

# **Lobzik: Semiautomatic Modularisation**

**Mikhail Levchenko**

**What we know about  
modularisation?**

**mobi**us  
2018 Piter

**Денис Неклюдов**  
90Seconds.tv

Как не состариться  
во время сборки: Карт  
и другие приключения



# What we know about modularisation?

- Reduces build times



**mobi**us

2018 Moscow

**Александр Блинов**

hh.ru

Властелин модулей



# What we know about modularisation?

- Reduces build times
- Improves separation of concerns



# What we don't know about modularisation?

- How to come up with a plan for modularisation?
- How to score extracted modules?
- How to monitor modularisation?

# What will we learn about modularisation?

- How to come up with a plan for modularisation?  
Community detection algorithms
- How to score extracted modules?  
Graph theory metrics for quality of modules
- How to monitor modularisation?  
Visualisations and Continuous Modularisation



# About me

- Senior Software Engineer @ Yandex.Classifieds
- Developer productivity geek
- Blogging at @izpodshtorki

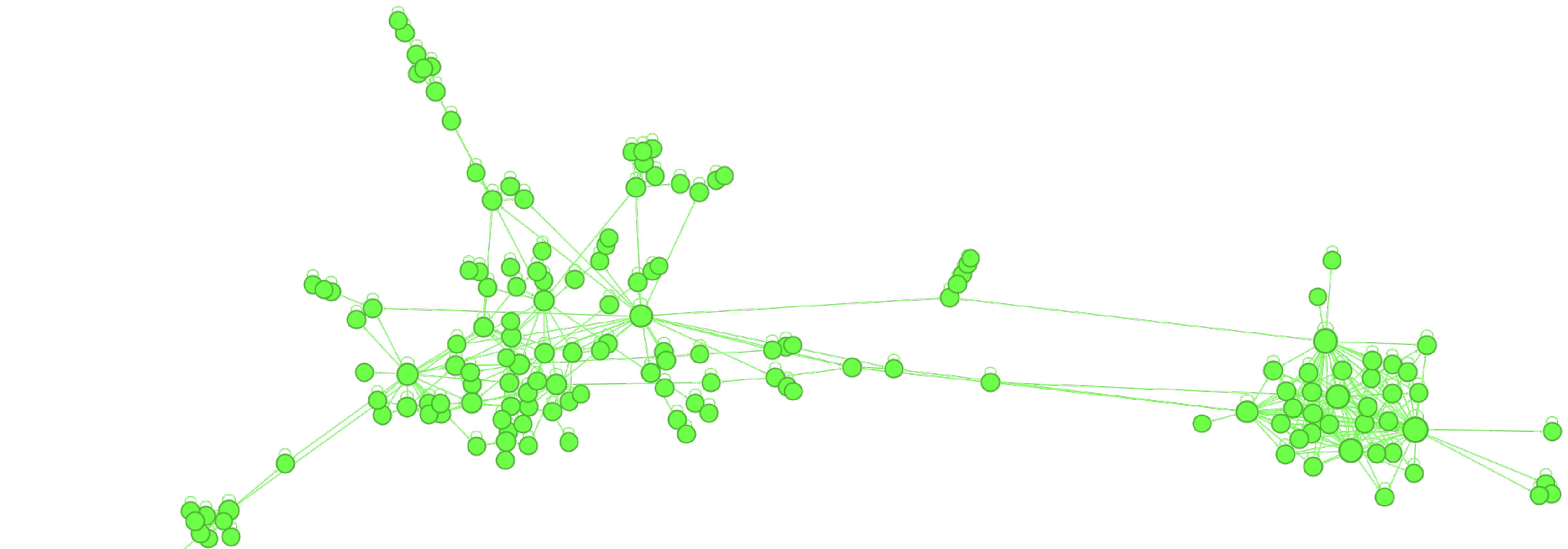




# State of our project in 2021

- >400k loc
- >200k loc в app
- team of ~10 people

**auto.ru**



 **feature-module**  
**20kloc**





**app-module**  
**>200k loc**



# How to scale modularisation?

# 20% Rule



# How not to scale modularisation

1. Developer picks up modularisation ticket

2. Week in dependency hell

3. Defeated developer leaves





**We need a plan.**

# The plan

- Give each potential module a ticket
- Evaluate each ticket
- Prioritise them by simplicity

**How to find modules without  
doing actual work?**

# Back to the Future

Смотрите, чего нашла <https://github.com/alexzaitsev/apk-dependency-graph>  
Строит **граф** зависимостей приложения


GitHub  
GitHub - alexzaitsev/apk-dependency-graph:  
Android class dependency visualizer. This tool helps to visualize the current state of the project. Android class dependency visualizer. This tool helps to visualize the current state of the project. - GitHub - alexzaitsev/apk-dependency-graph: Android class dependency visualizer. This tool helps...

alexzaitsev/apk-dependency-graph

Android class dependency visualizer. This tool helps to visualize the current state of the project.

4 Contributors 6 Issues 721 Stars 70 Forks

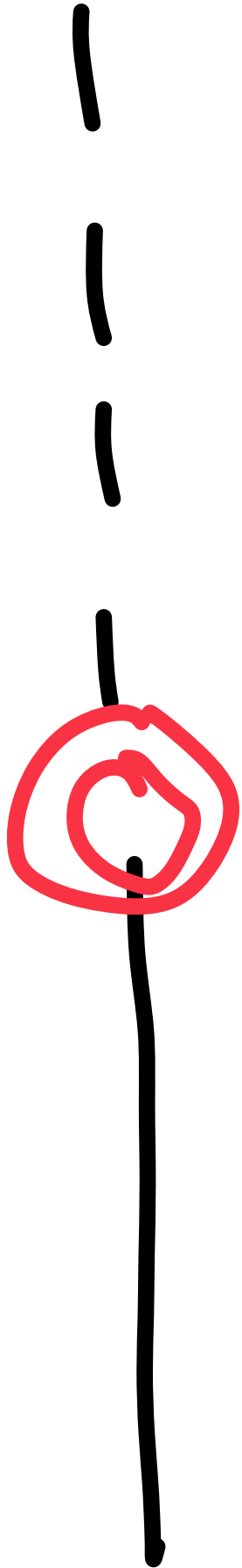
08:56

 image\_2018-03-05\_10-57-05.png  
1.1MB - [Show in Finder](#)

