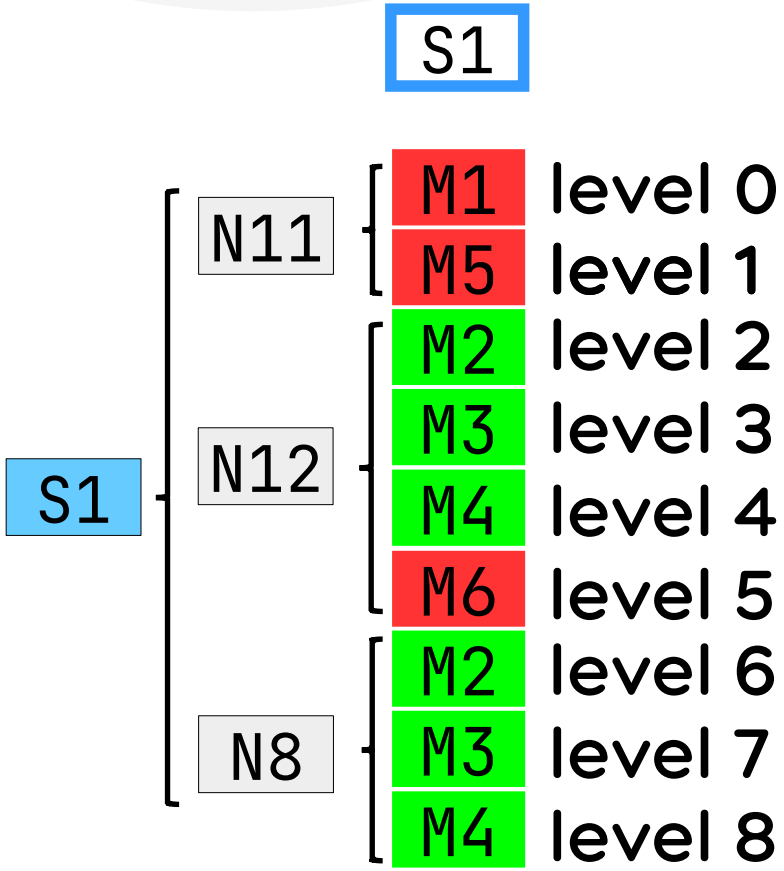
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 0
g(…) = M1 = N11[0]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

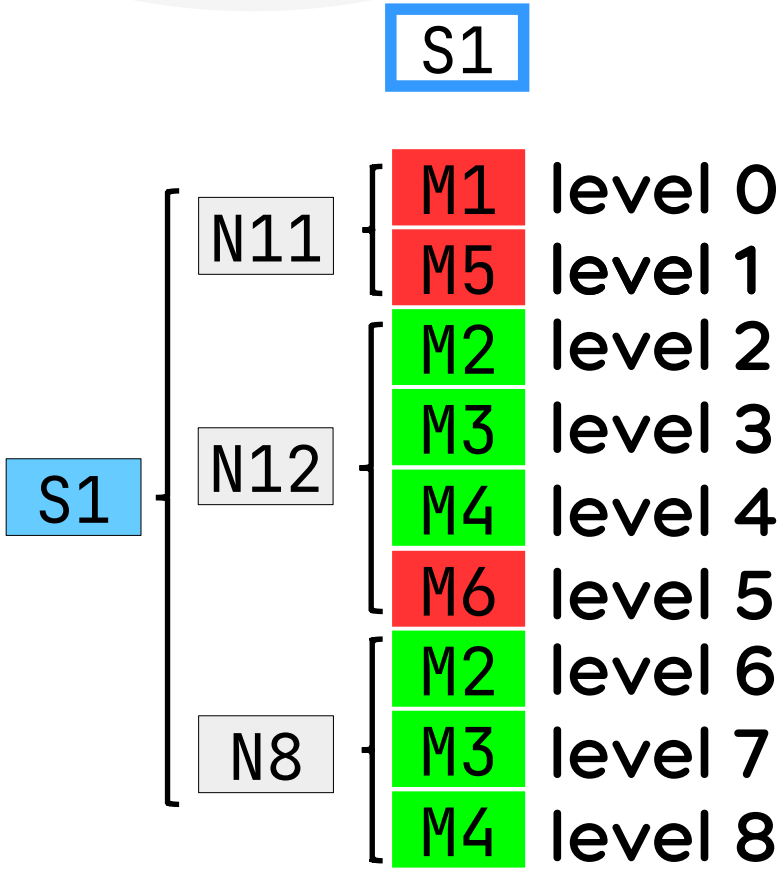
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 1
g(…) = M5 = N11[1]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

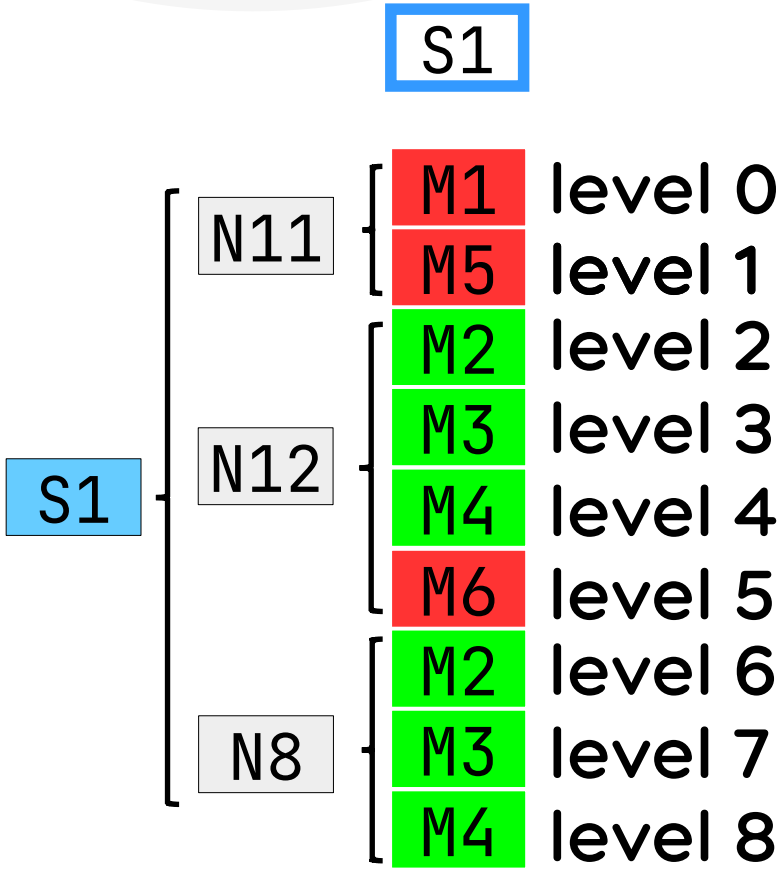
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 2
g(…) = M2 = N12[0]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

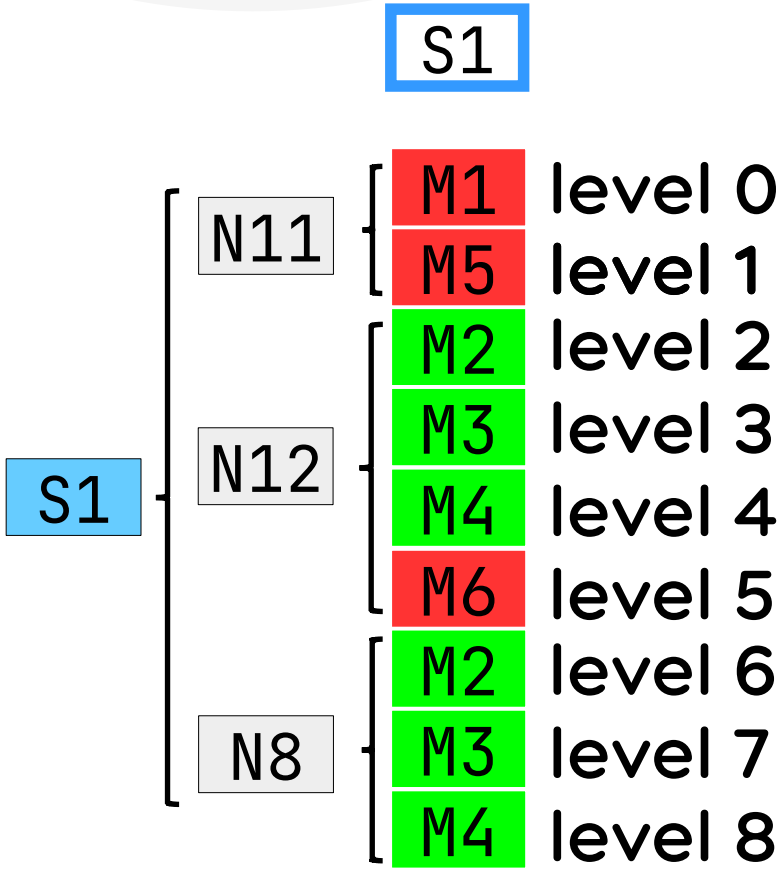
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 3
g(…) = M3 = N12[1]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

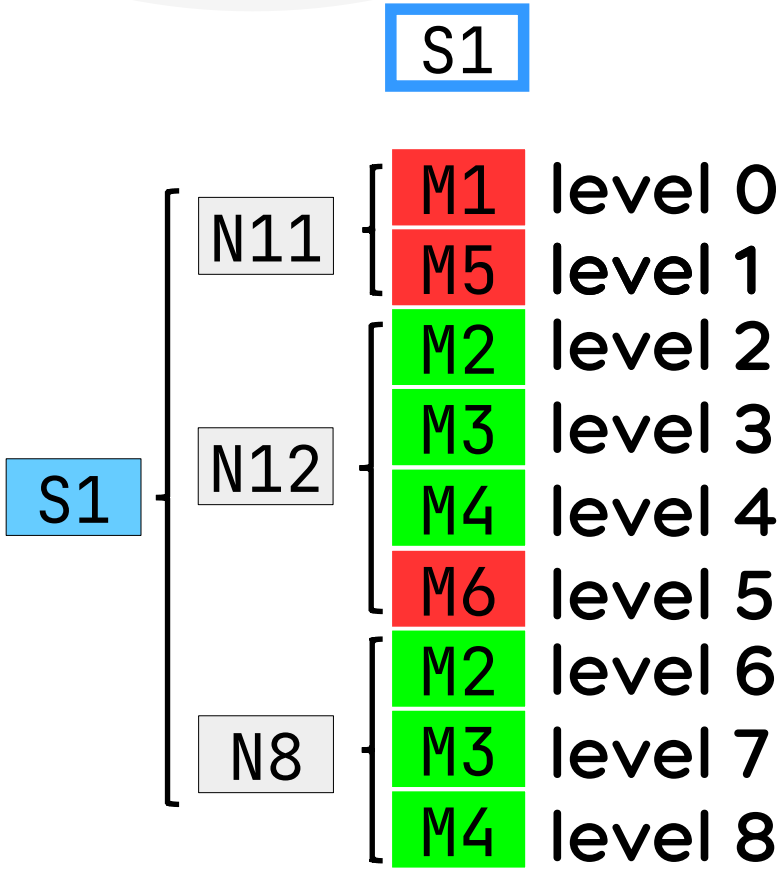
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 4
g(…) = M4 = N12[2]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

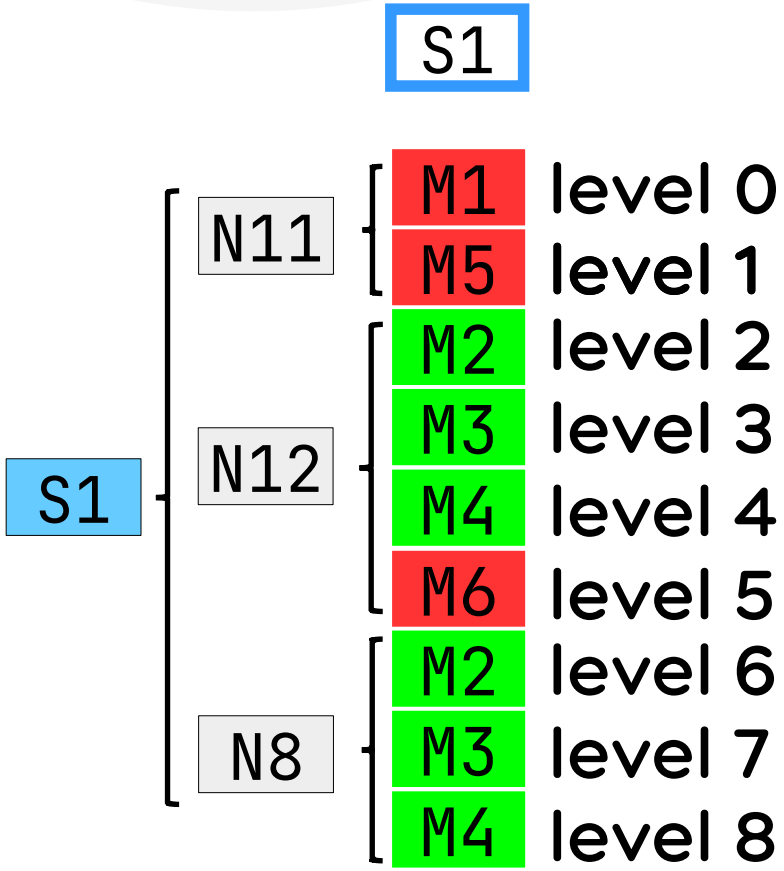
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 5
g(…) = M6 = N12[3]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

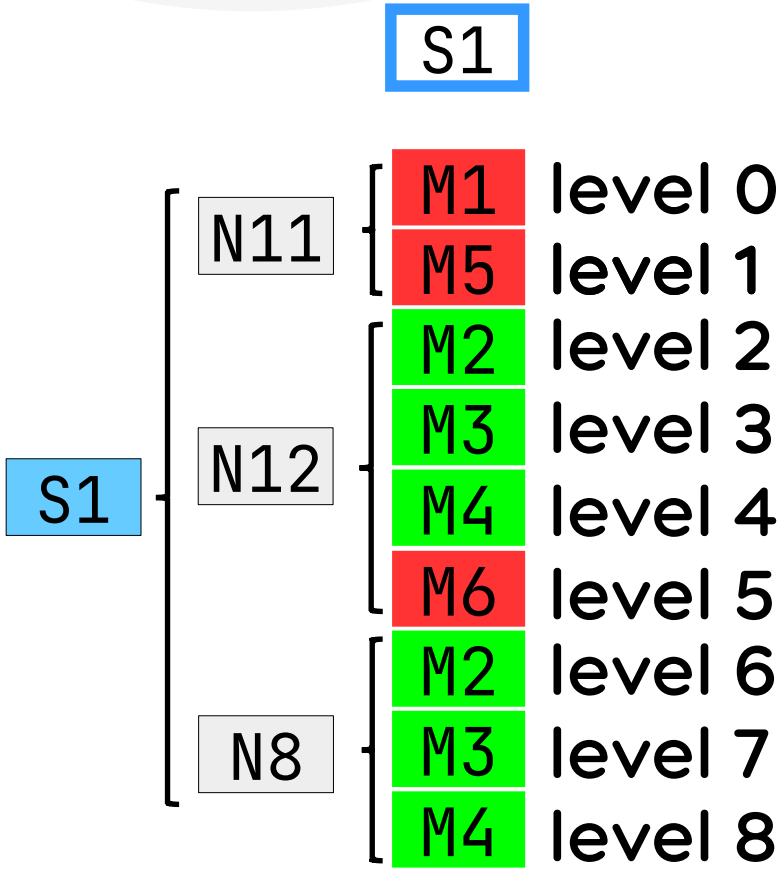
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 6
g(…) = M2 = N8[0]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

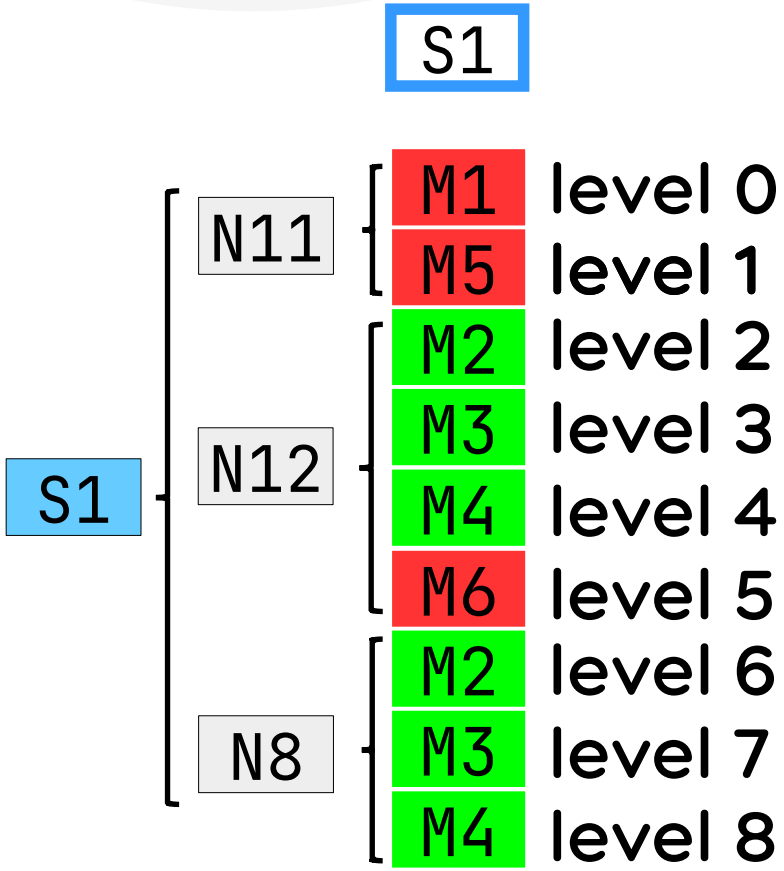
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 7
g(…) = M3 = N8[1]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

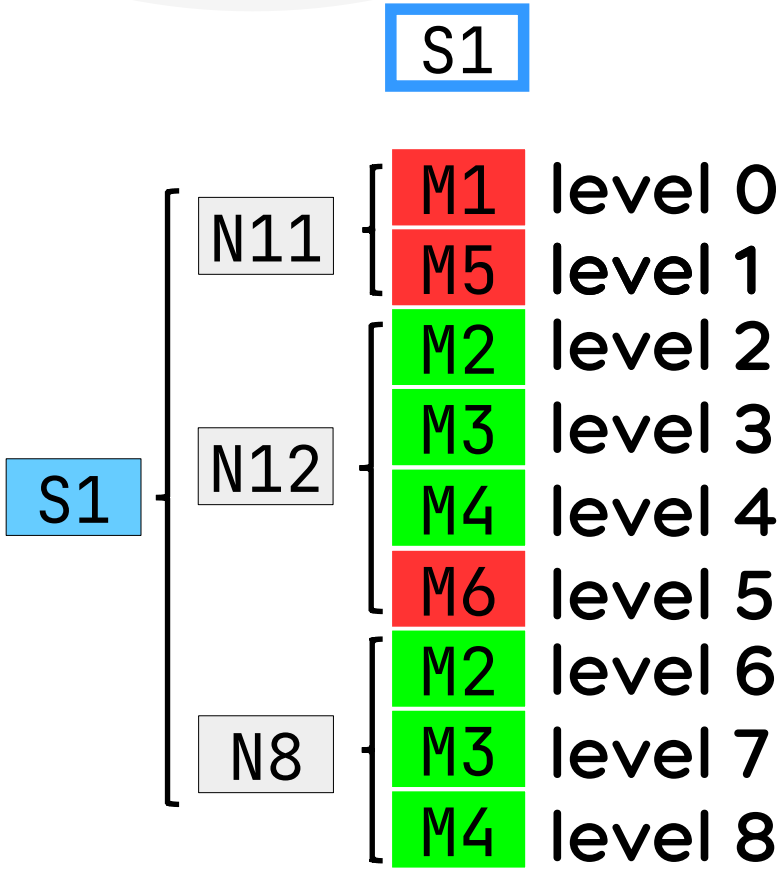
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 8
g(…) = M4 = N8[2]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

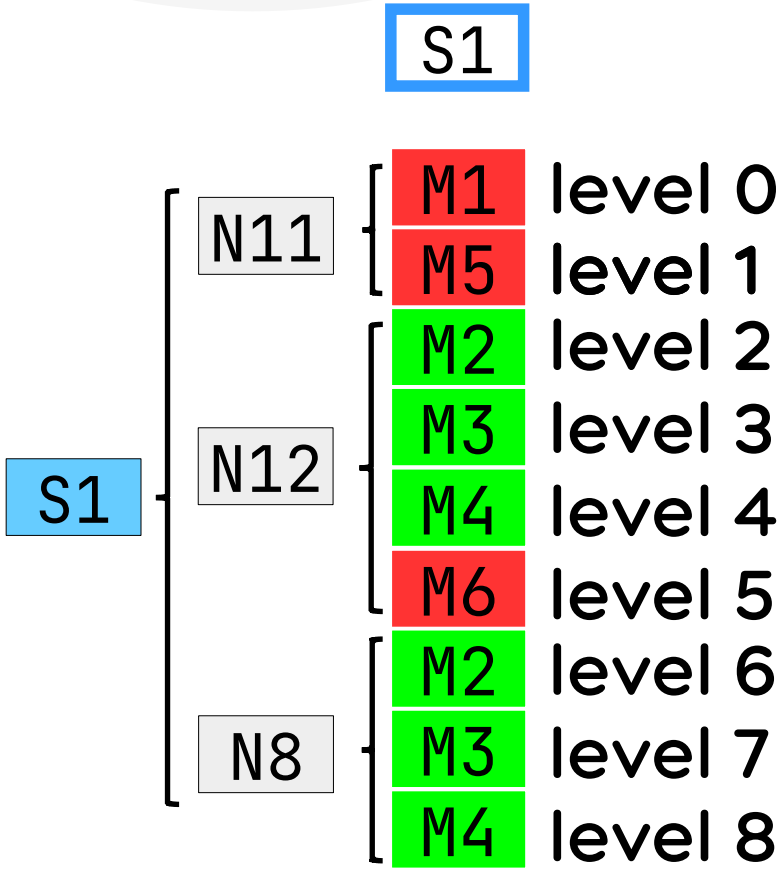
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 8
g(…) = M4 = N8[2]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

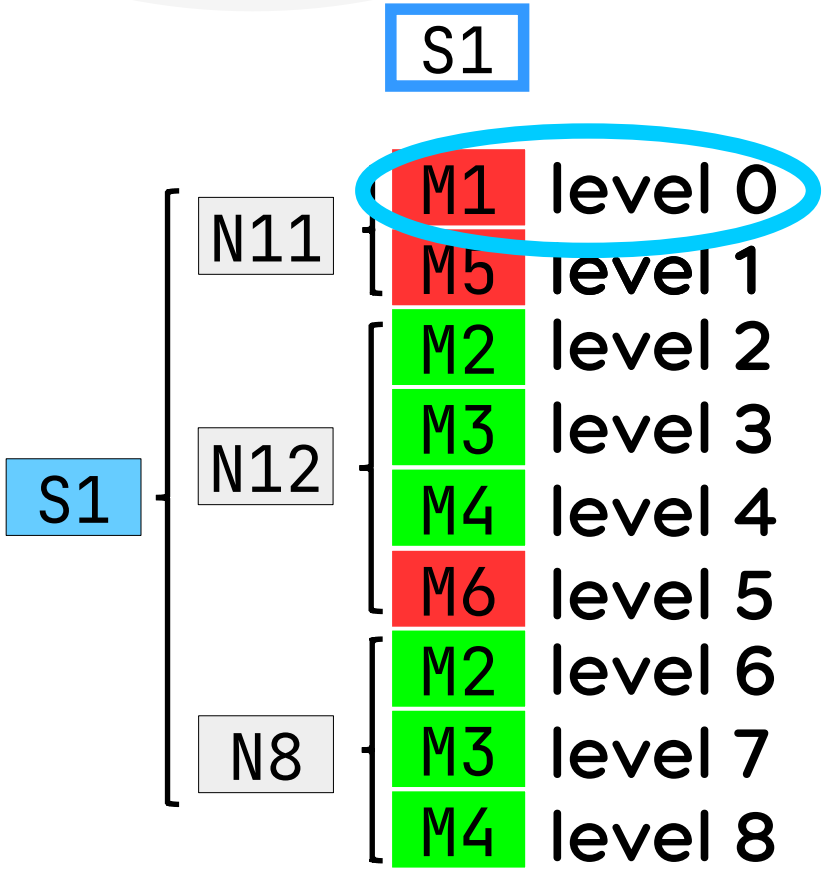
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 8
g(…) = M4 = N8[2]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

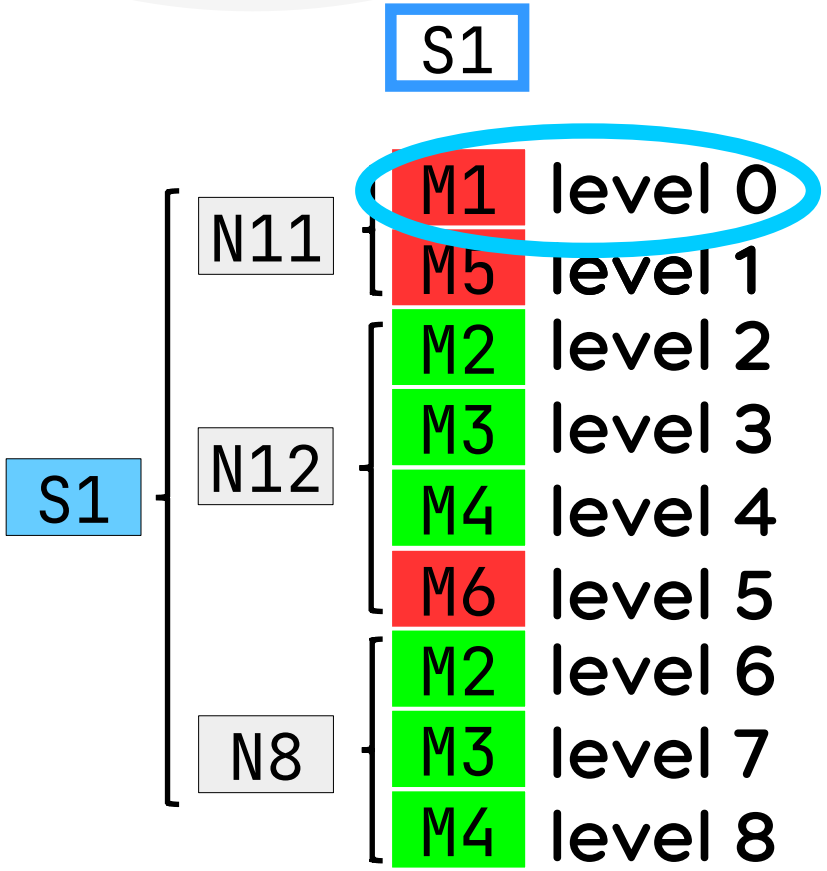
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 0
g(…) = M1 = N11[0]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

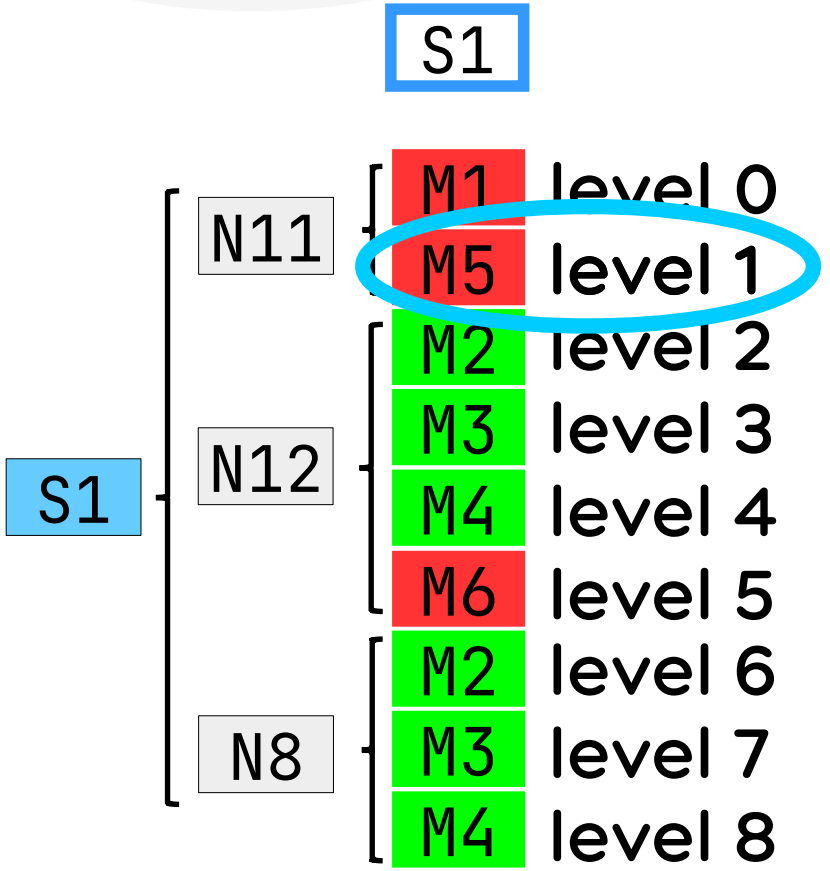
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 0
g(…) = M1 = S1[0][level]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