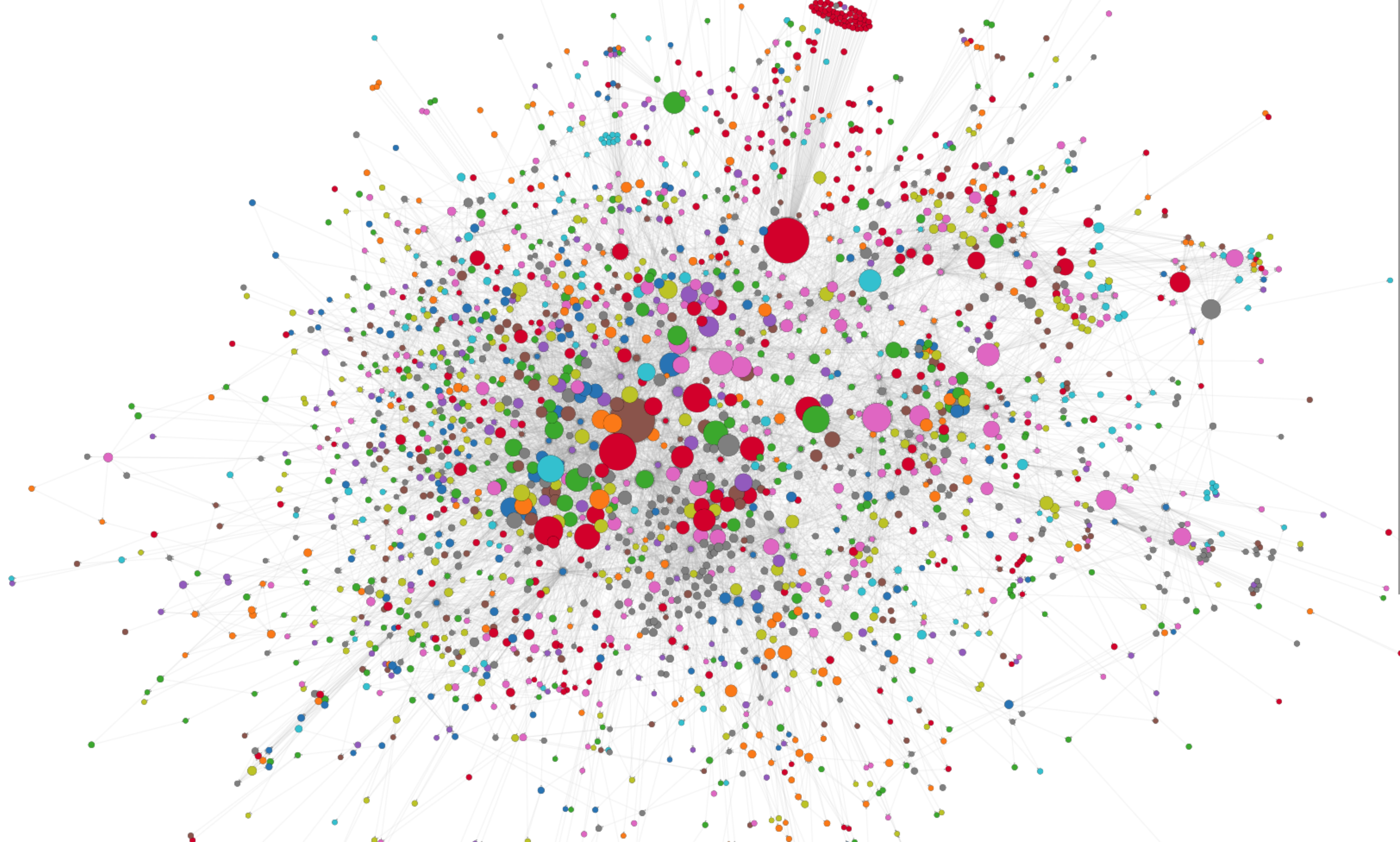
Вот наше) 08:57

А че так страшно-то? 08:59 ✓✓

year 2018

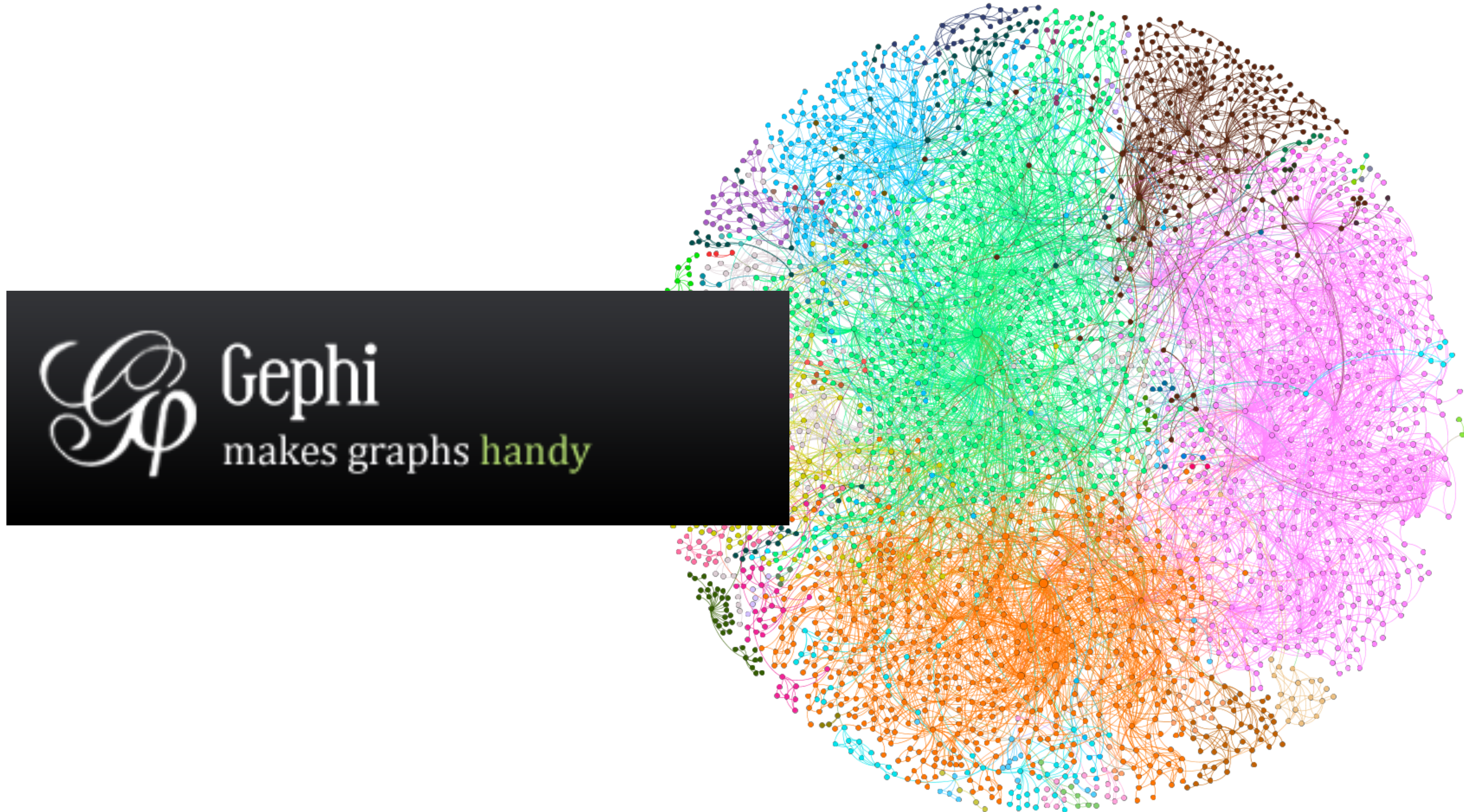








# MVP: android dependency graph + Gephi

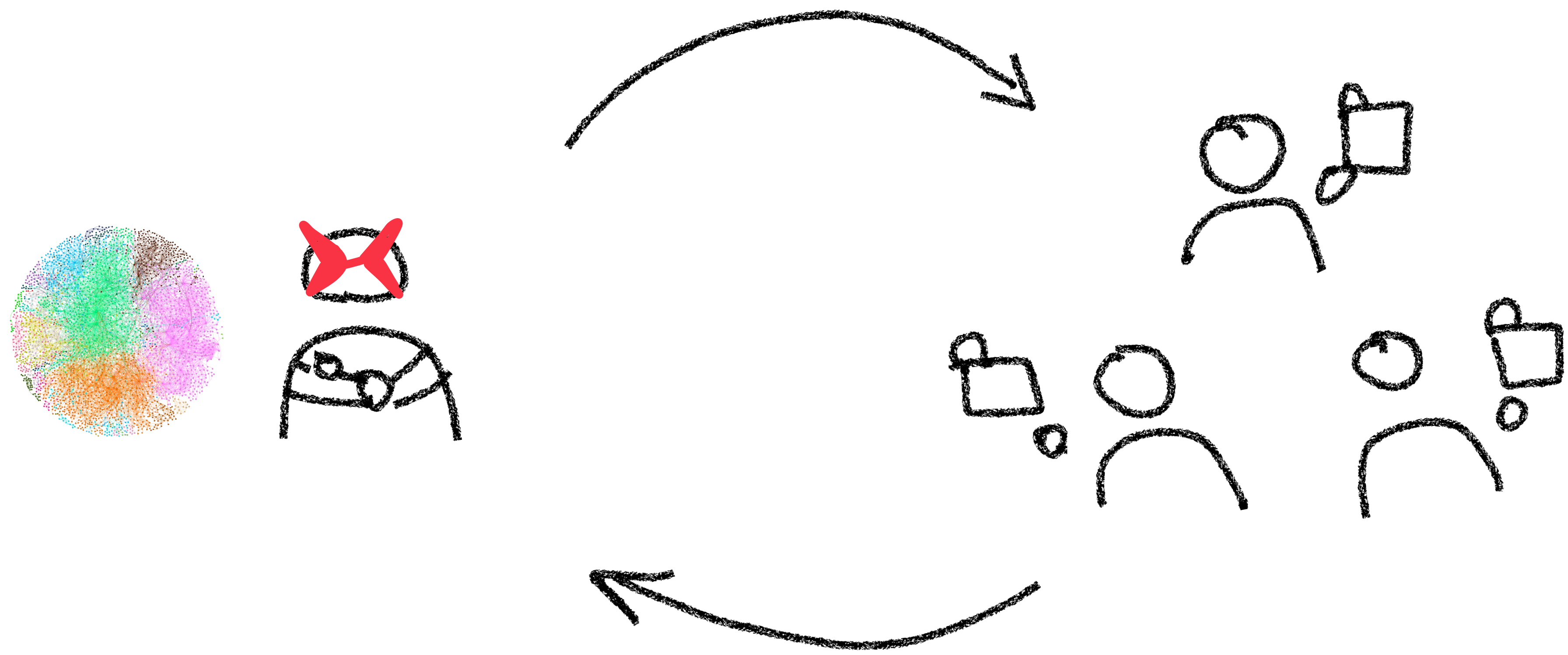




# Ord Visualisation



# Workflow

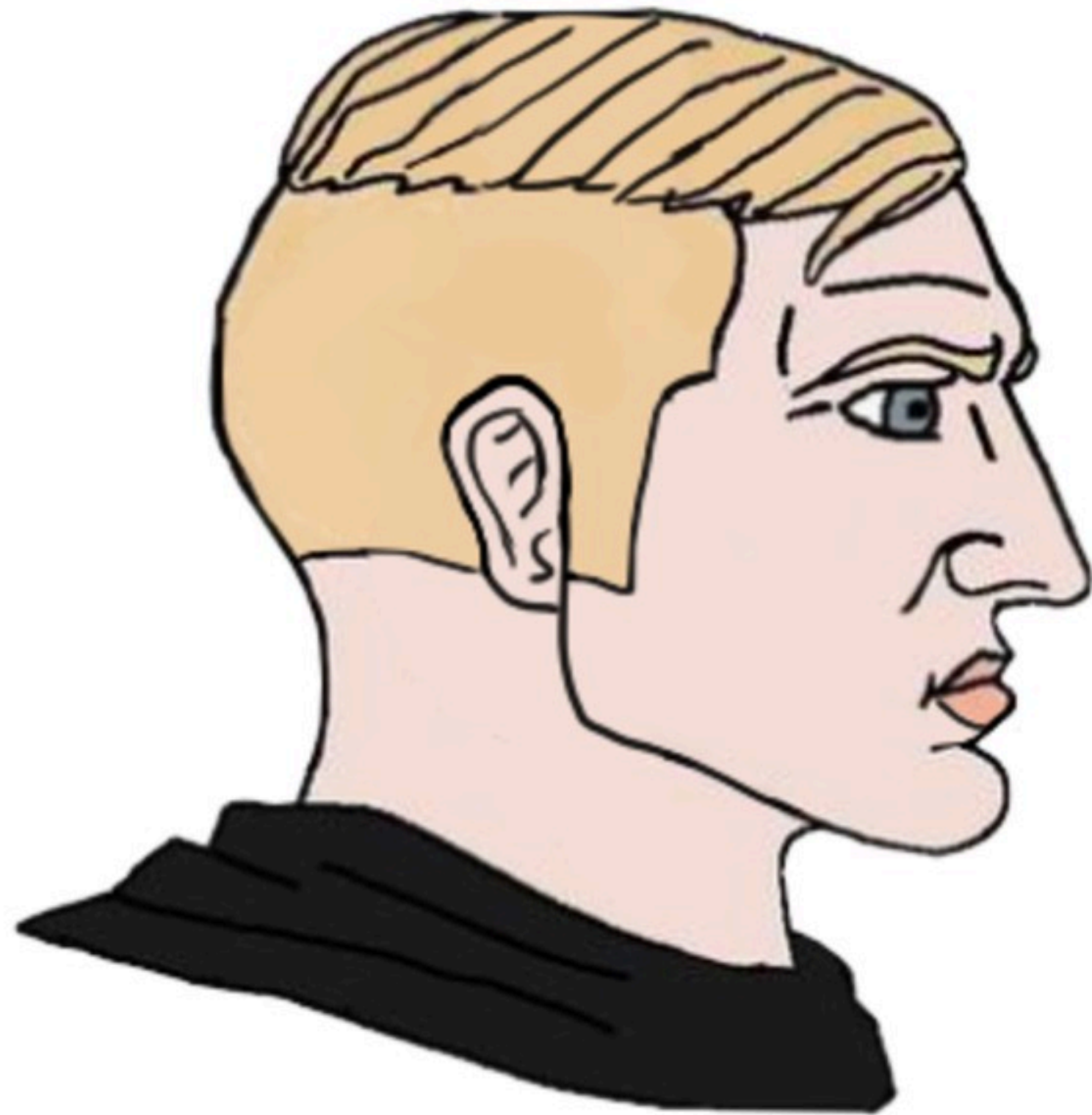


# MVP

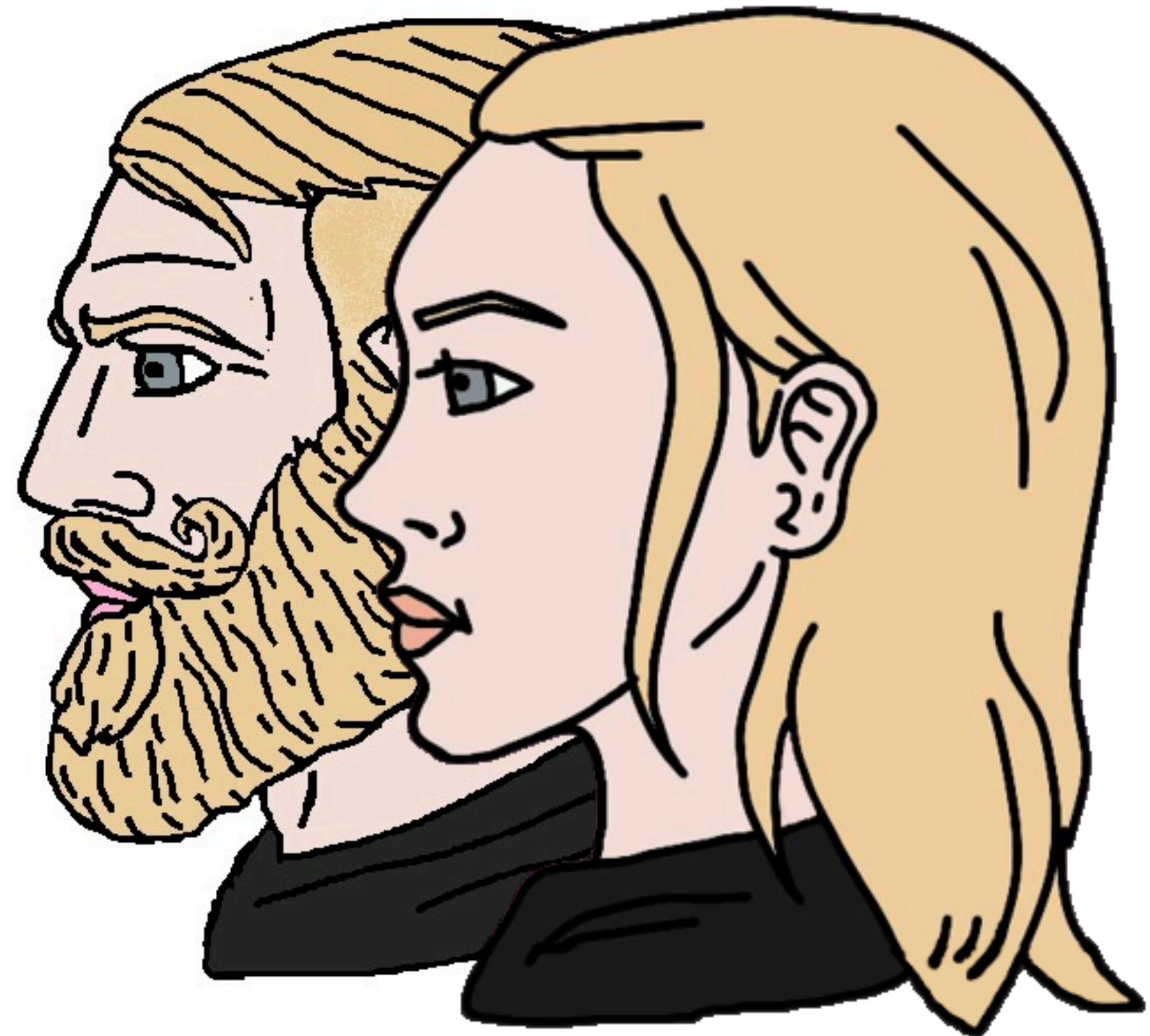
- Pros:
  - Every time we have an easy modularisation ticket to pick up
- Cons:
  - We need a lot of senior engineer time
  - Tickets needs to be kept up-to-date