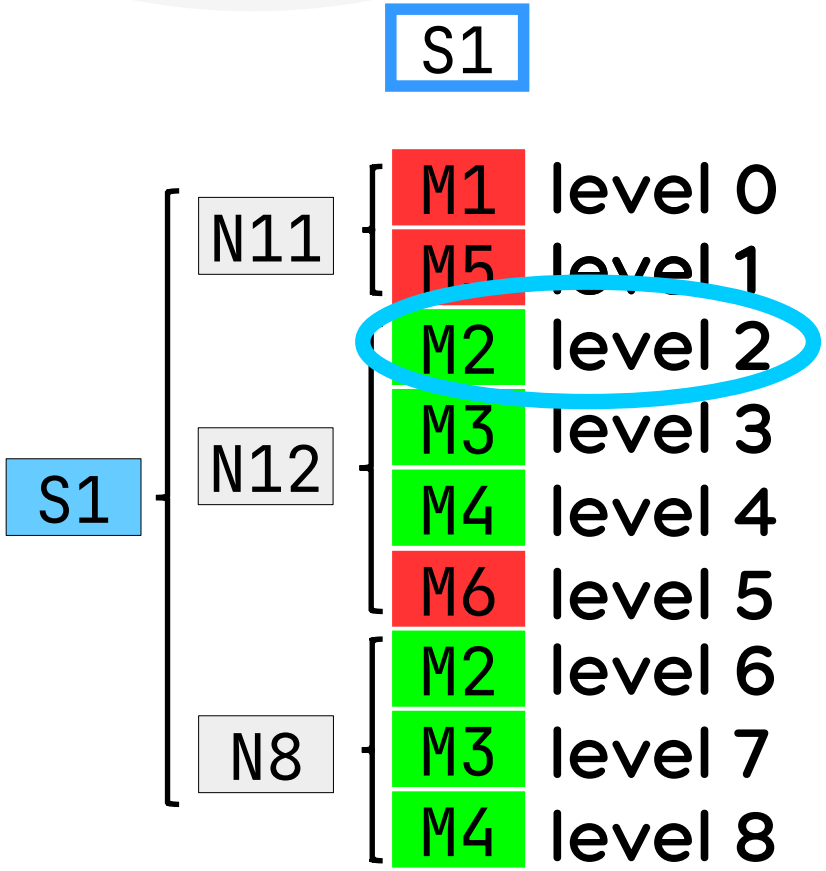
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 1
g(…) = M5 = S1[0][level]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

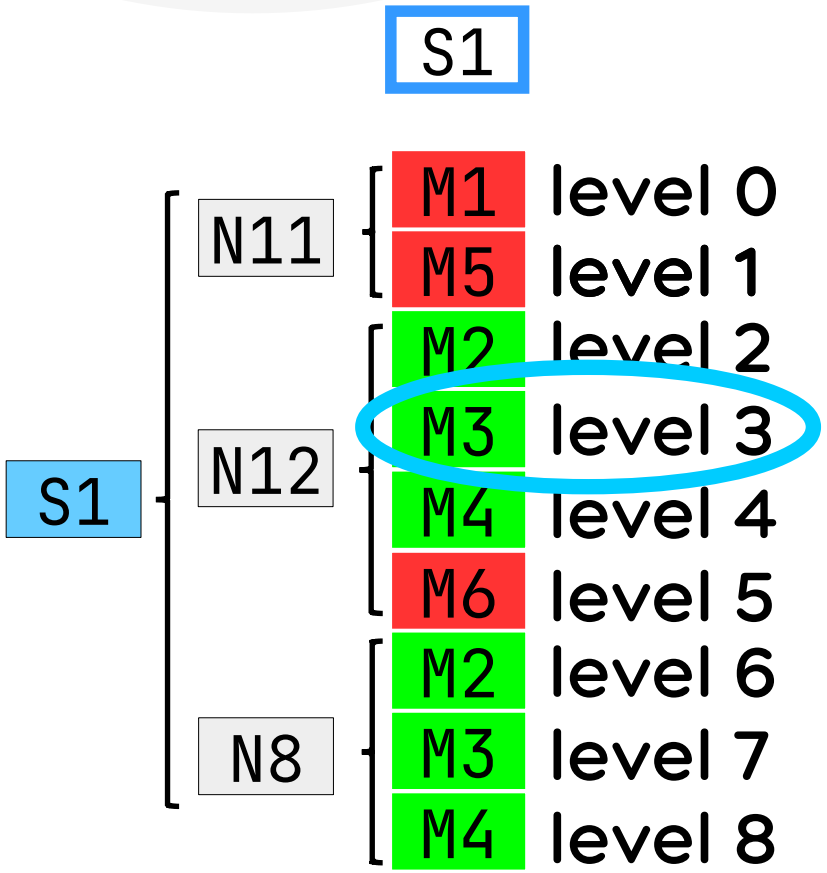
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 2
g(…) = M2 = S1[1][level - 2]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

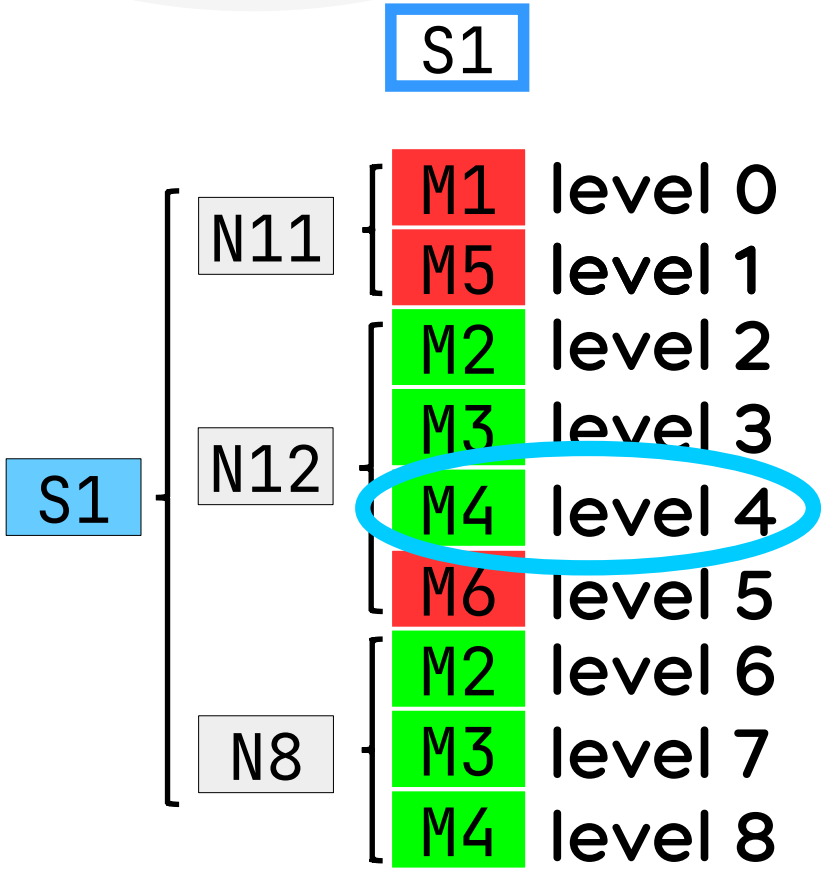
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 3
g(…) = M3 = S1[1][level - 2]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

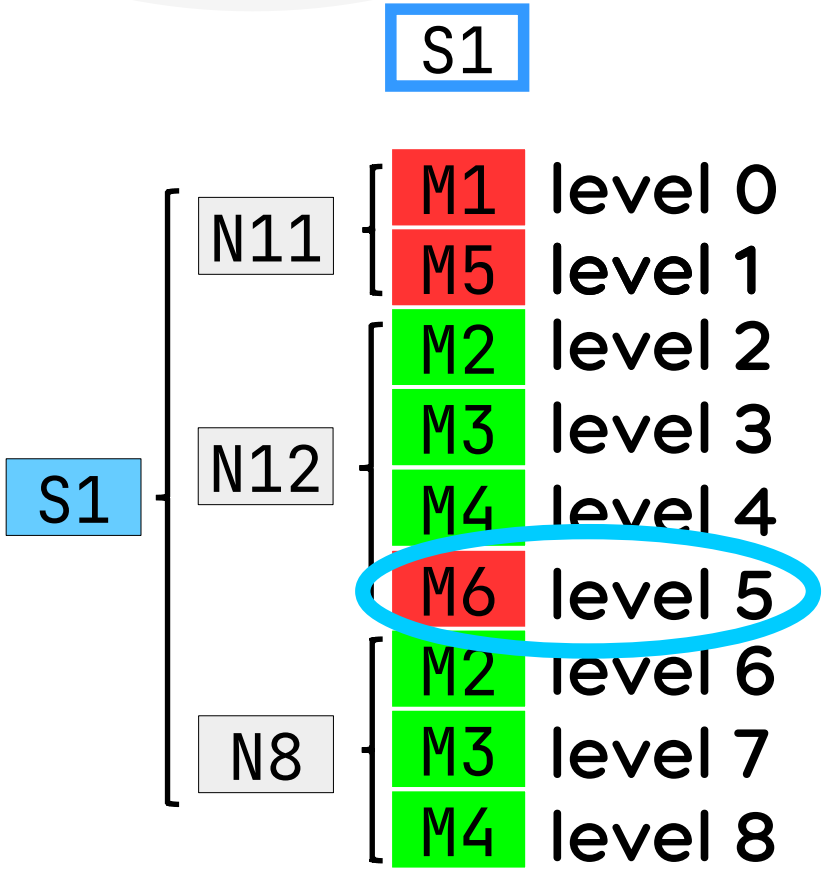
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 4
g(…) = M4 = S1[1][level - 2]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

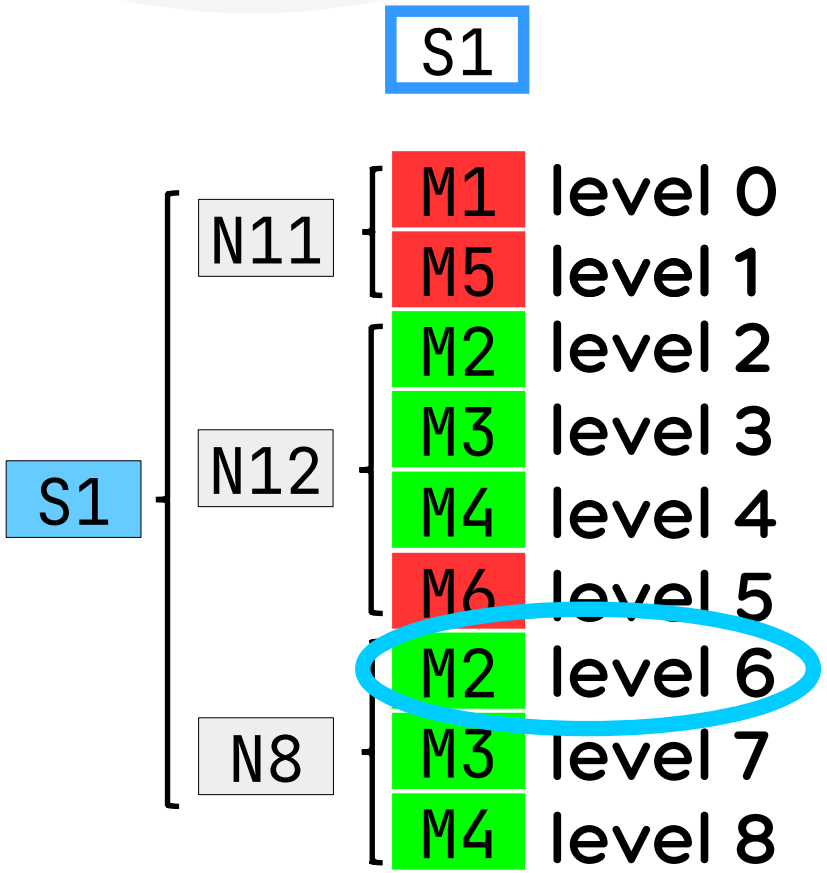
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 5
g(…) = M6 = S1[1][level - 2]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

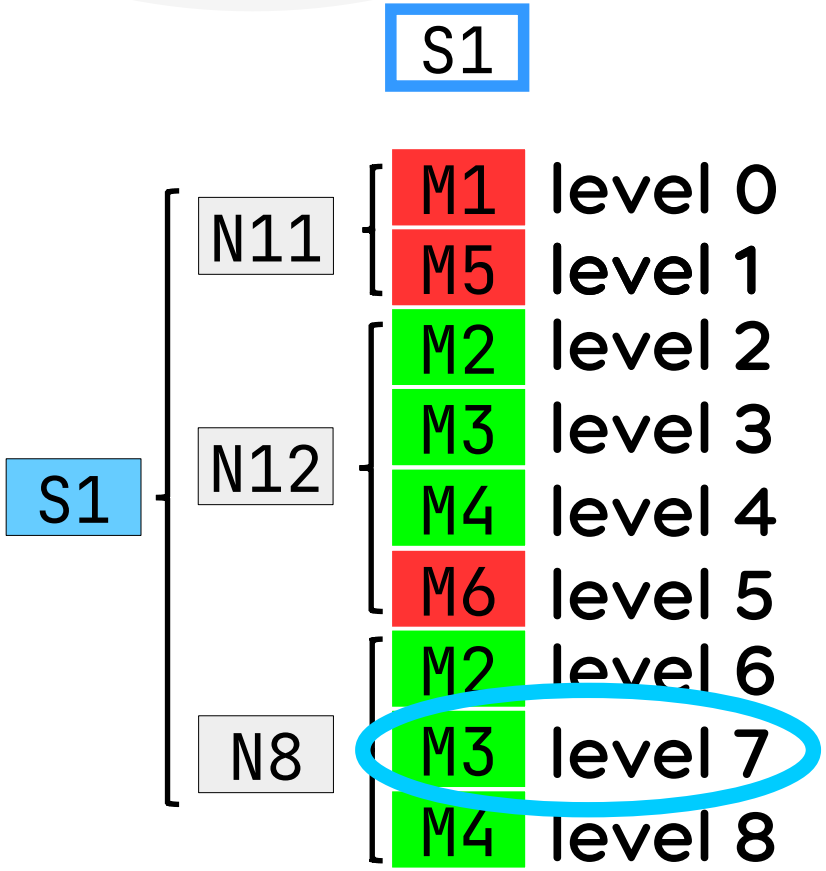
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 6
g(…) = M2 = S1[2][level - 6]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

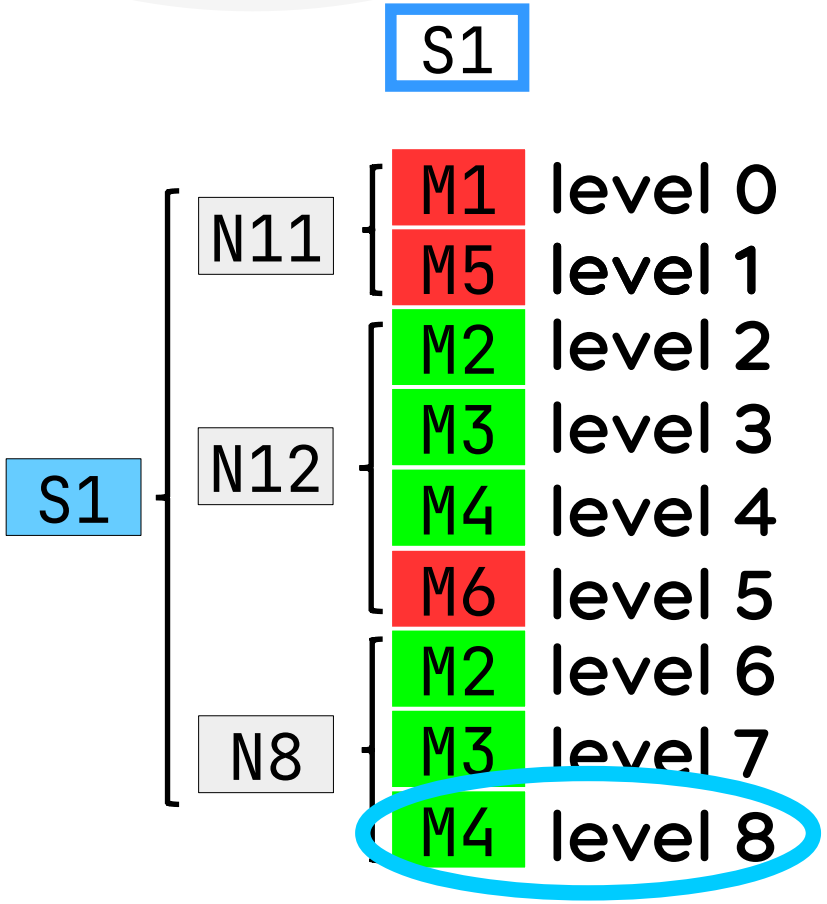
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 7
g(…) = M3 = S1[2][level - 6]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

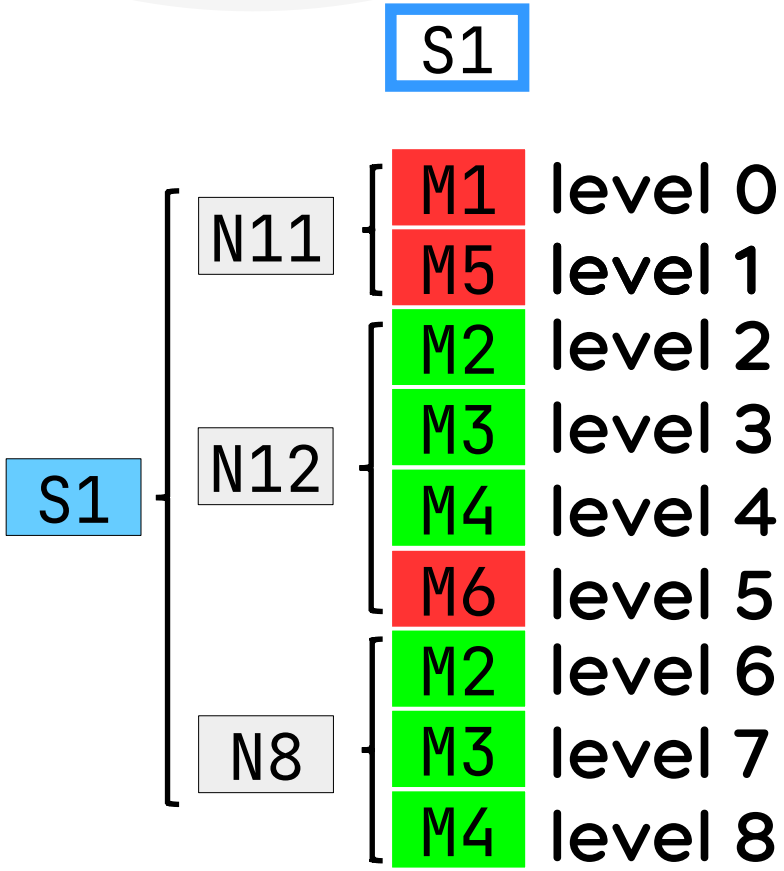
```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 8
g(…) = M4 = S1[2][level - 6]
```



samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 8
g(…) = M4 = S1[2][level - 6]

state.nodeIndex
```

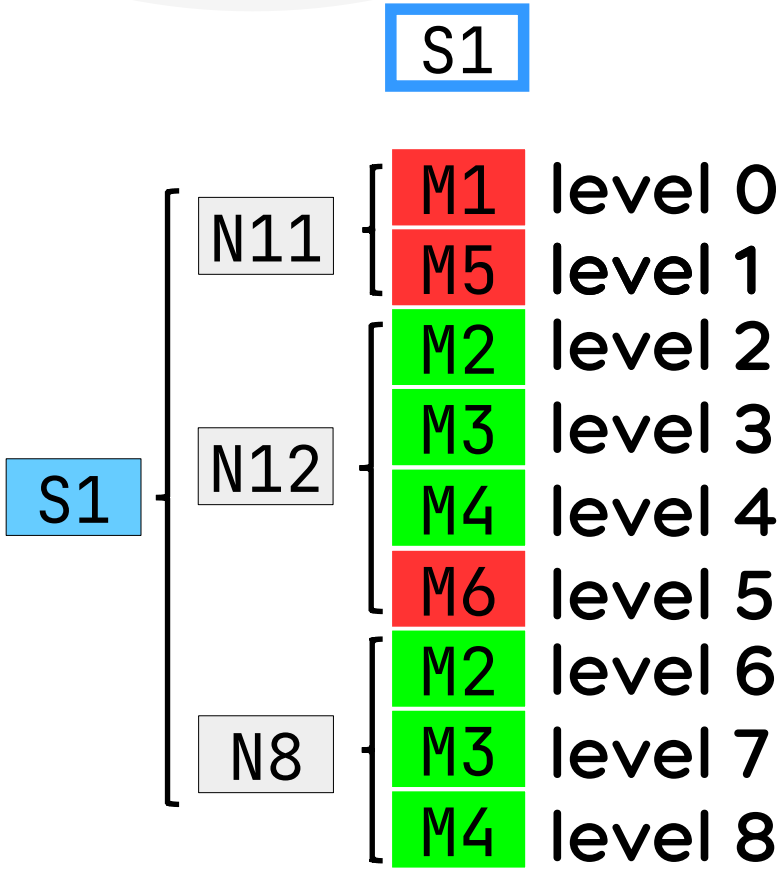


samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

Рисуем

```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 8
g(…) = M4 = S1[2][level - 6]

state.nodeIndex
state.diff
```

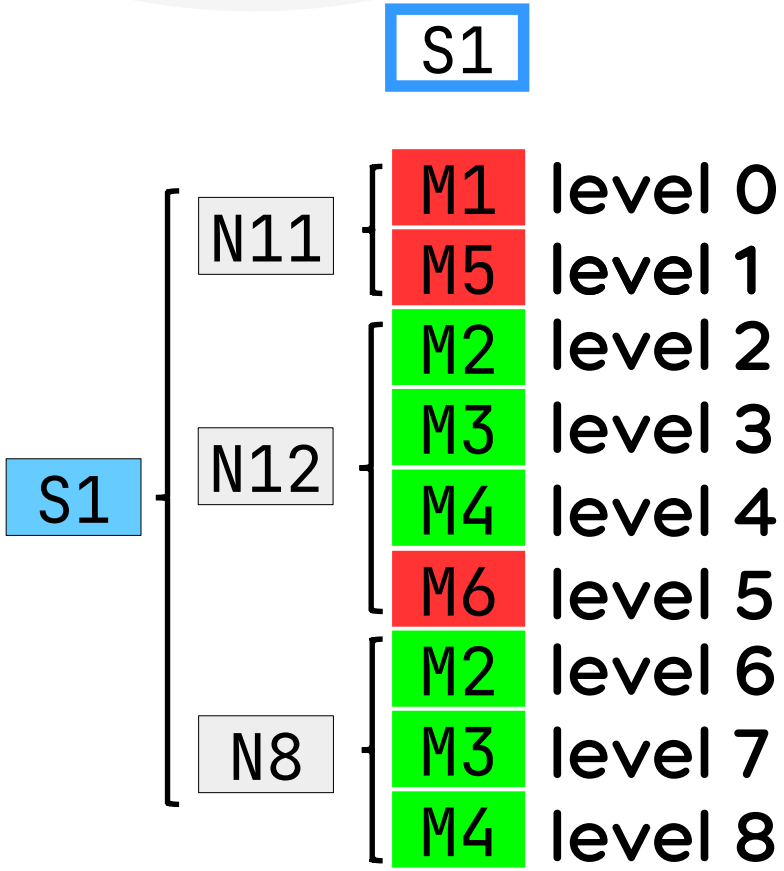
samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...



Рисуем

```
g(state,level,sample) = f(i(…),n(…))

sample = S1
level = 8
g(…) = M4 = S1[2][level - 6]

node = sample[state.nodeIndex]
method = node[level - state.diff]
```

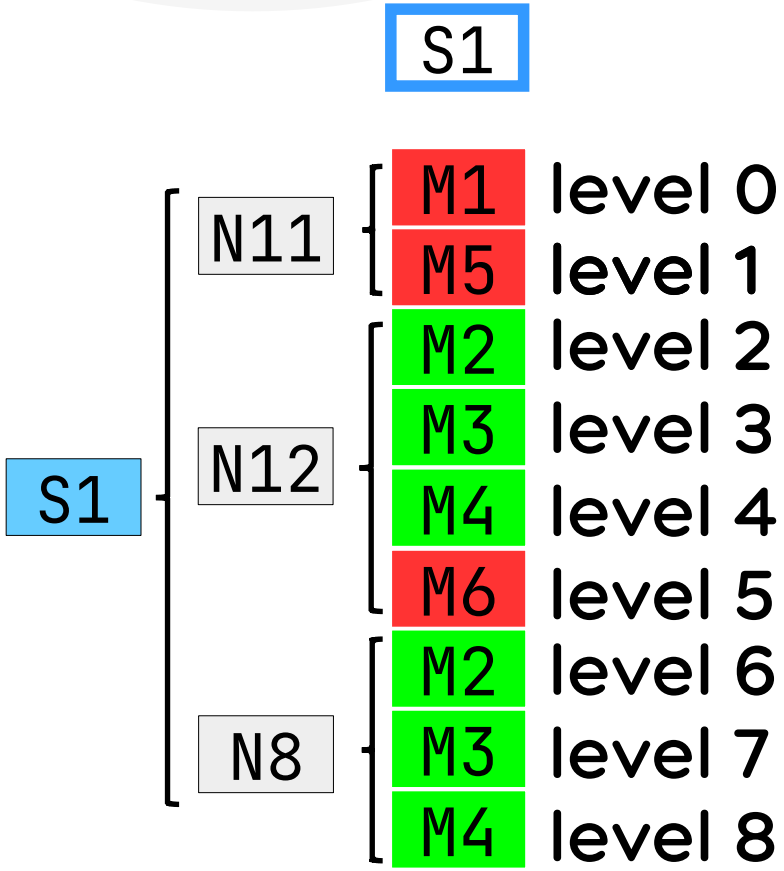
samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...



Рисуем