# How to replace engineers?



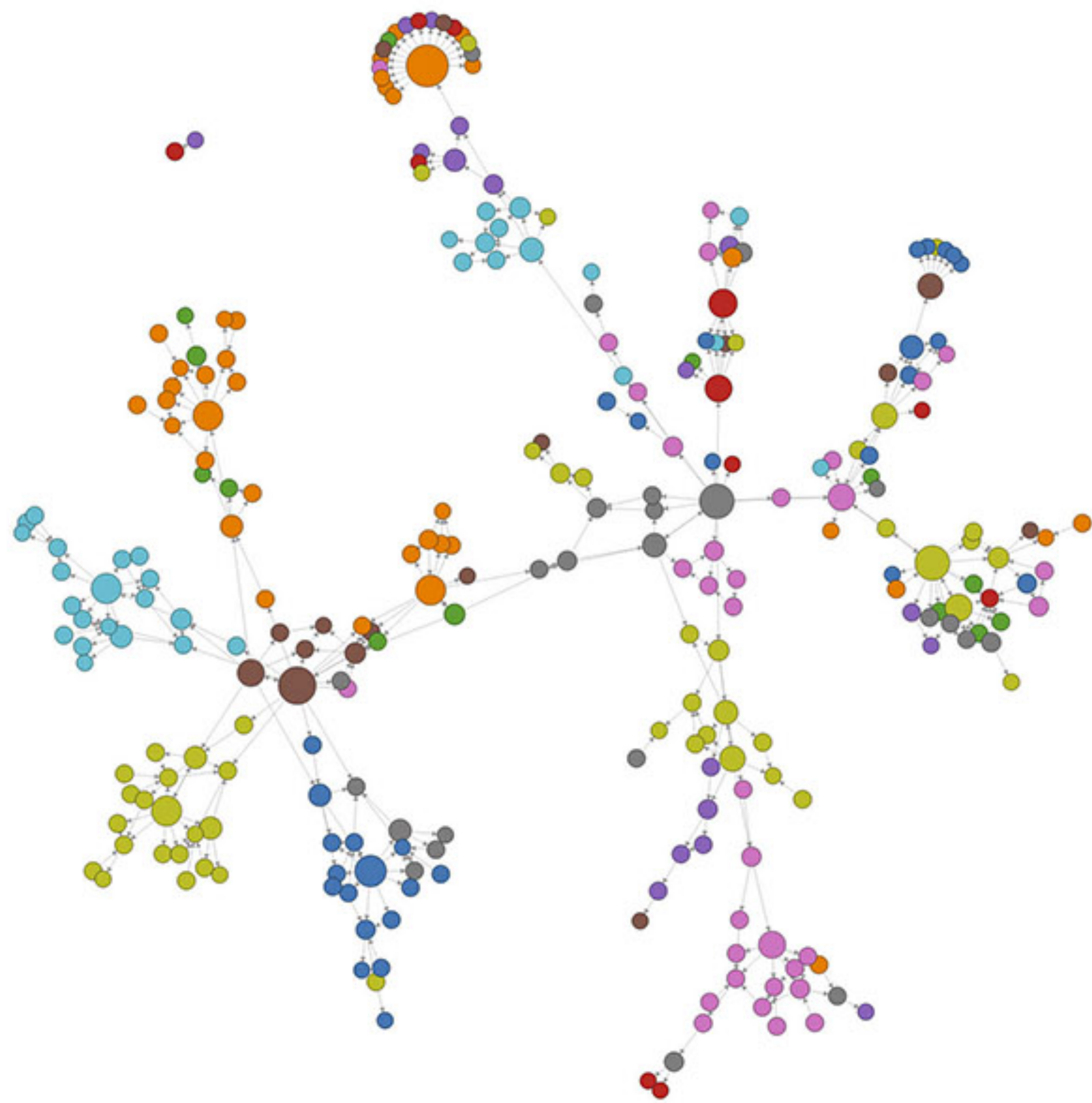
I wrote a program that will  
make us loose our jobs