```
g(state,level,sample) = f(i(…),n(…))

node = sample[state.nodeIndex]
method = node[level - state.diff]

if (level - state.diff ⩾ node.size)
      state.nodeIndex += 1
      state.diff = level
```

samples	...	S1	S2	S3	S4	...
nodes	...	N1	N6	N11	N13	...
offsets	...	0	1	2	3	...
storage	...	M4	M6	M7	M8	...

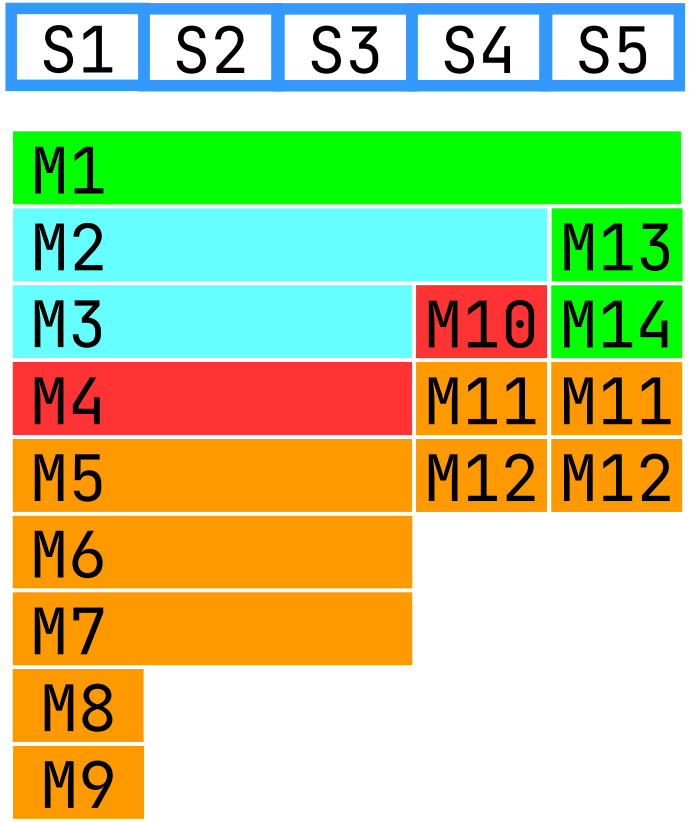


Рисуем

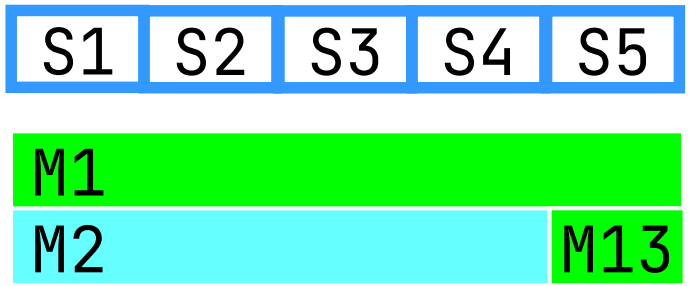
Frame: `{x, width, text, color}`

g(..., sample) = method

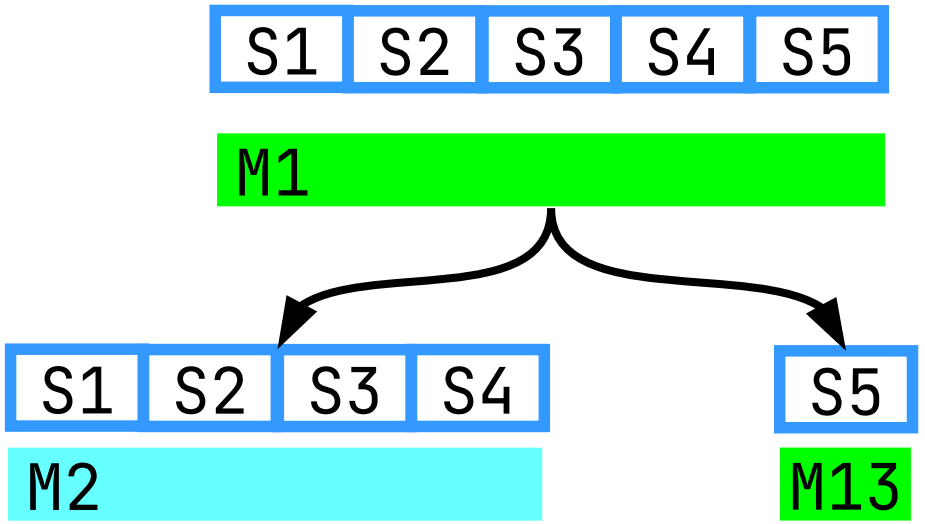
Рисуем



Рисуем

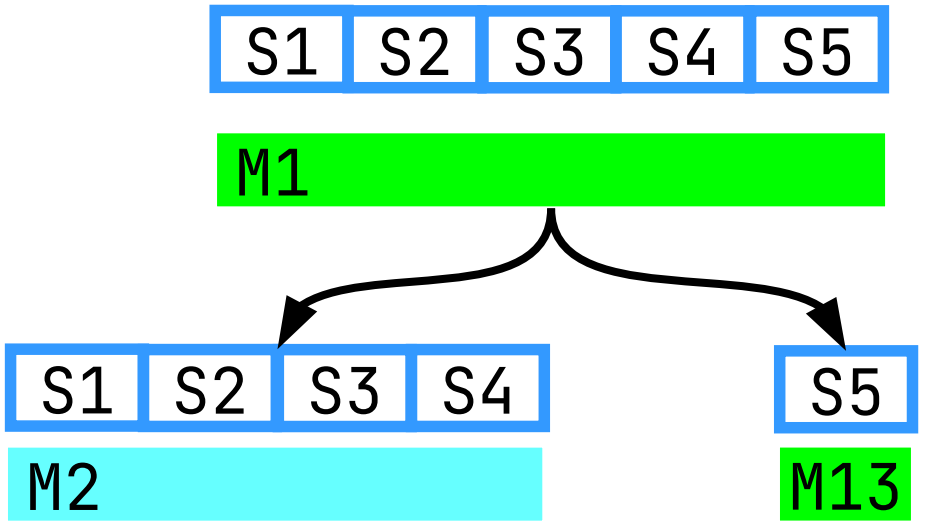


Рисуем



Рисуем

Frame: {x, width, text, color}



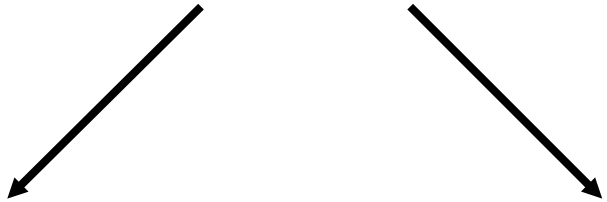
Frame: Map<Method, List<Sample>>

Рисуем

```
Map<Method, List<Sample>>

typeof(Method) = int
typeof(Sample) = int
```

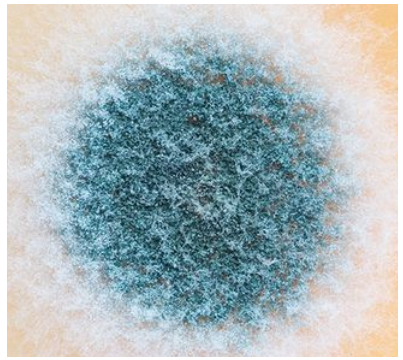
Рисуем

```
Map<Method, List<Sample>>
```

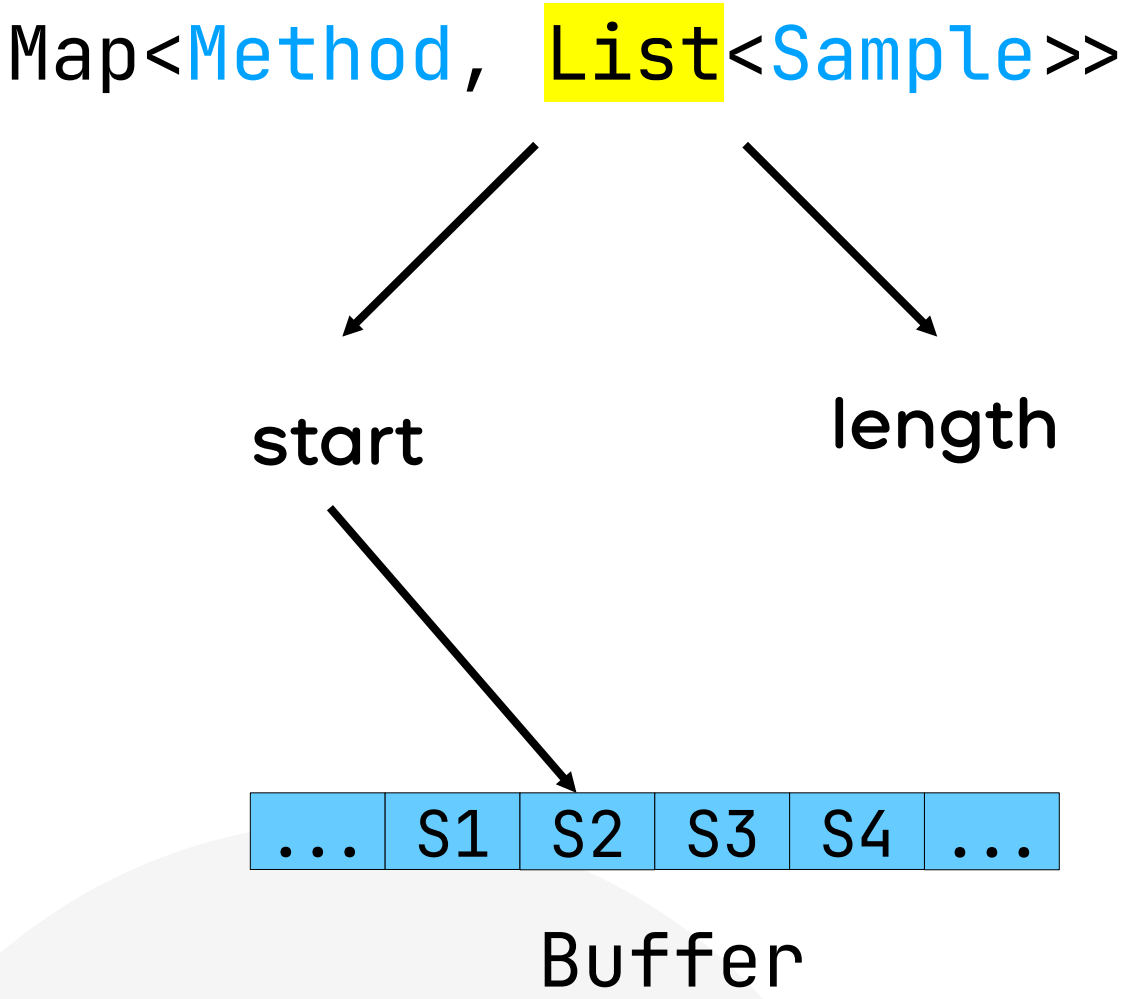


Рисуем

```
Map<Method, List<Sample>>
```

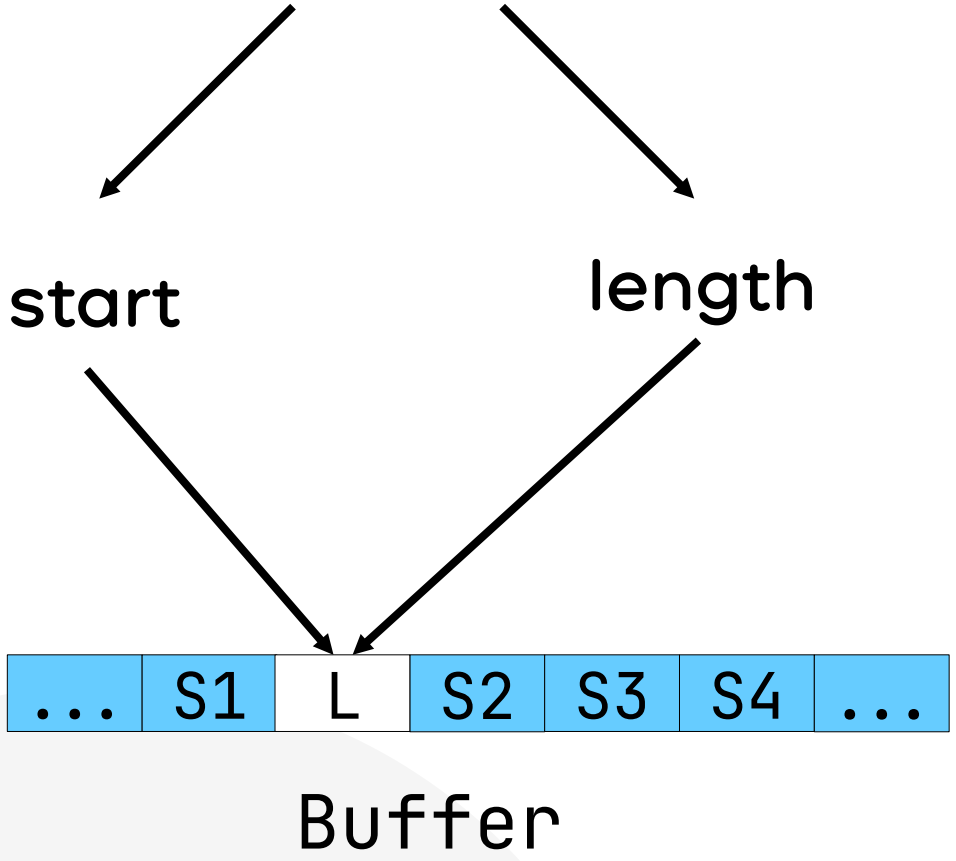


Рисуем



Рисуем

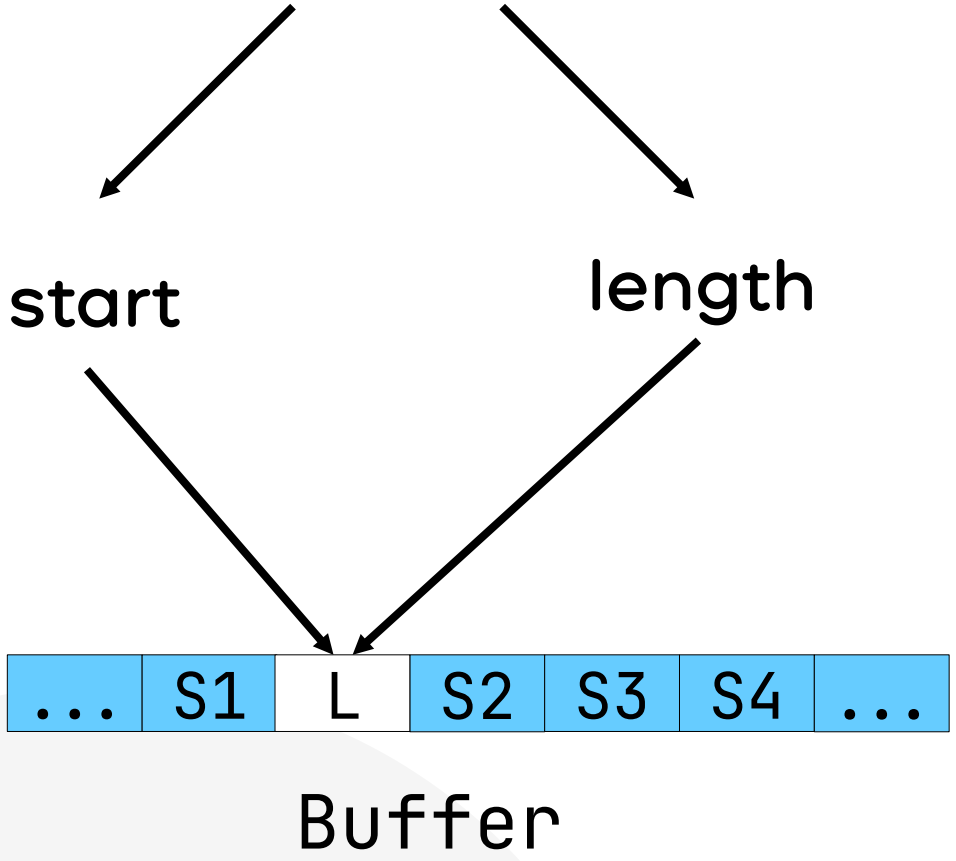
```
Map<Method, List<Sample>>
```



Рисуем

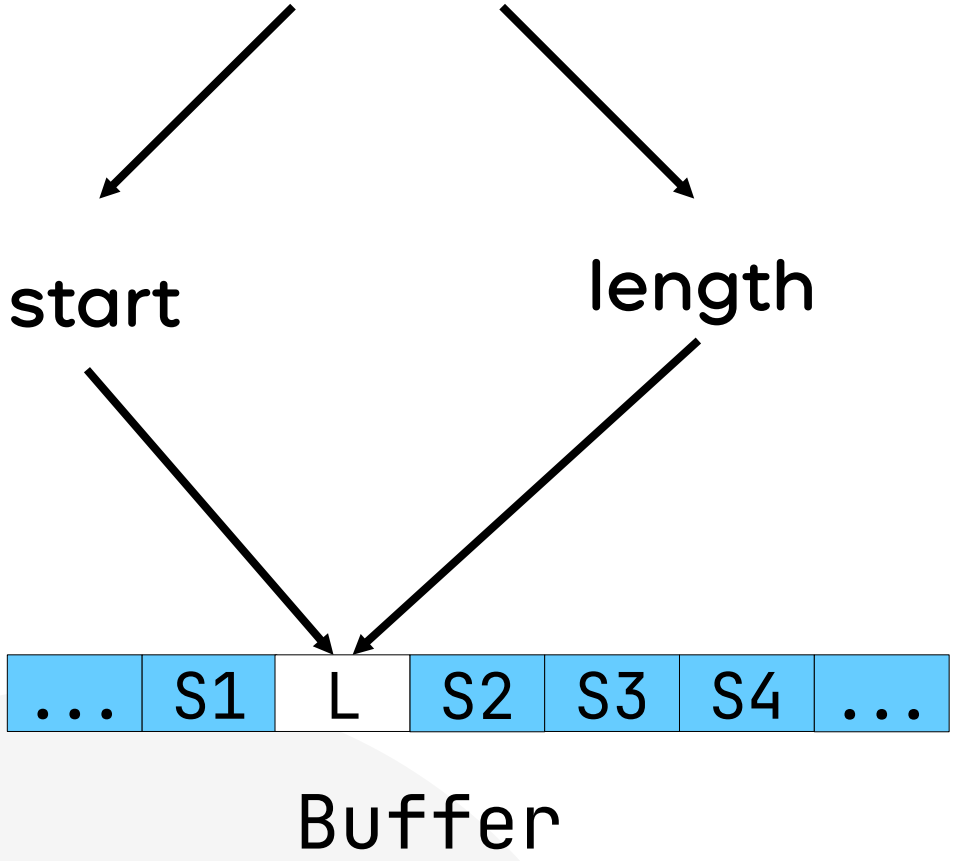
```
Map<Method, List<Sample>>
```