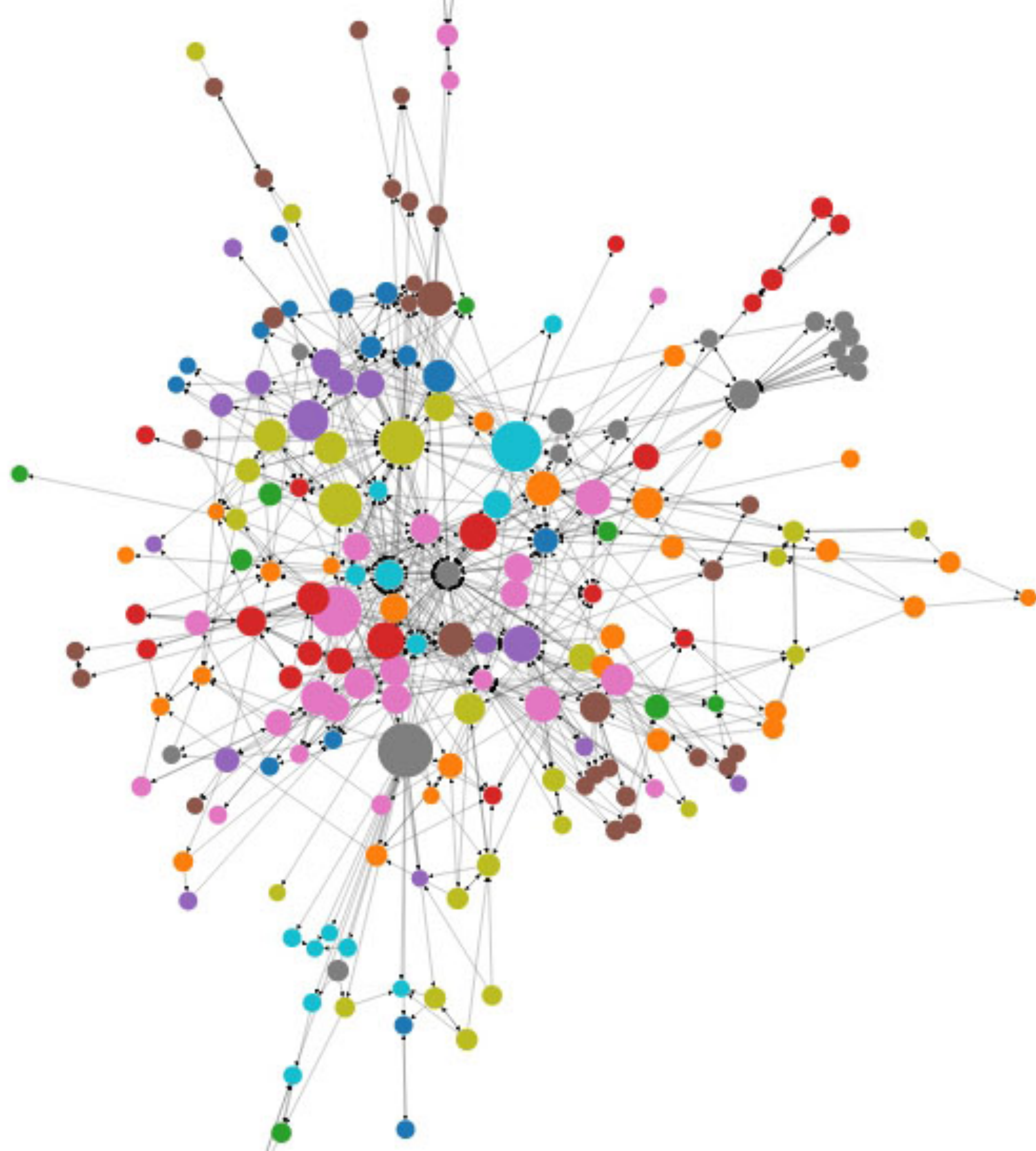
Finally

**What is the objective function to optimise?**

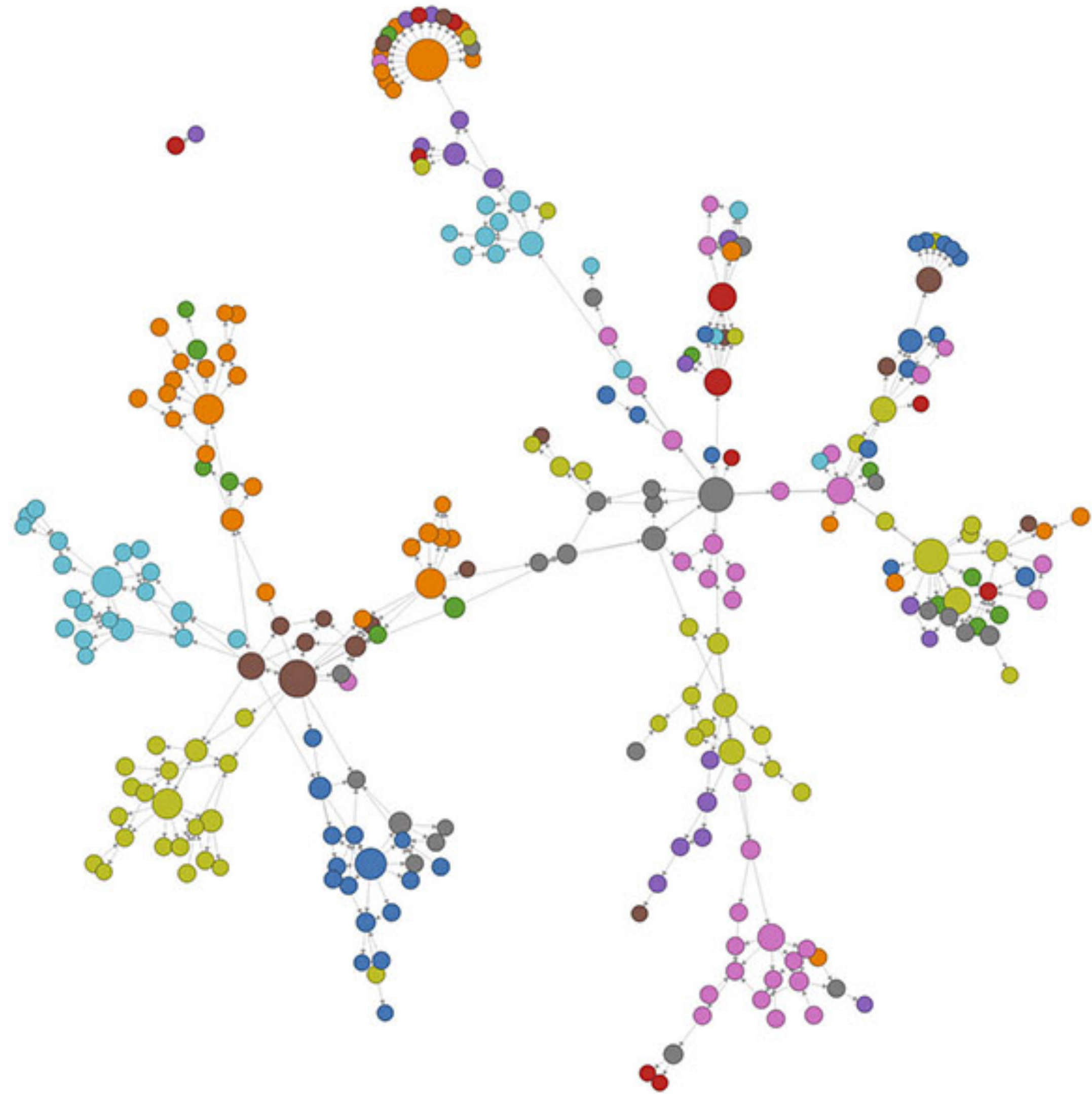








**Why №1 is "better"?**

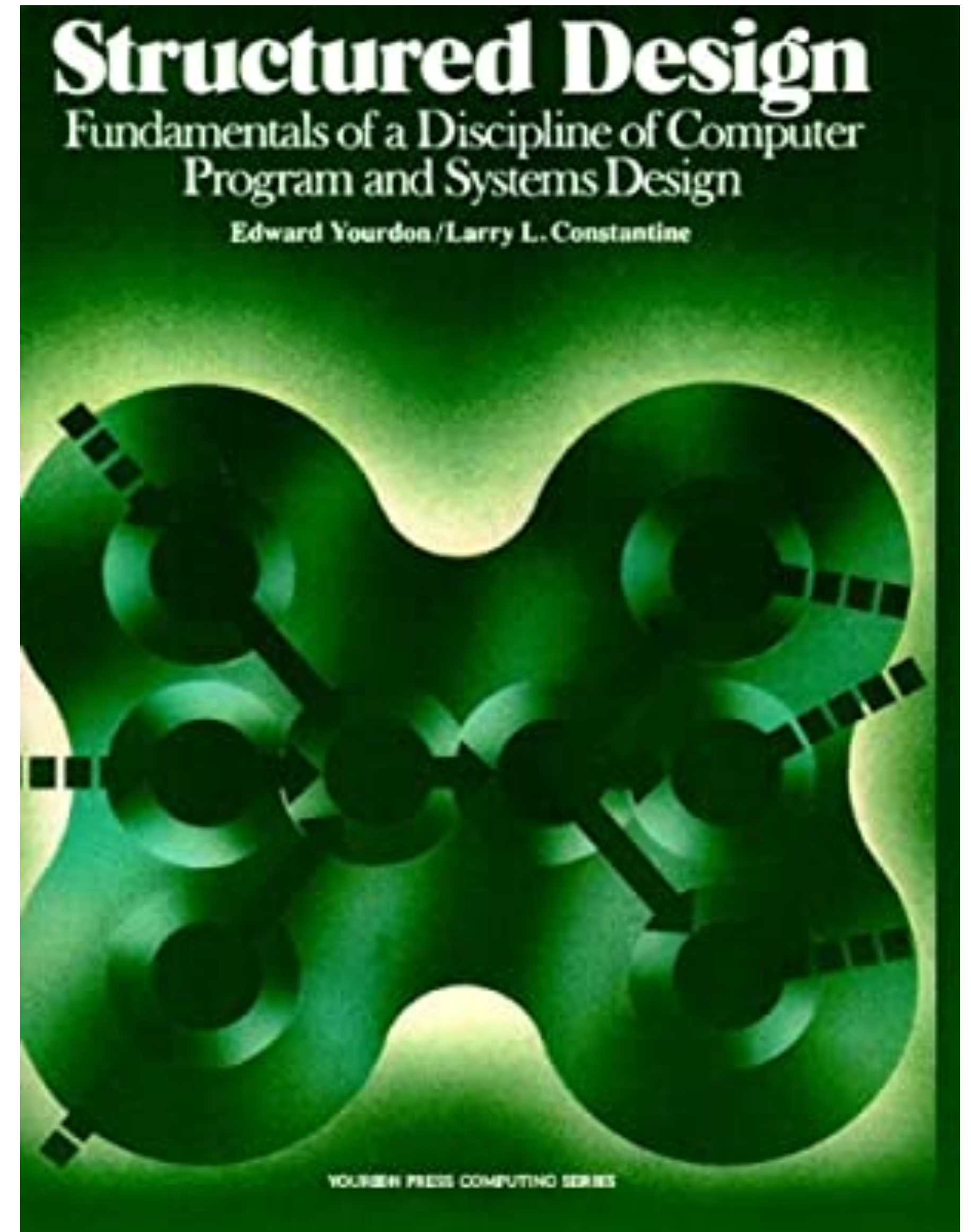




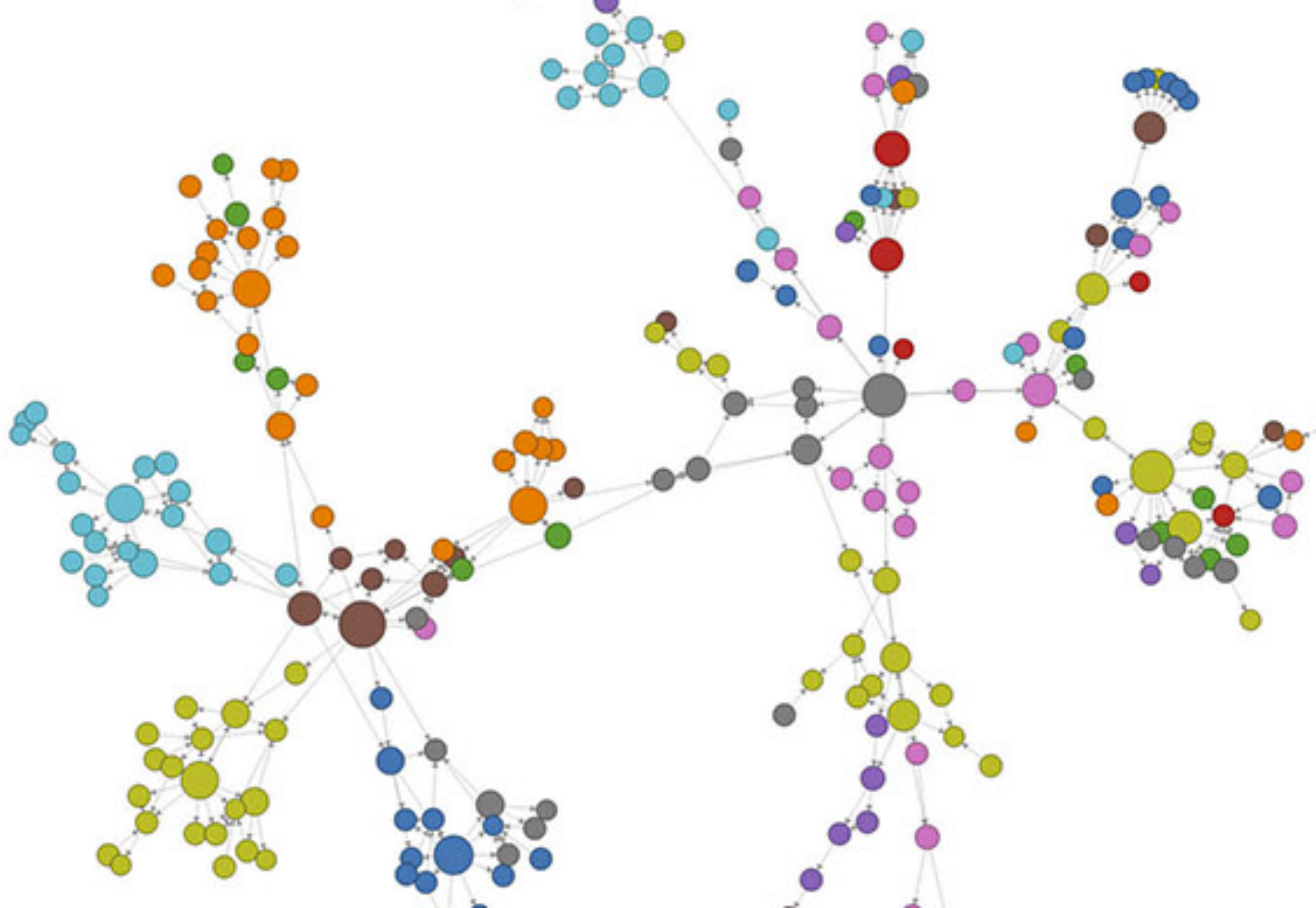
# Structured Design

Yourdon, Constantine 1974

- "The objective here is to reduce coupling by striving for high cohesion"
- Low Coupling
- High Cohesion

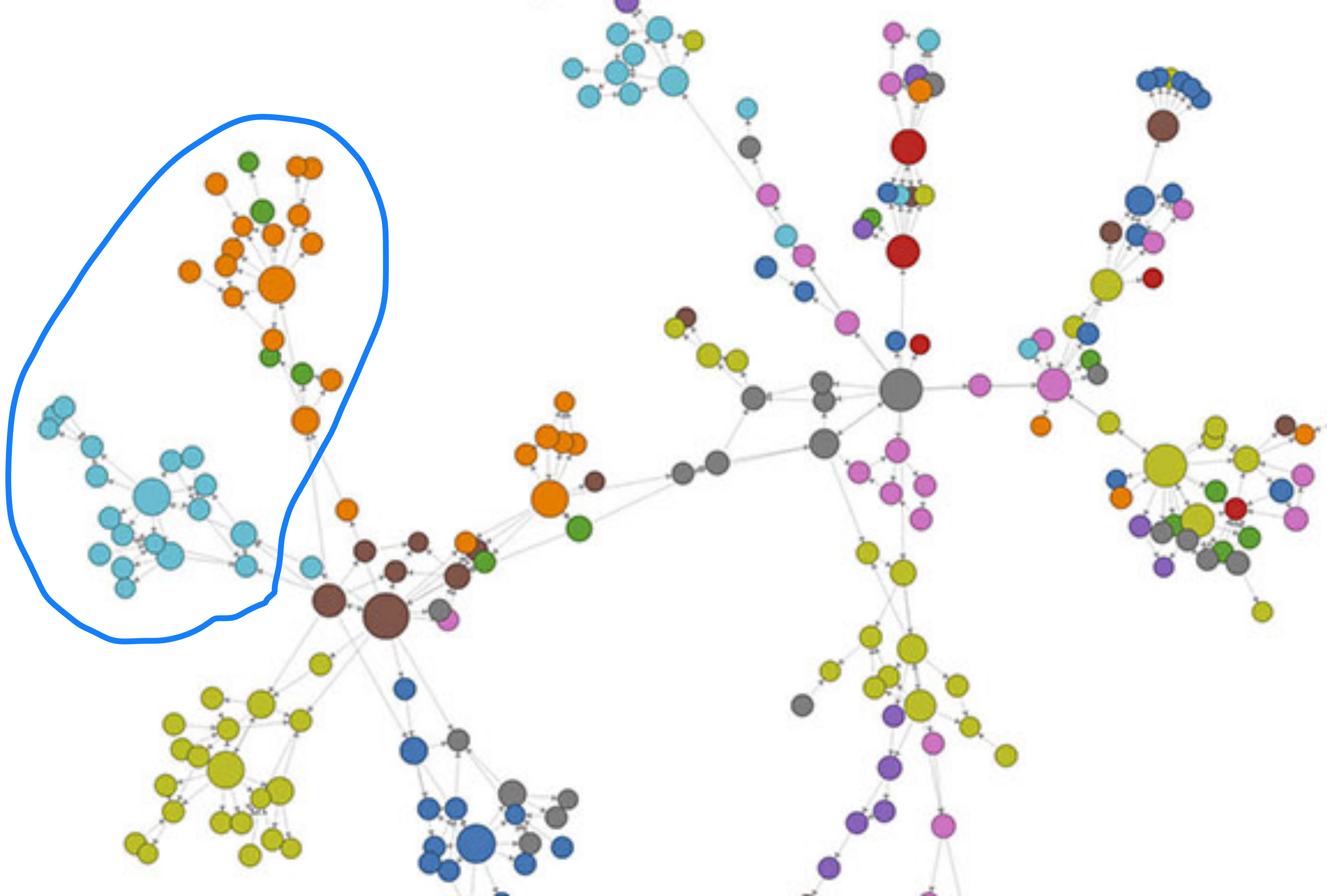








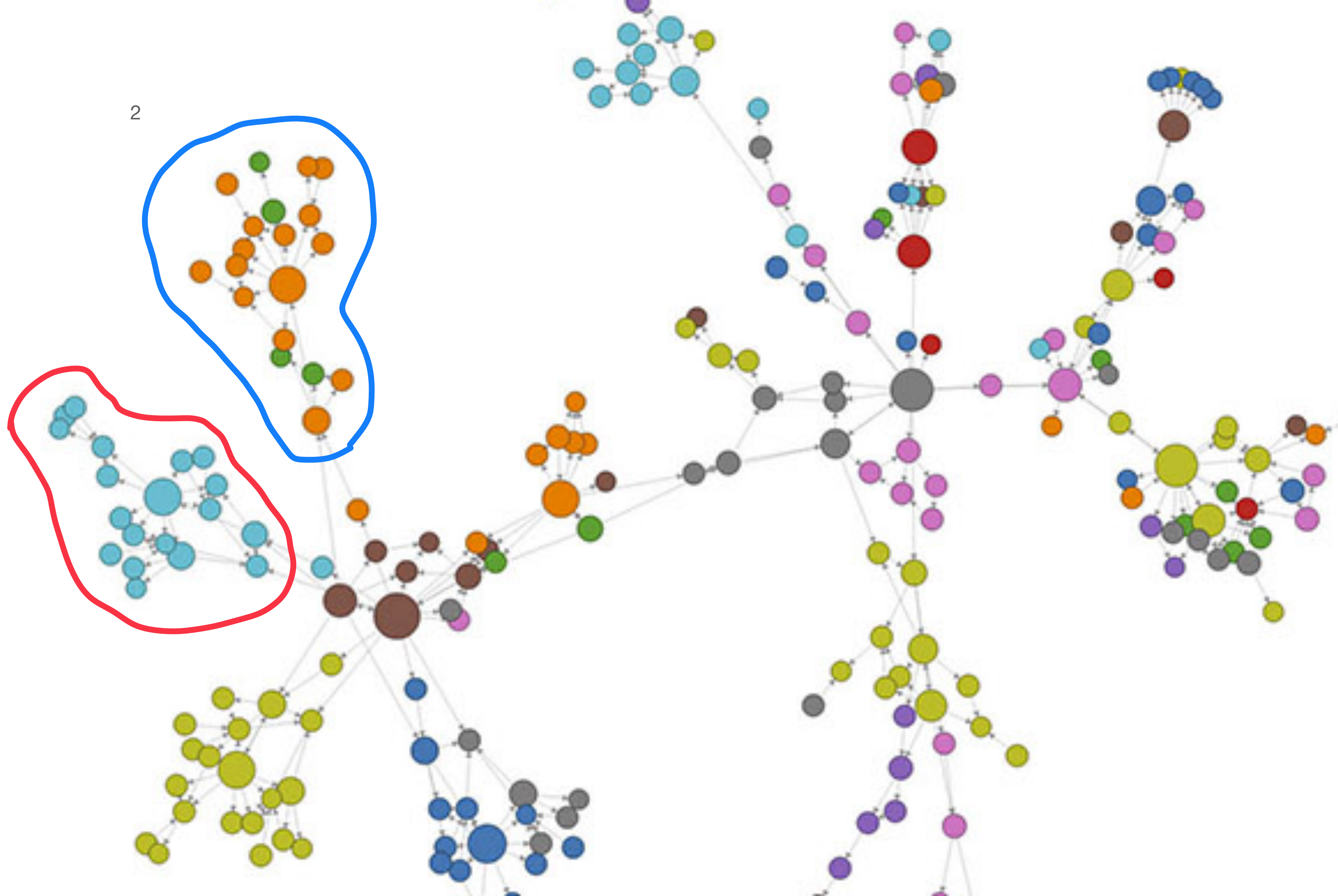
1



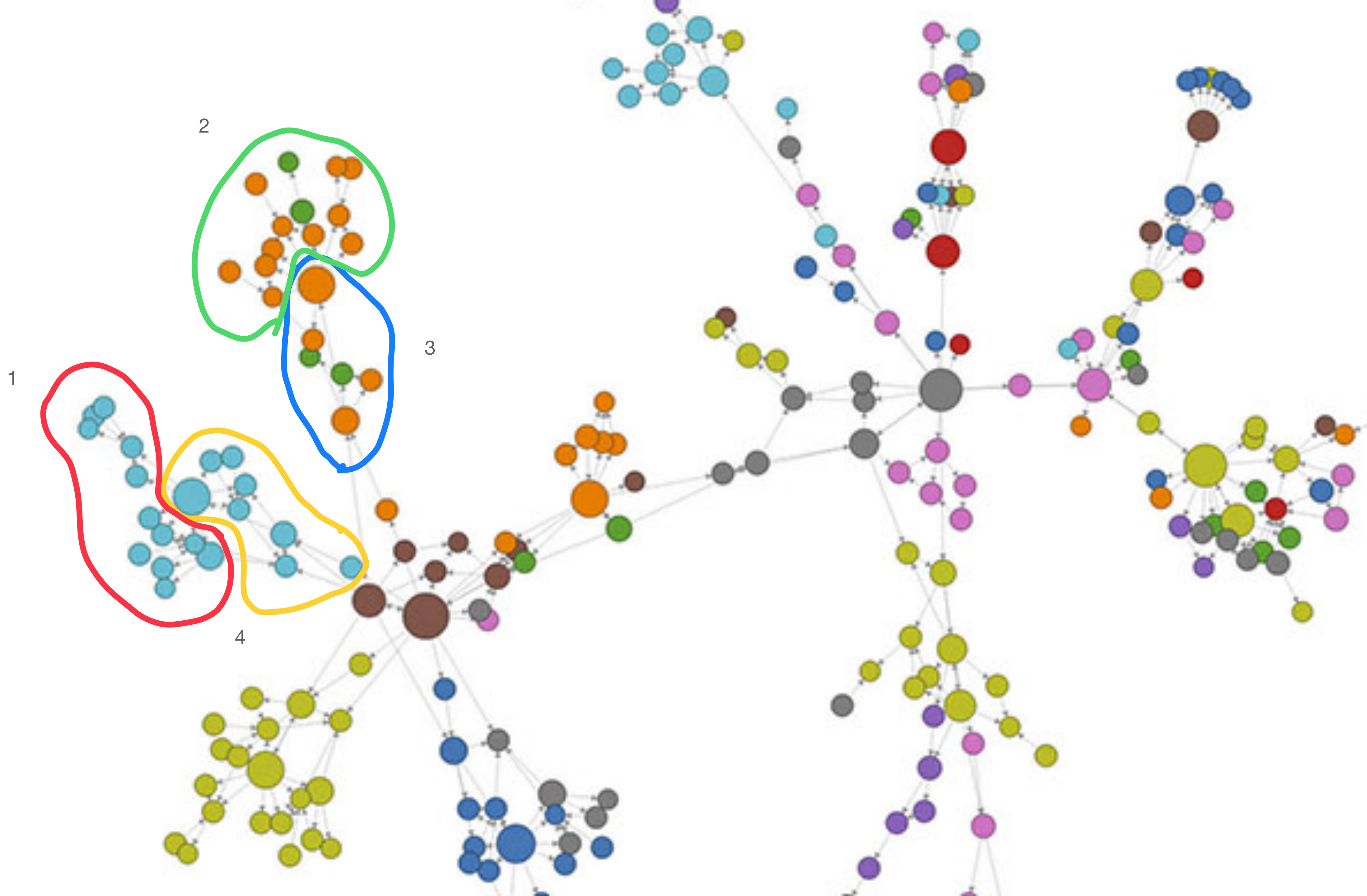


1

2









# Modularity

Clauset, Aaron, Newman and Moore, 2004

$$Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \gamma \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

# Engineeri-fy



Freya Holmér  
@FreyaHolmer

...

btw these large scary math symbols are just for-loops

[Перевести твит](#)

**Summation**  
(capital sigma)

$$\sum_{n=0}^4 3n$$

```
sum = 0;
for( n=0; n<=4; n++ )
    sum += 3*n;
```

**Product**  
(capital pi)

$$\prod_{n=1}^4 2n$$

```
prod = 1;
for( n=1; n<=4; n++ )
    prod *= 2*n;
```

4:21 PM · 11 сент. 2021 г.

6 968 ретвитов   1 317 твитов с цитатами   37,8 тыс. отметка «Нравится»



# Engineeri-fy

```
sum = 0
for i in g.nodes.count:
    for j in g.nodes.count:
        ni = g.nodes[i]
        nj = g.nodes[j]
        if sameModule(ni, nj):
            edge = g.edges[i,j] ? 1 : 0
            sum += edge -  $\gamma$  * (ni.degree * nj.degree) / 2 * m
```