```
start[0] = 0
```



Рисуем

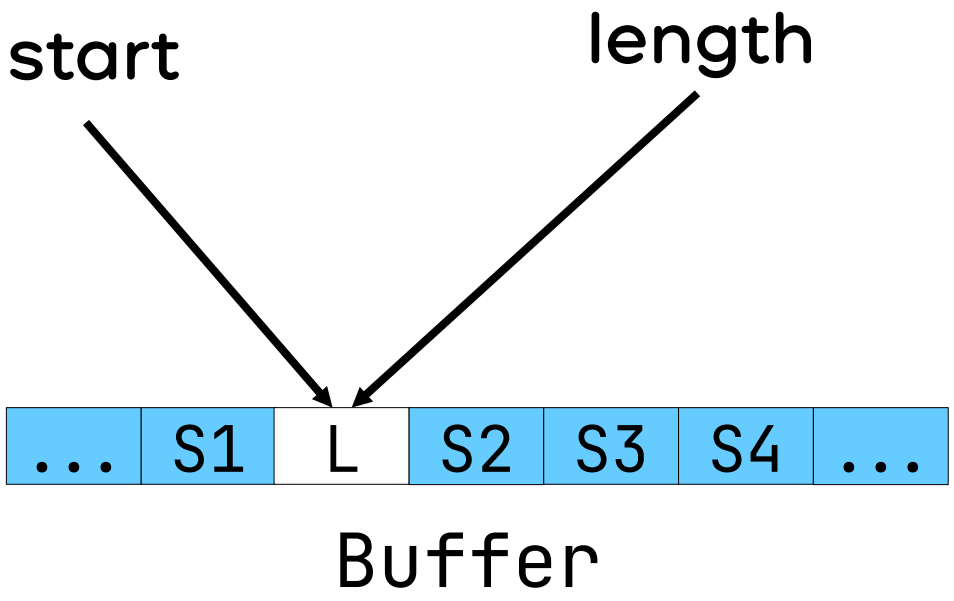
```
Map<Method, List<Sample>>
```




```
start[0] = 0

start[k] = start[k-1] +
    length[k-1] + 1
```

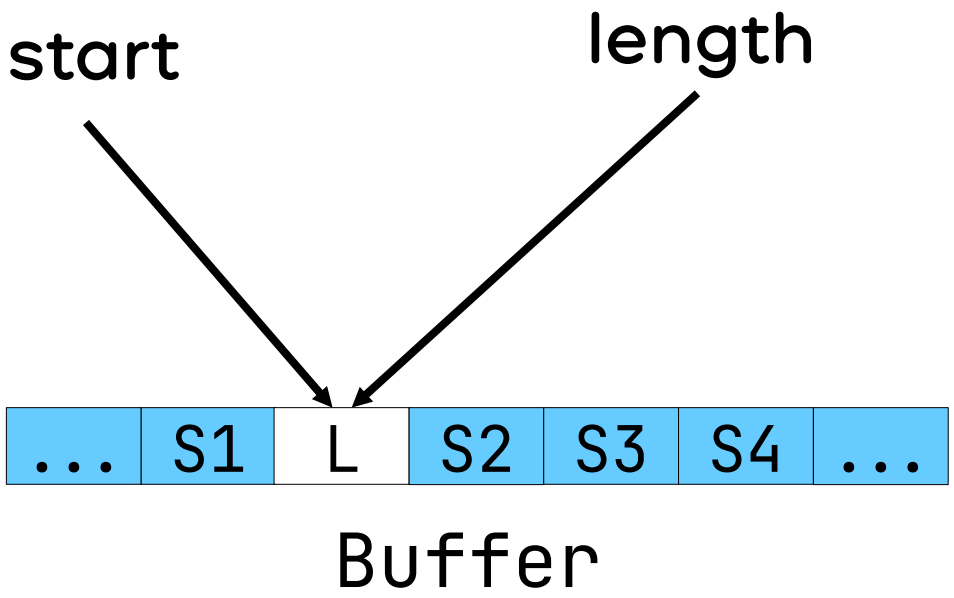
Рисуем

```
Map<Method, List<Sample>>
=
Map<Method, Length>
+
Map<Method, Start>
+
Buffer
```



Рисуем

```
Map<Method, List<Sample>>
=
Map<Method, Length>
+
Map<Method, Start>
+
Buffer
```



Рисуем

```
Map<Method, Length> - ???
```

```
typeof(Method) = int
typeof(Length) = int
```

Рисуем

```
Map<Method, Length> - ???
```

```
typeof(Method) = int 0..100_000
typeof(Length) = int
```

Рисуем

```
Map<Method, Length> - int[100_001]
```

```
typeof(Method) = int 0..100_000
typeof(Size) = int
```

Рисуем

```
Map<Method, Length> - int[100_001]
```

упрлс

```
typeof(Method) = int 0..100_000
typeof(Length) = int
```

Рисуем

```
Map<Method, Length> - int[100_001]
```

Рисуем

```
Map<Method, Length> - int[100_001]
get - O(1)
put - O(1)
```

Рисуем

```
Map<Method, Length> - int[100_001]
get - O(1)
put - O(1)
inc – put(x, get(x) + 1)
```

Рисуем

```
Map<Method, Length> - int[100_001]
get - O(1)
put - O(1)
inc - O(1)
```

Рисуем

```
Map<Method, Length> - int[100_001]
get - O(1)
put - O(1)
inc - O(1)
iterate - O(sizeOf(Method))
```

Рисуем

```
Map<Method, Length> - int[100_001]
get - O(1)
put - O(1)
inc - O(1)
iterate - O(sizeOf(Method))
```

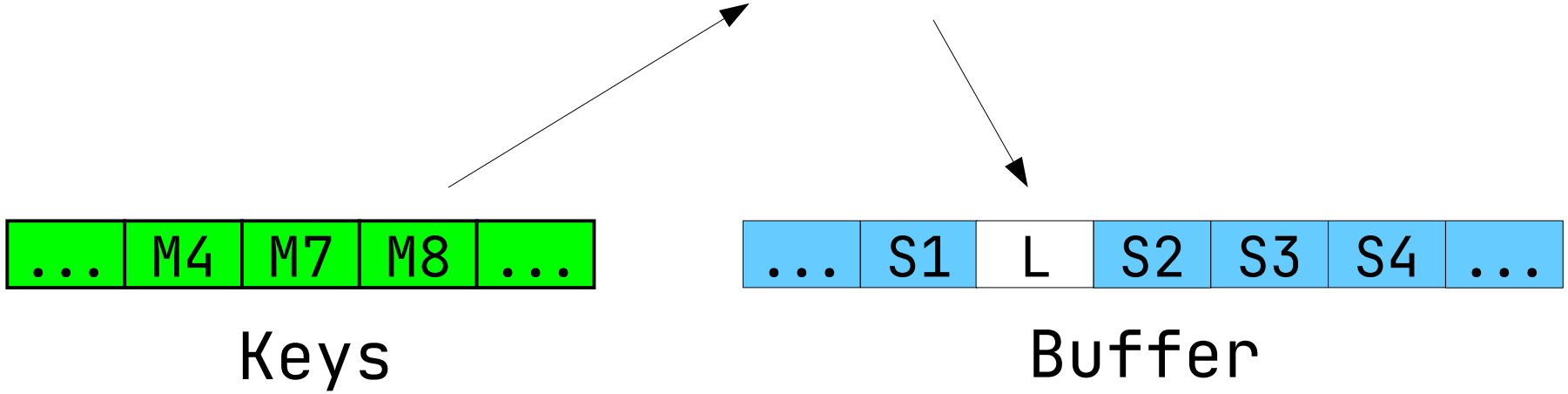
Добавим буфер ключей!

Рисуем

```
Map<Method, Length> - int[100_001]
get - O(1)
put - O(1)
inc - O(1)
iterate - O(entriesCount)

if (inc(x) == 1) keys[entriesCount++] = x
```

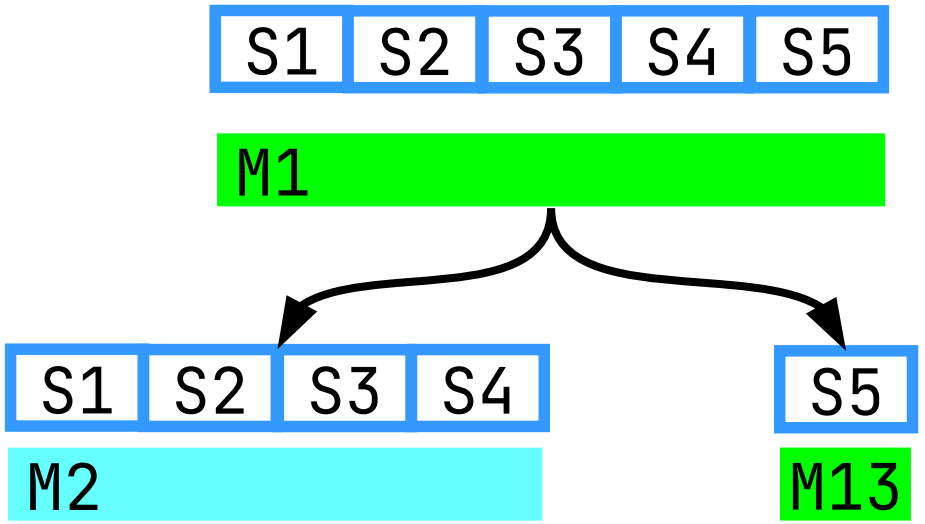
Рисуем