# Modularity visualized

Mark Needham, Amy E. Hodler, 2019

Negative Modularity  
 $M=0.12$



Single Community  
 $M=0$



Suboptimal Partition  
 $M=0.22$



Optimal Partition  
 $M=0.41$





# Demistify

- Modularity = Cohesion
- ? = Coupling

# Conductance

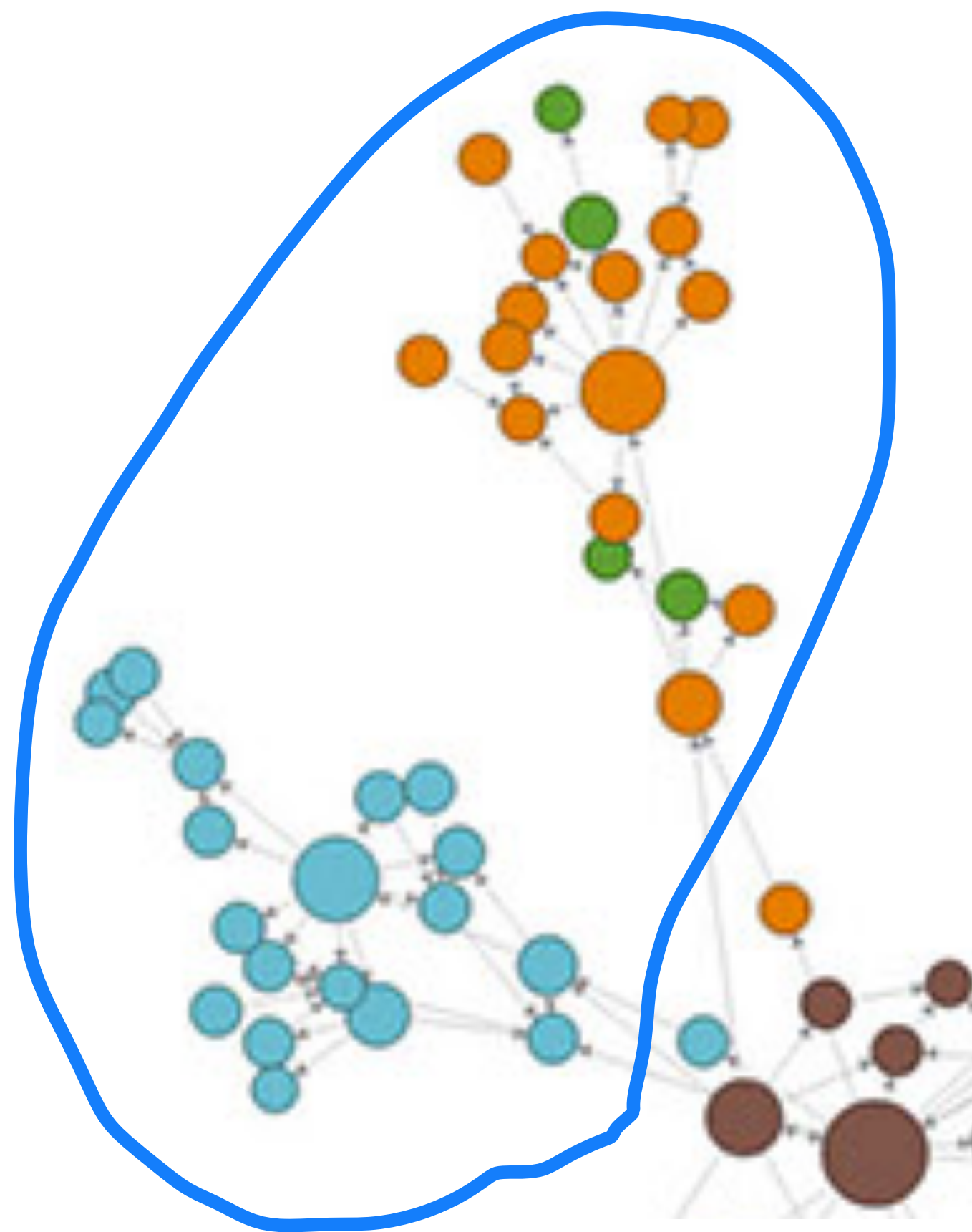
Yang and Leskovec, 2012

$$C = \frac{e_{out}}{2e_{in} + e_{out}}$$



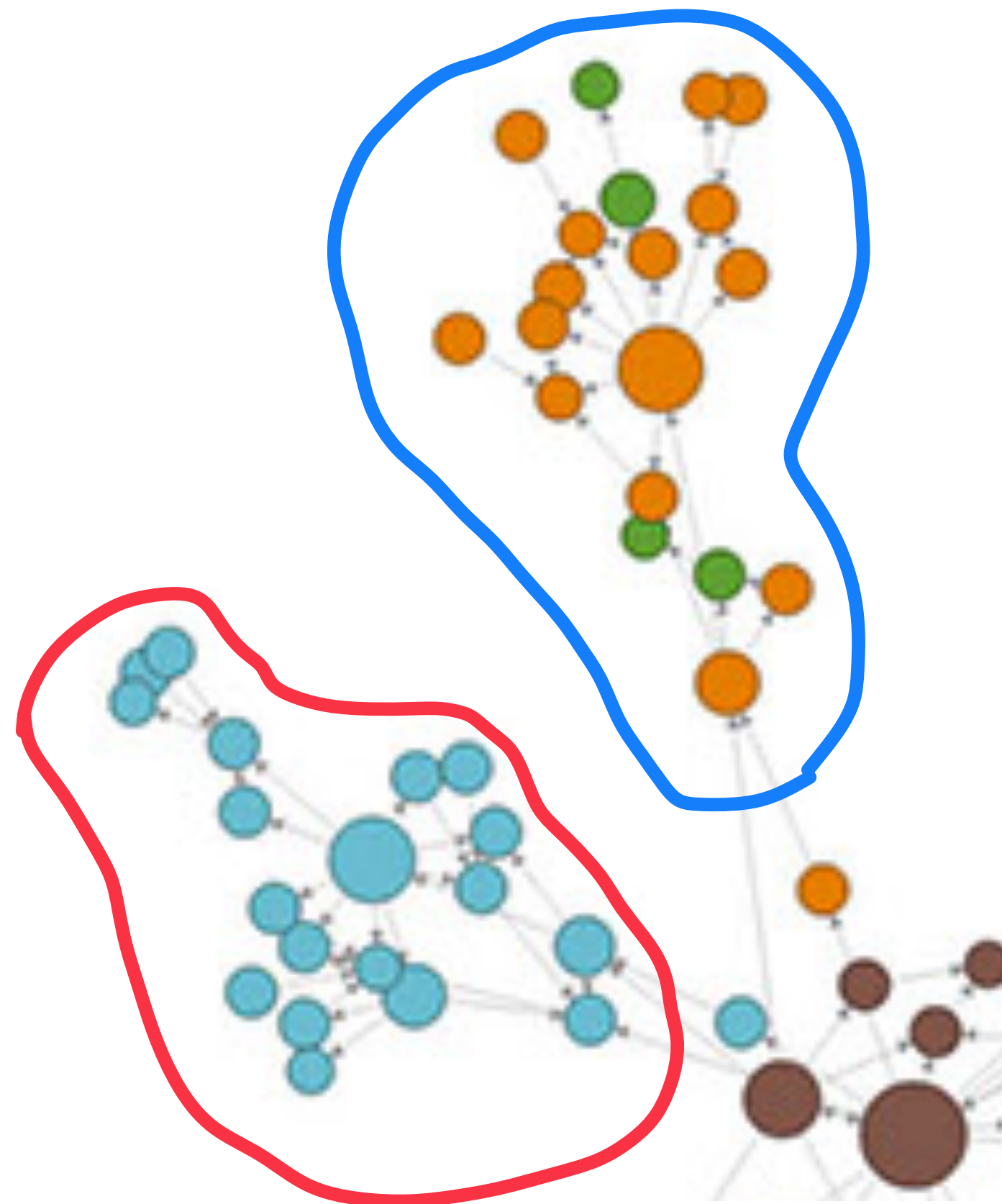
# Demistify

- Modularity = Cohesion
- Conductance = Coupling



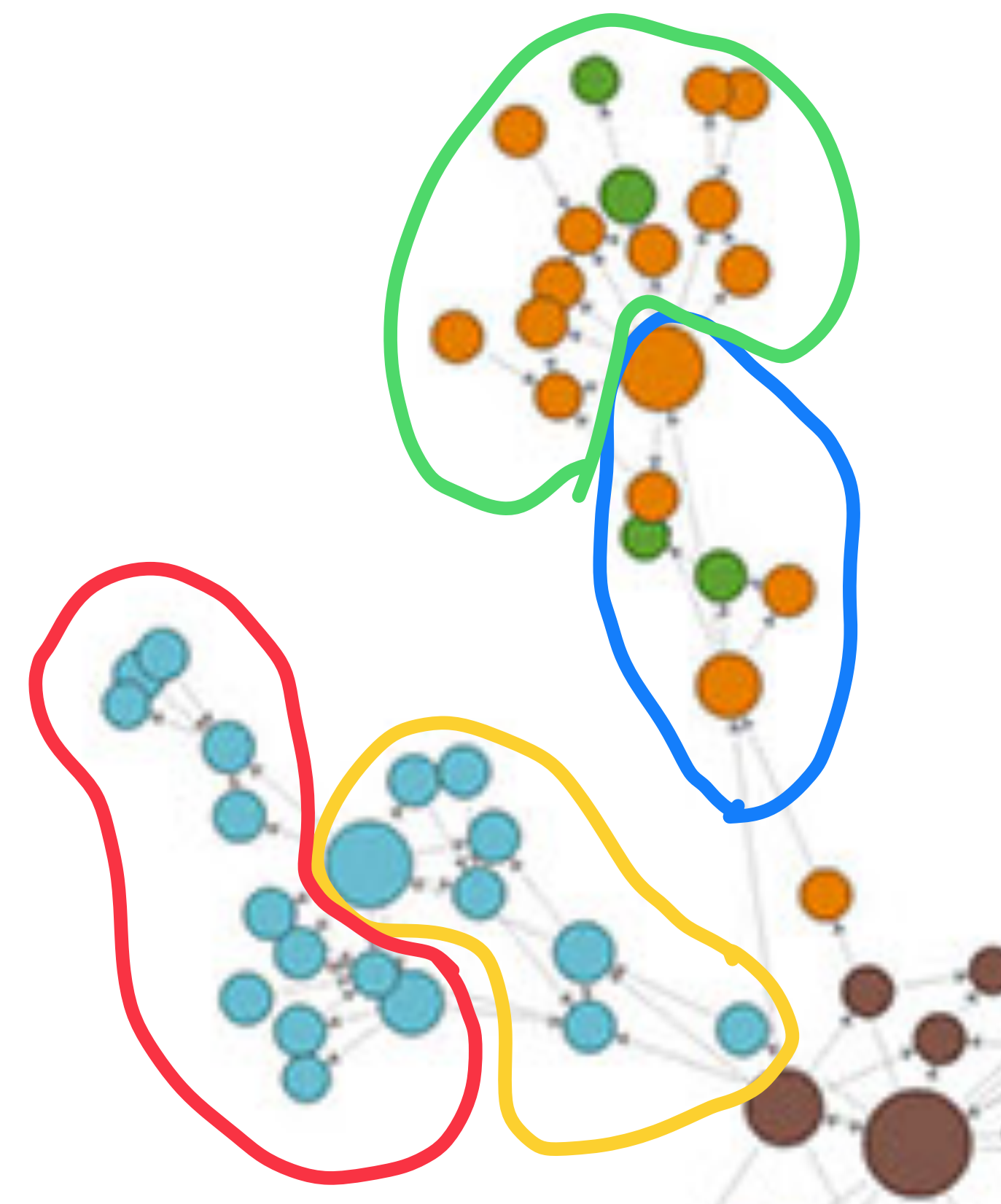
Mod: 0.1195

Cond: 0.166



Mod: 0.4257

Cond: 0.061



Mod: 0.4348

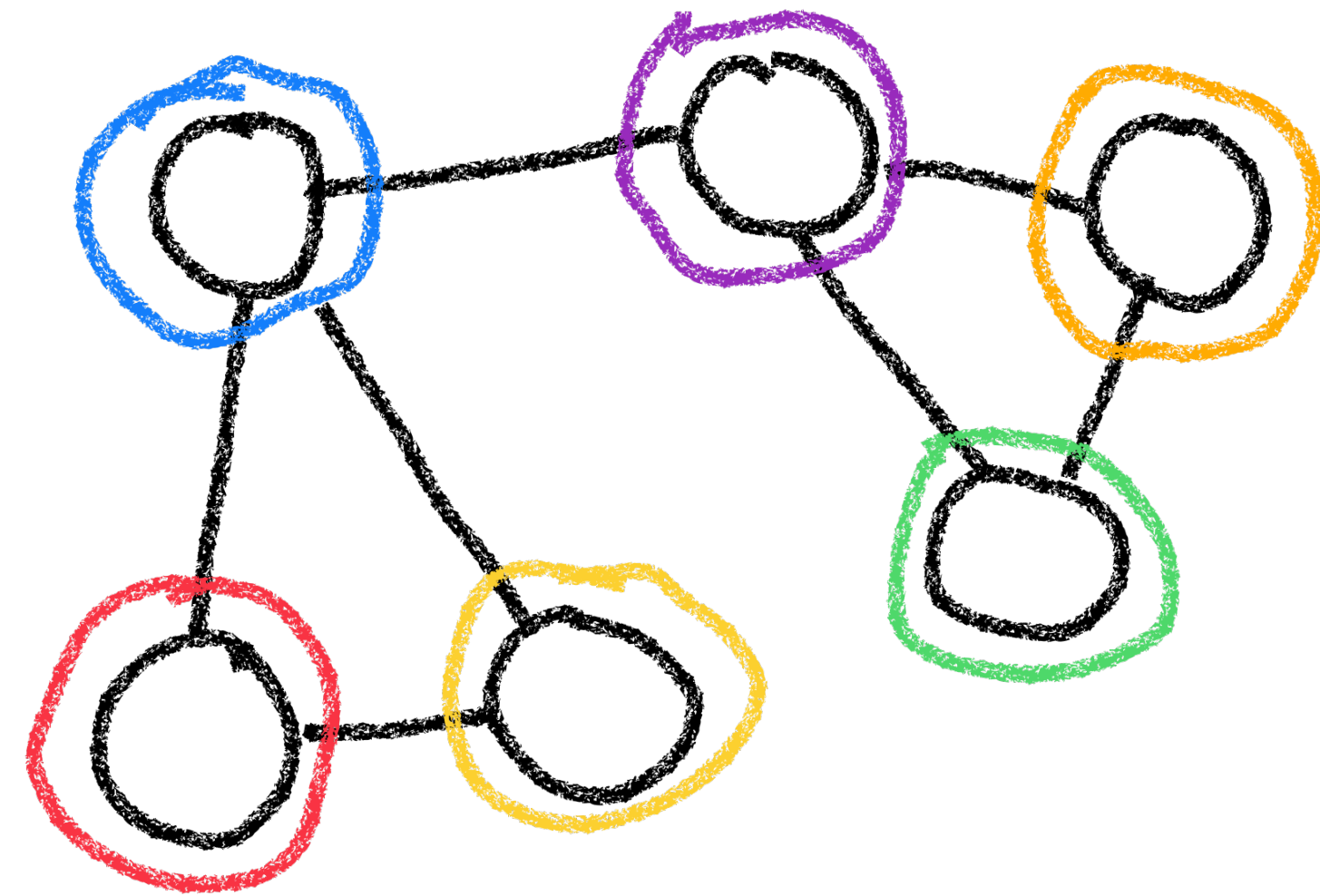
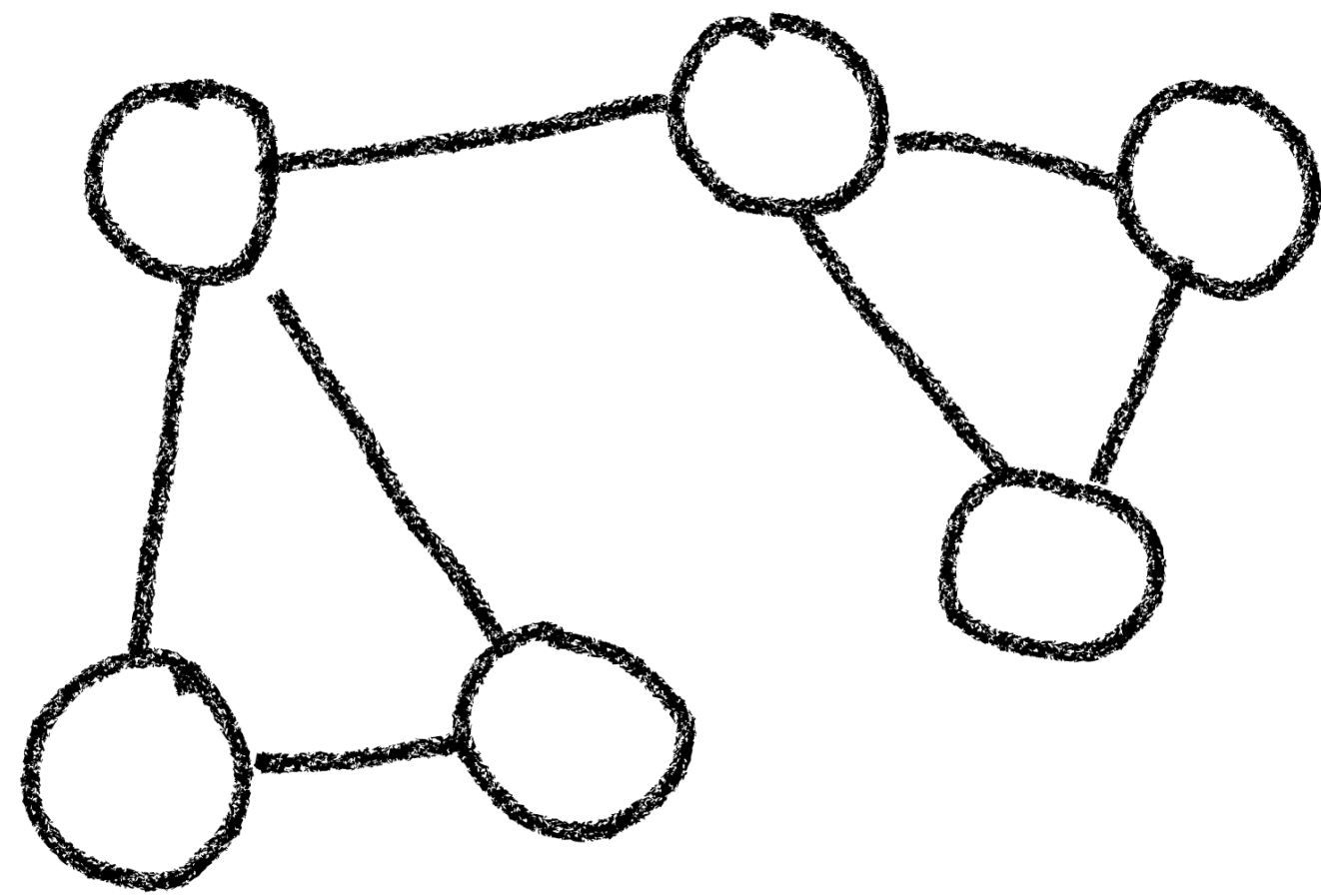
Cond: 0.192