```
Map<Method, List<Sample>> - int[100_001]
```



Рисуем

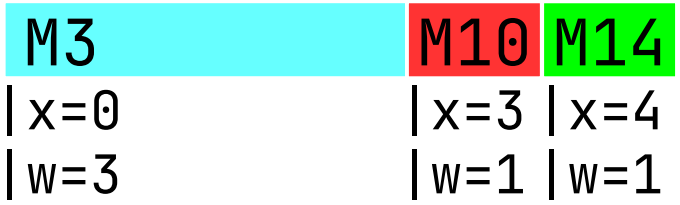
Frame: {x, width, text, color}



Frame: Map<Method, List<Sample>>

Рисуем

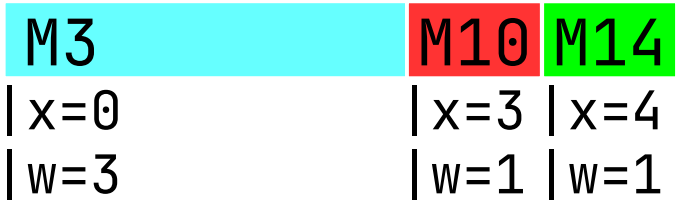
```
Frame: {x, width, text, color}
```



Рисуем

Frame: {x, width, text, color}

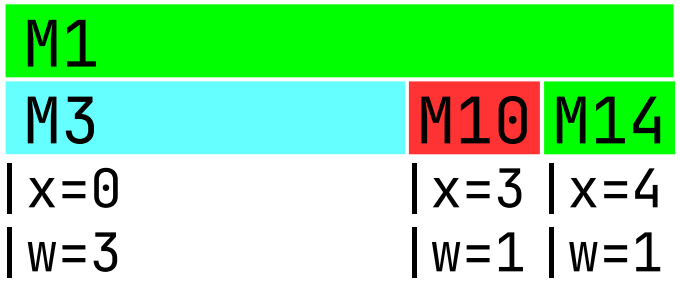
x[i] = x[i-1] + width[i-1]



Рисуем

Frame: {x, width, text, color}

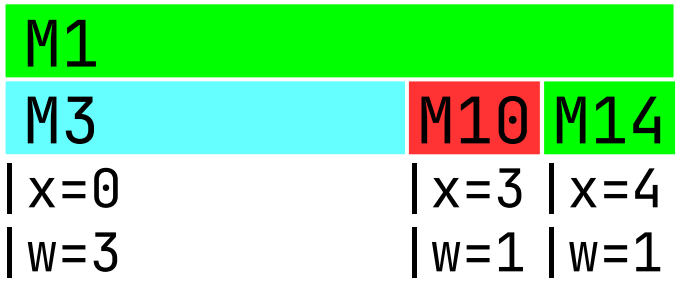
```
x[i] = x[i-1] + width[i-1]
x[0] = parent.x
```



Рисуем

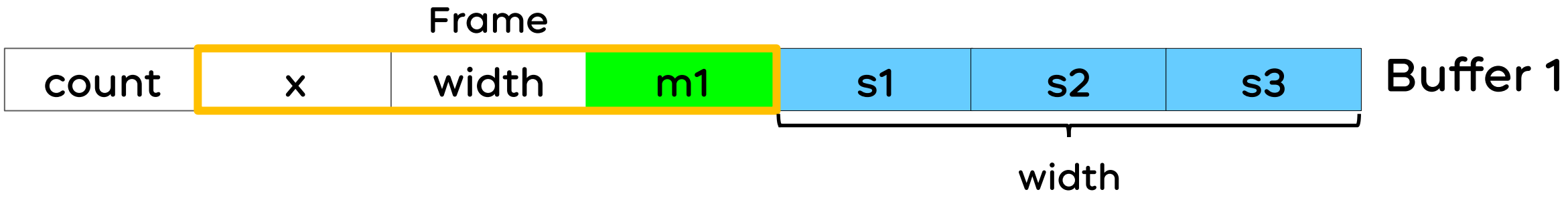
Frame: {x, width, text, color}

```
x[i] = x[i-1] + width[i-1]
x[0] = parent.x

width[i] ⩾ width[i+1]
```



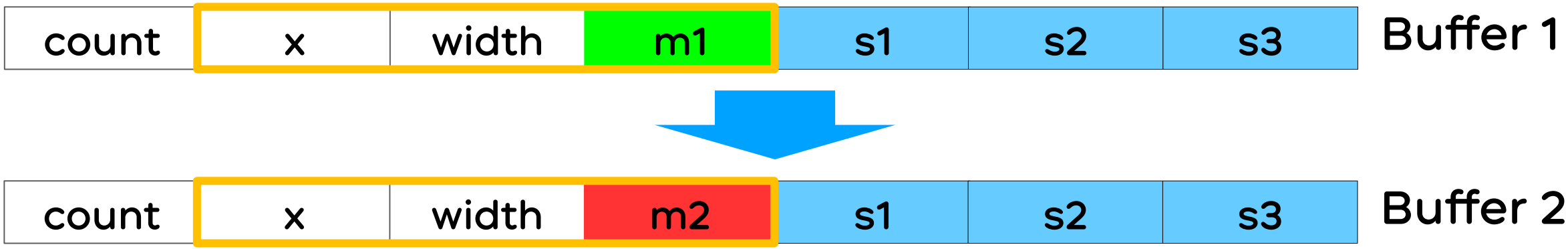
Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]
```



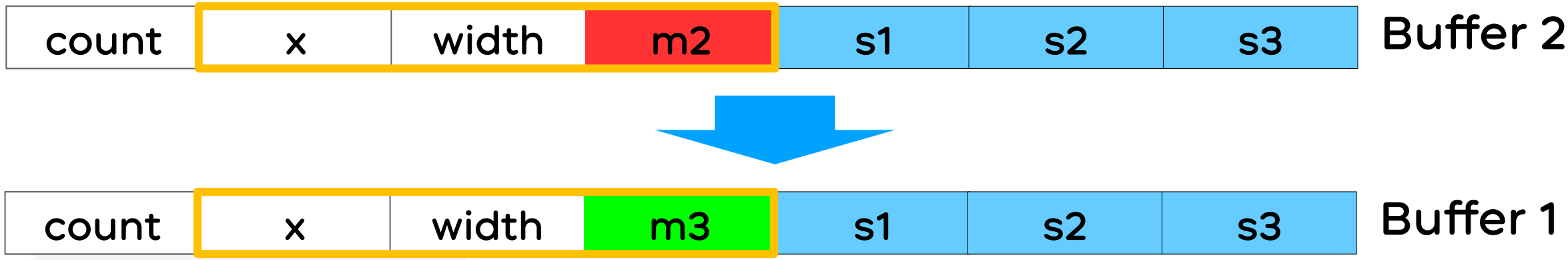
Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]
```



Рисуем

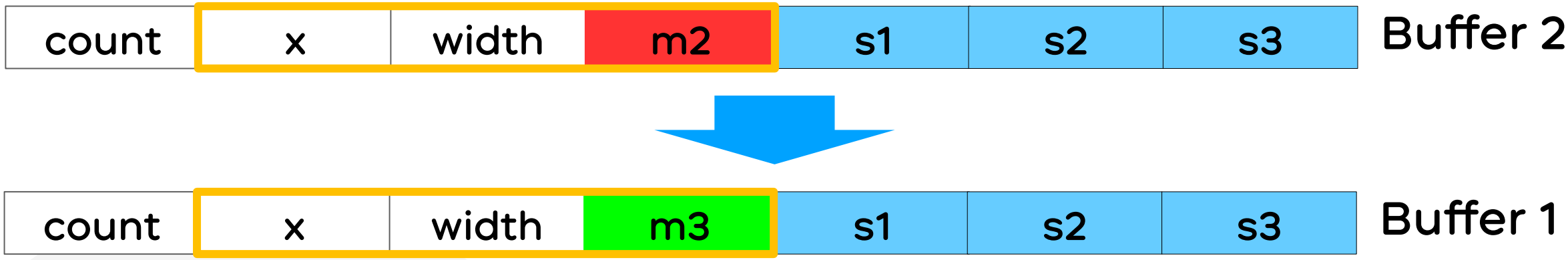
```
Frame: {x, width, text, color}
Level: Frame[]
```



Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]
```

Q: Почему не кольцевой буфер?



Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]

SizeOf(Buffer) - ?
```

Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]

SizeOf(Buffer) - 100 MB
```

Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]

SizeOf(Buffer) - 1 GB
```

Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]

SizeOf(Buffer) - 10 GB
```

Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]

SizeOf(Buffer) - 10 GB ?????????
```

Рисуем

```
Frame: {x, width, text, color}
Level: Frame[]

SizeOf(Buffer) - 10 GB
```

Lazy
Allocation!

Рисуем

CPU (6 млн сэмплов):

< 1 секунды на первый уровень

3 секунды

Рисуем

CPU (6 млн сэмплов):

< 1 секунды на первый уровень

3 секунды

< 5 секунд на первые 20 уровней

15 секунд

Рисуем

CPU (6 млн сэмплов):

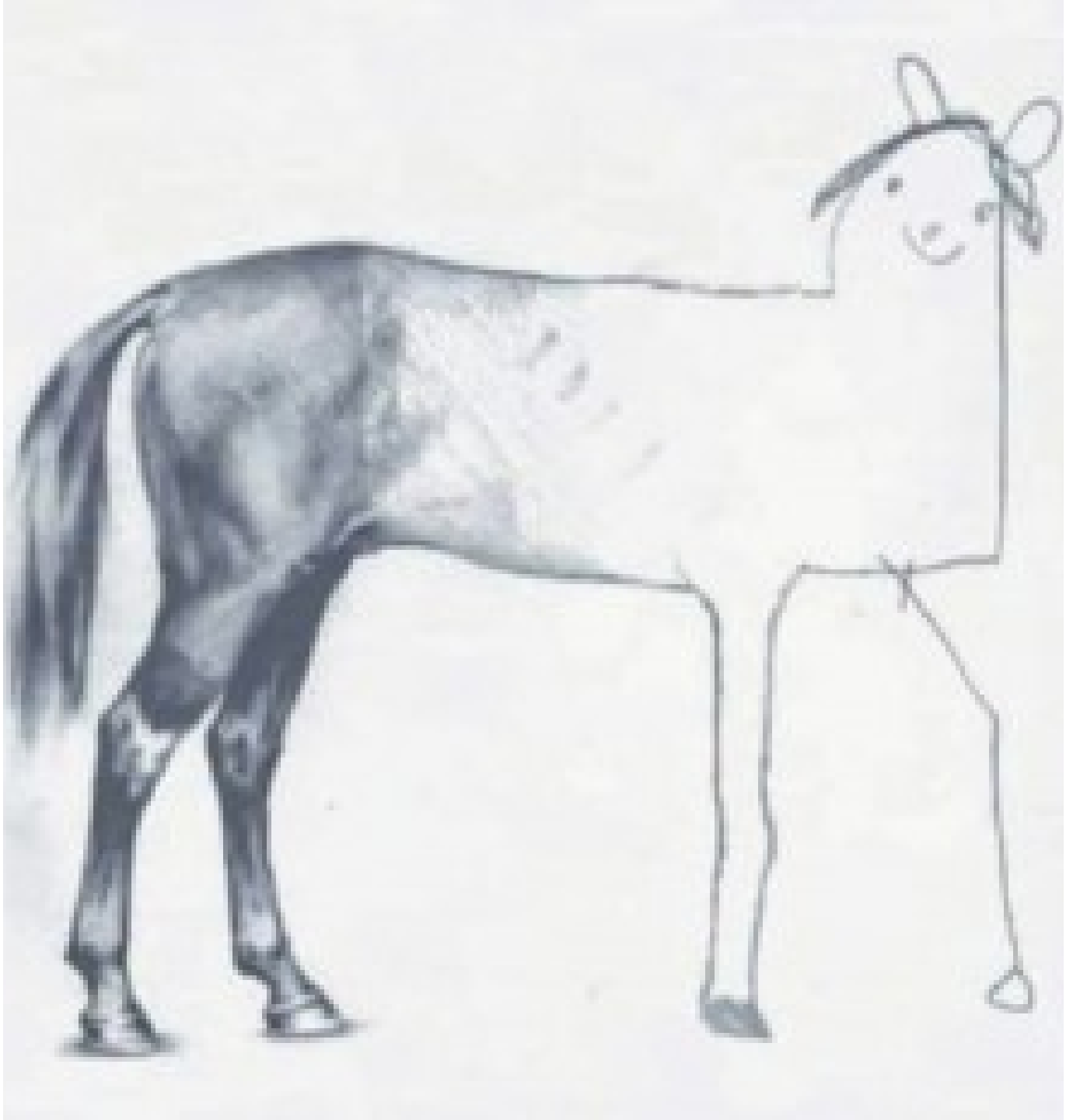
< 1 секунды на первый уровень

3 секунды

< 5 секунд на первые 20 уровней

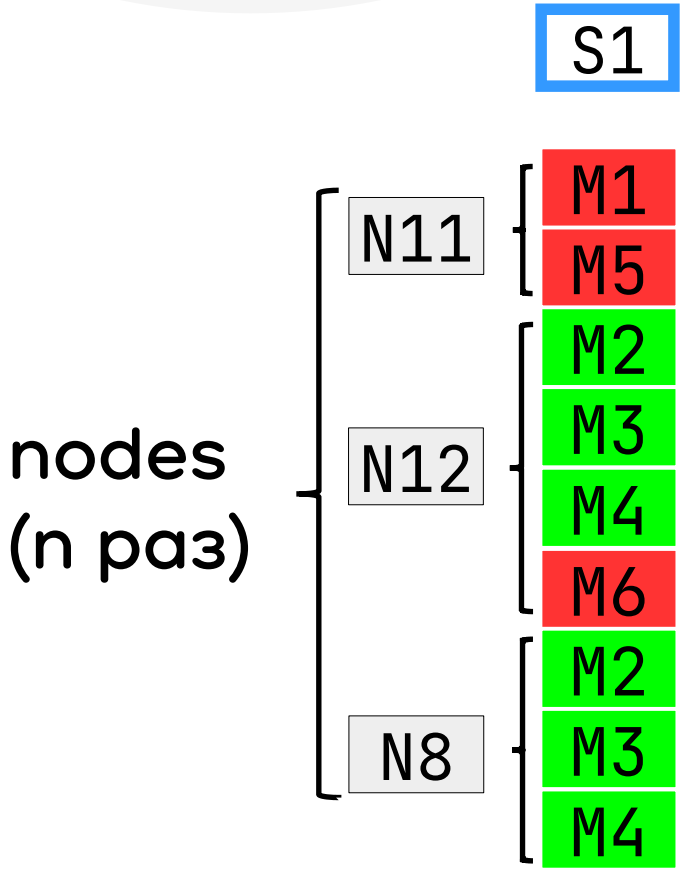
15 секунд

Оптимизация отрисовки



Оптимизируем

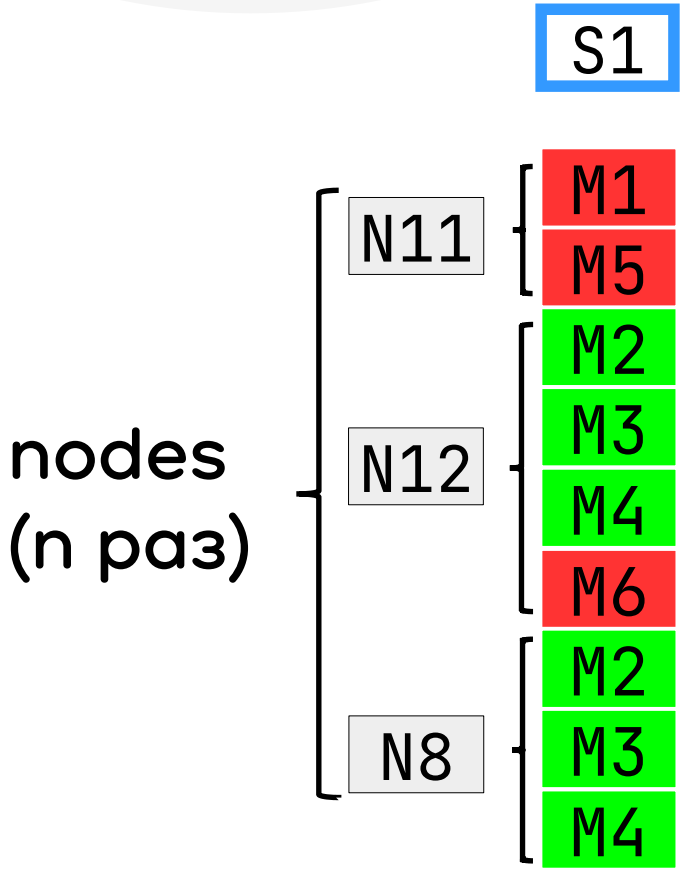
В 90% случаев n = 1



Оптимизируем

В 90% случаев n = 1

Потому что сэмплы часто повторяются!

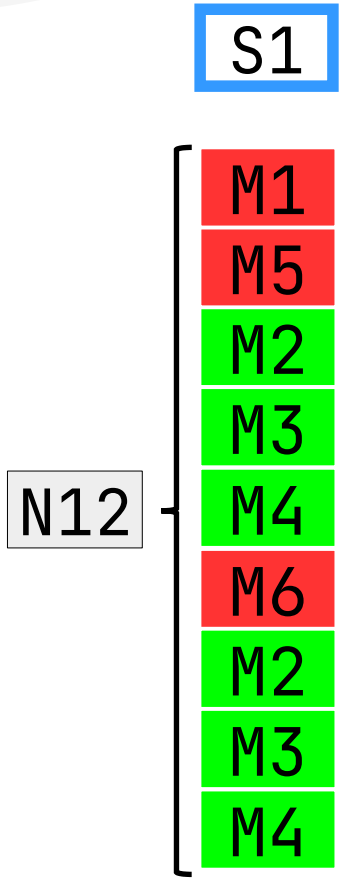


Оптимизируем

В 90% случаев n = 1

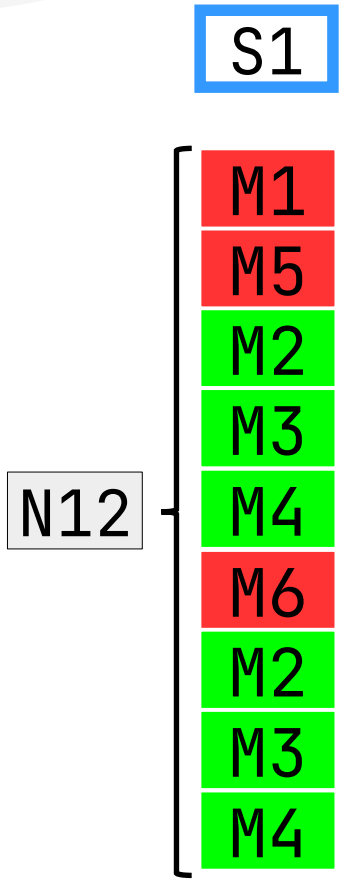
Потому что сэмплы часто повторяются!

И именно те, у которых n = 1!



Оптимизируем

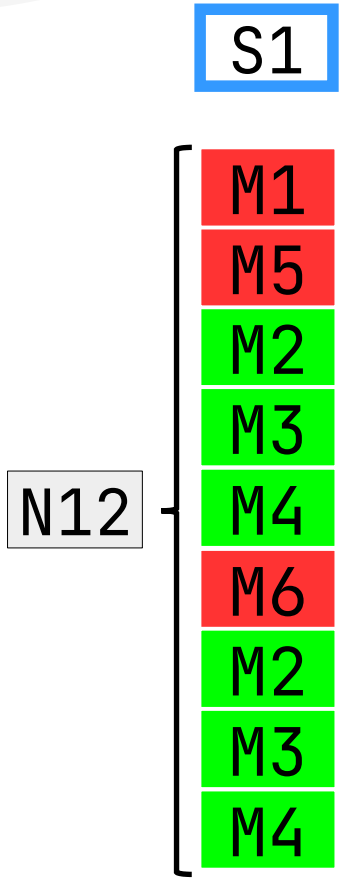
```
Map<Method, Length>
```



Оптимизируем