# Community Detection Algorithms

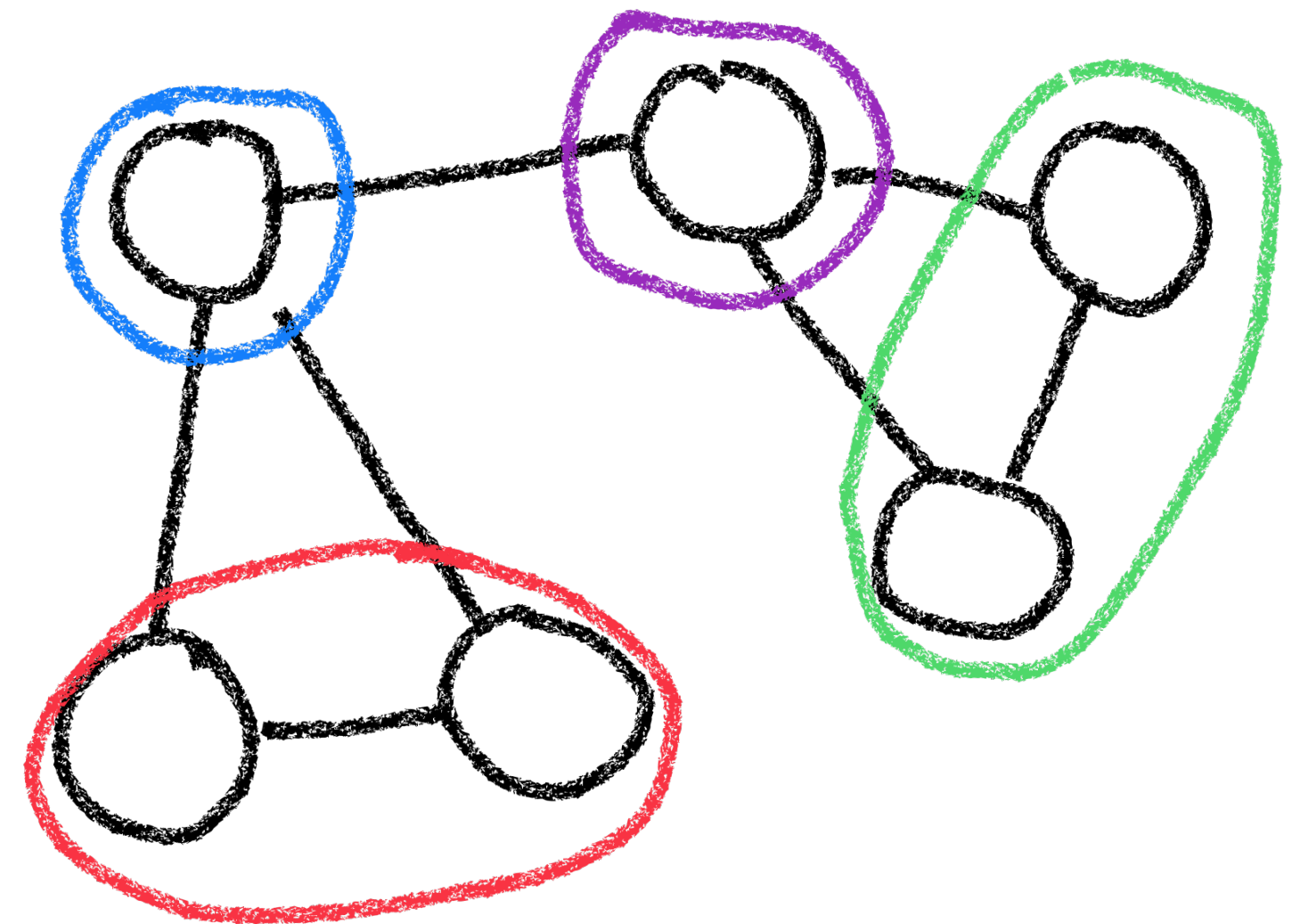
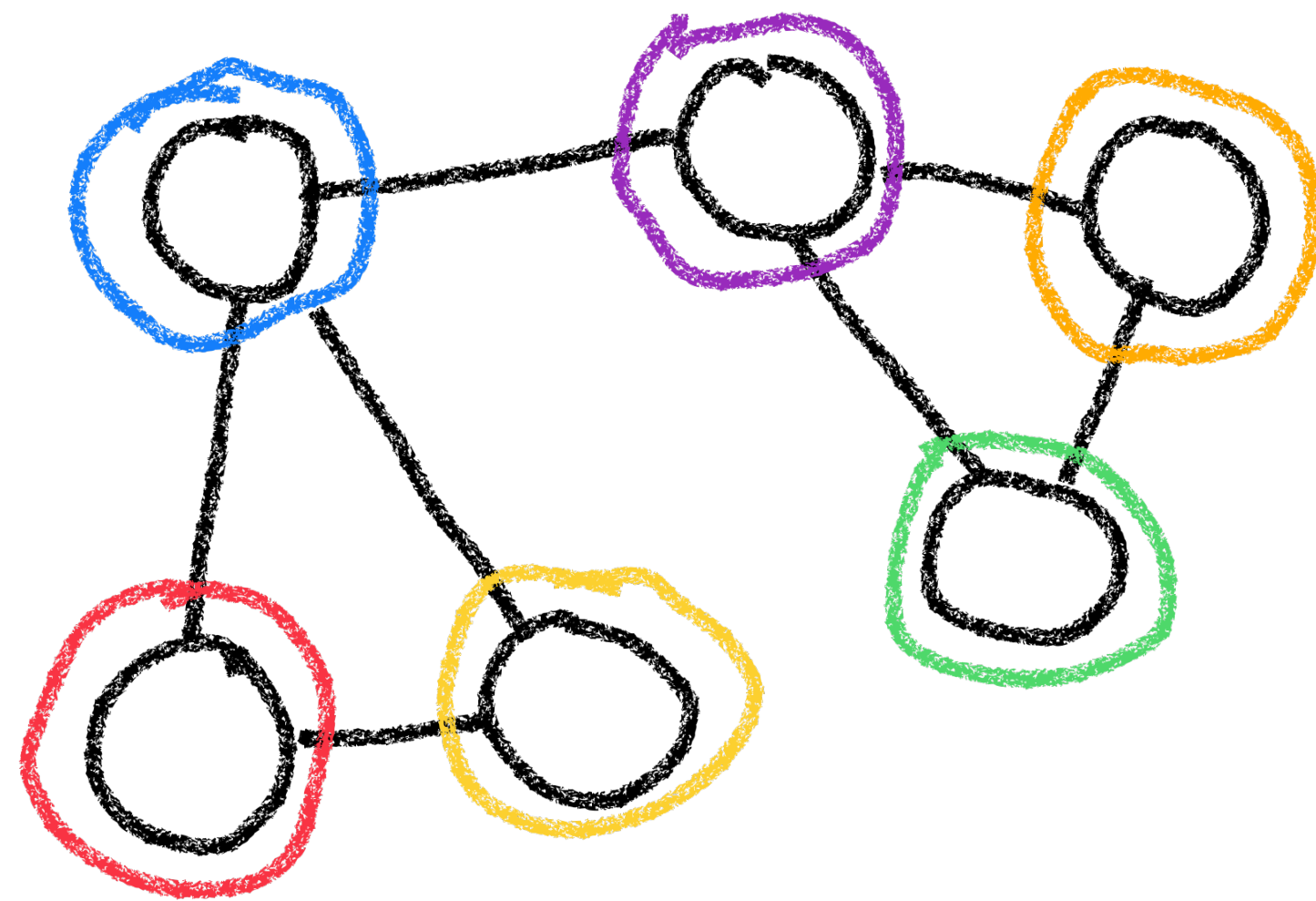
- Girvan – Newman
- Louvain Method