```
Map<Method, Length>
```

```
Map<Node, Count>
```

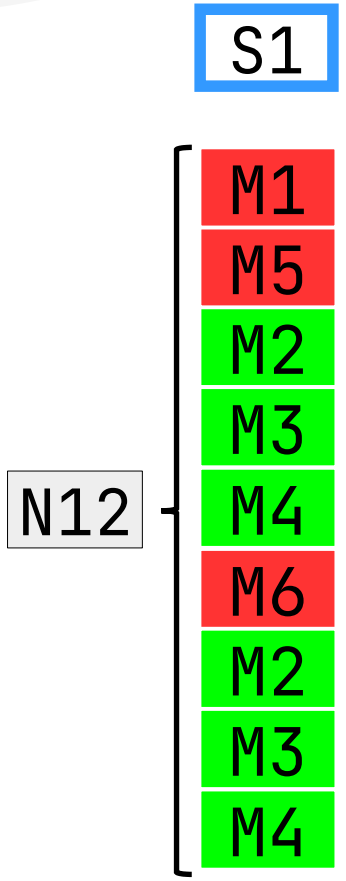


Оптимизируем

```
Map<Method, Length>
```

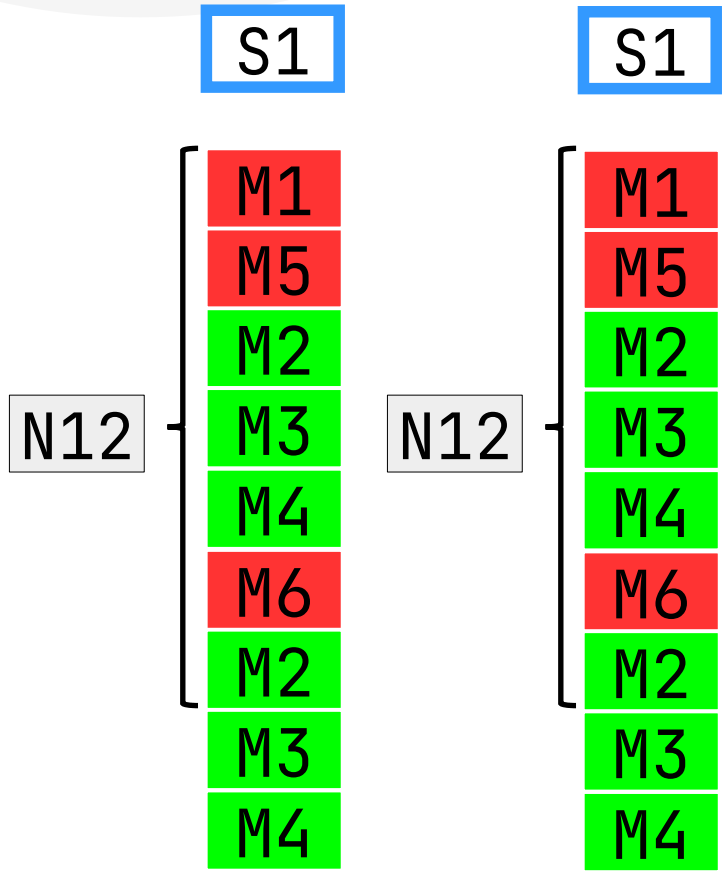
```
Map<Node, Count>
```

6 млн сэмплов  450 000



Оптимизируем ещё

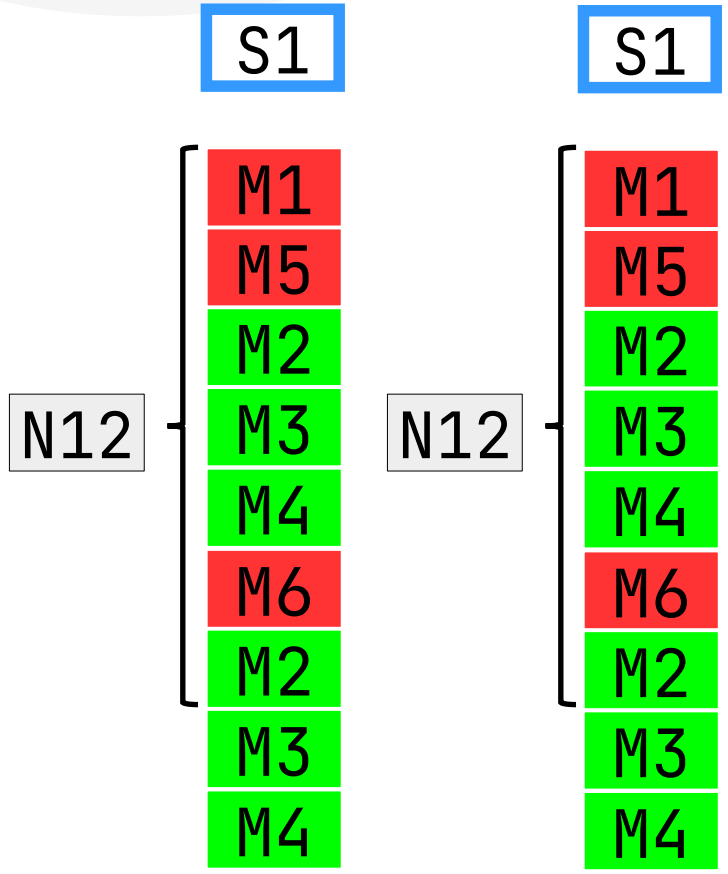
```
Map<Method, List<Sample>>
```



Оптимизируем ещё

```
Map<Method, List<Sample>>
```

```
Map<Node, List<Sample>>
```

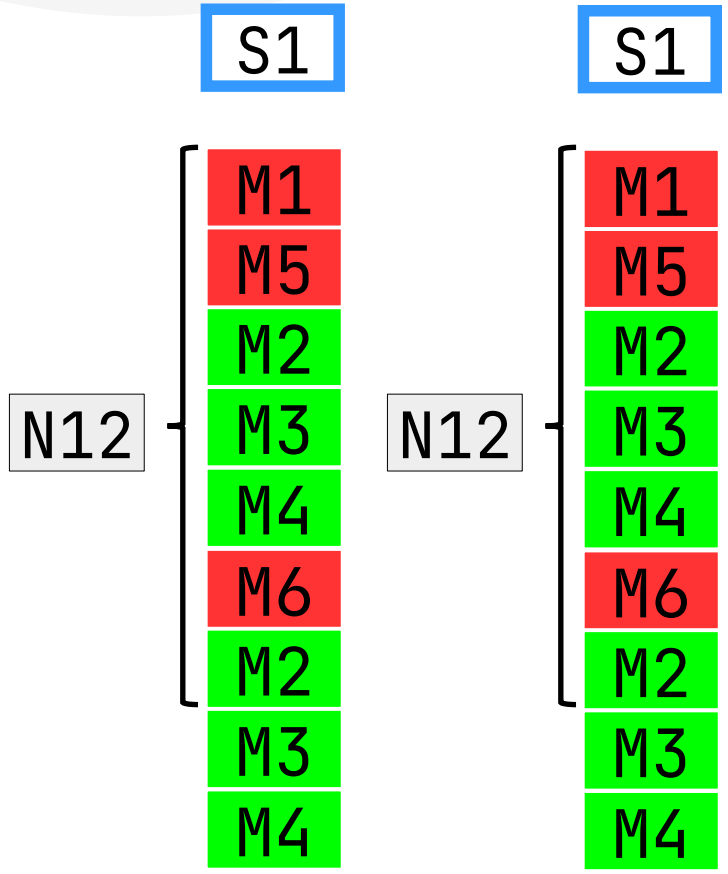


Оптимизируем ещё

```
Map<Method, List<Sample>>
```

```
Map<Node, List<Sample>>
```

450 000  250 000



Результат

CPU (6 млн сэмплов):

Результат

CPU (6 млн сэмплов):

- < 1 секунды на первый уровень

Результат

CPU (6 млн сэмплов):

- < 1 секунды на первый уровень
  - 0.5 секунды

Результат

CPU (6 млн сэмплов):

- < 1 секунды на первый уровень
  - 0.5 секунды

- < 5 секунд на первые 20 уровней

Результат

CPU (6 млн сэмплов):

- < 1 секунды на первый уровень
  - 0.5 секунды

- < 5 секунд на первые 20 уровней
  - 3 секунды

Результат

Alloc (78 млн сэмплов):

- < 5 секунд на первый уровень
  - 3 секунды

- < 10 секунд на первые 20 уровней
  - 2 секунды

Результат

Alloc (78 млн сэмплов):

- < 5 секунд на первый уровень
  - 3 секунды

- < 10 секунд на первые 20 уровней
  - 2 секунды на все уровни

https://github.com/Artyomcool/async-profiler/tree/heatmap

Выводы

Выводы

- Мы хотели...

Выводы

- Мы хотели...
- Мы добились...

Выводы

- Мы хотели...
- Мы добились...
- Бла-бла-бла...

Выводы

- Алгоритмы правда помогают, если данных много

Выводы

- Алгоритмы правда помогают, если данных много
- Высокой производительности нельзя достичь, не открыв “чёрные ящики”

Выводы

- Алгоритмы правда помогают, если данных много
- Высокой производительности нельзя достичь, не открыв “чёрные ящики”
- Частные решения зачастую эффективнее общих

Вопросы?

E-mail: artyomcool2@gmail.com
Tg: @Artyomcool