# Louvain Method

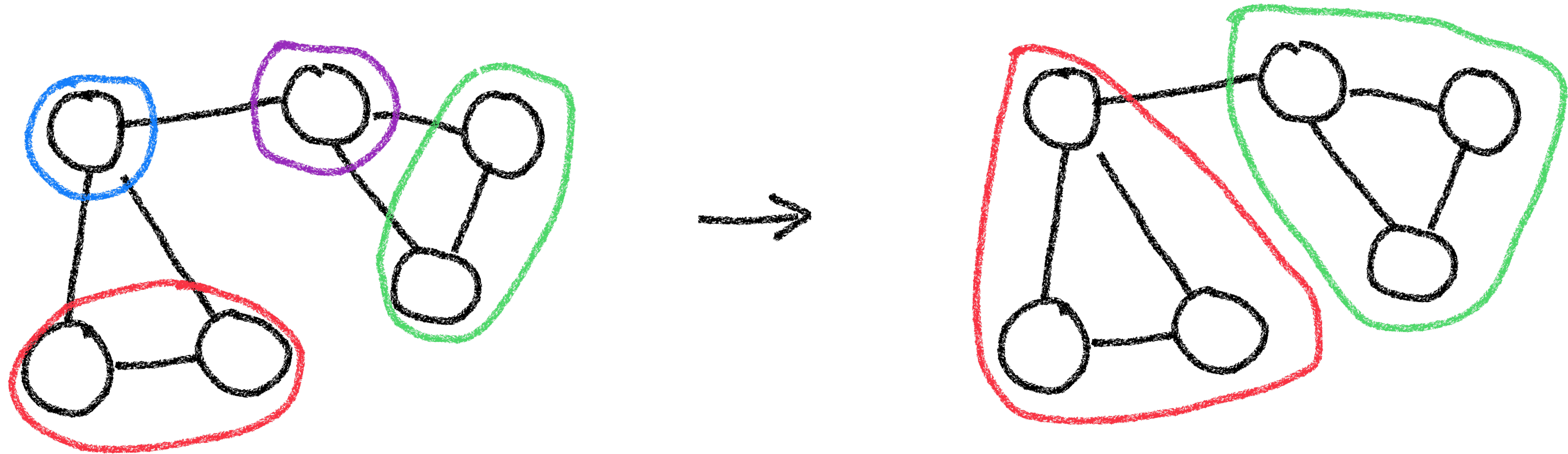




# Louvain Method



# Louvain Method

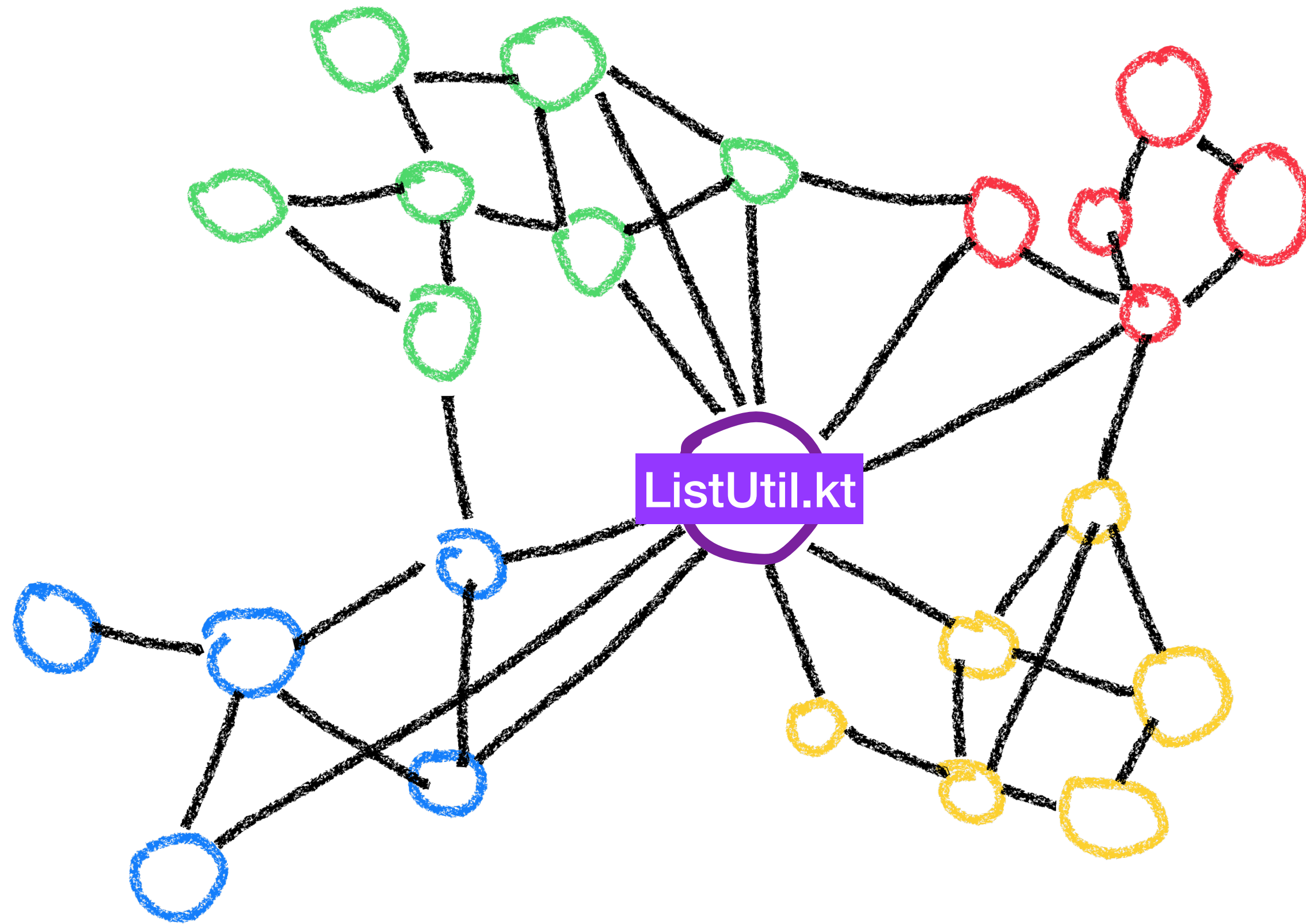






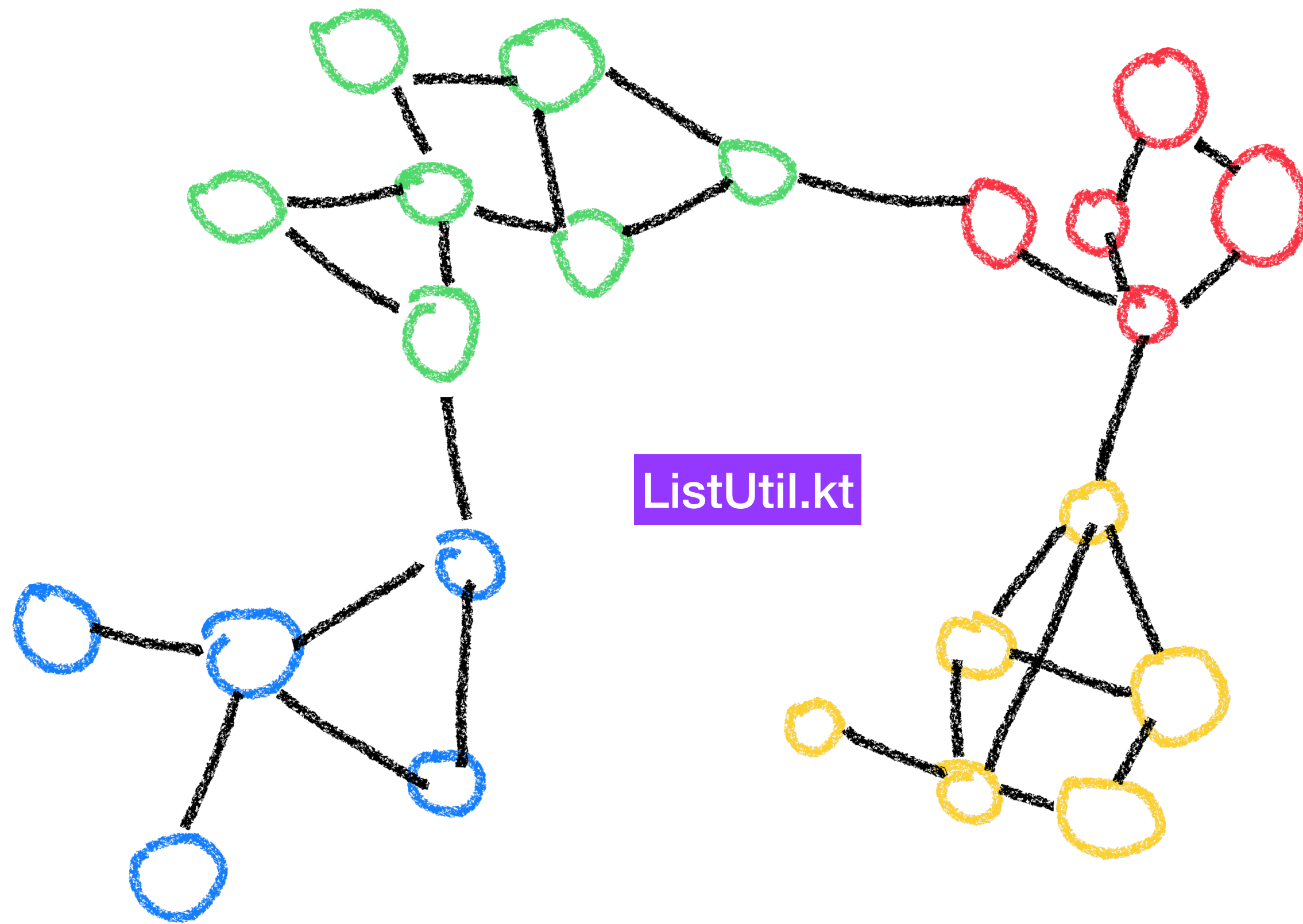


# "Star" problem

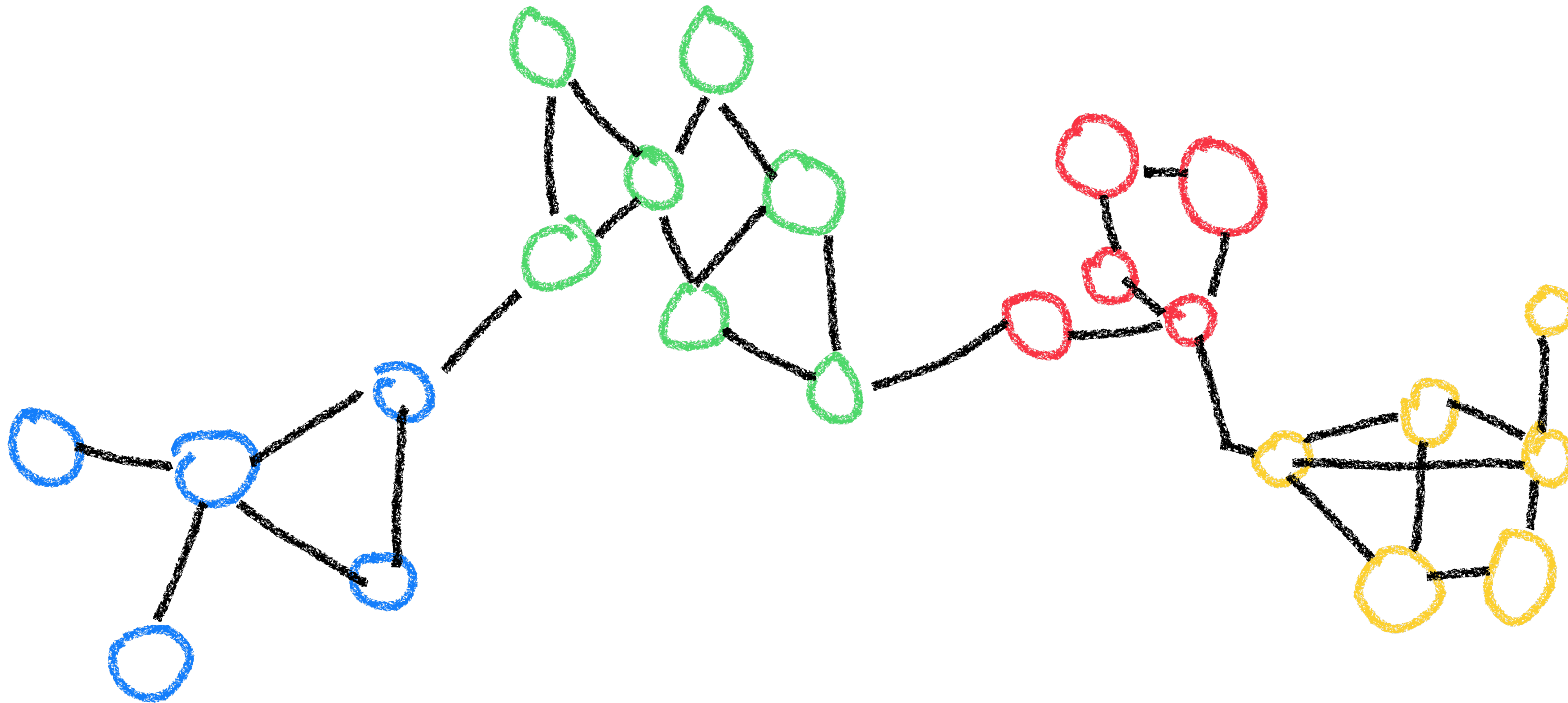




# "Star" problem



# "Star" problem





# How to filter "stars"?

- DI & Navigation

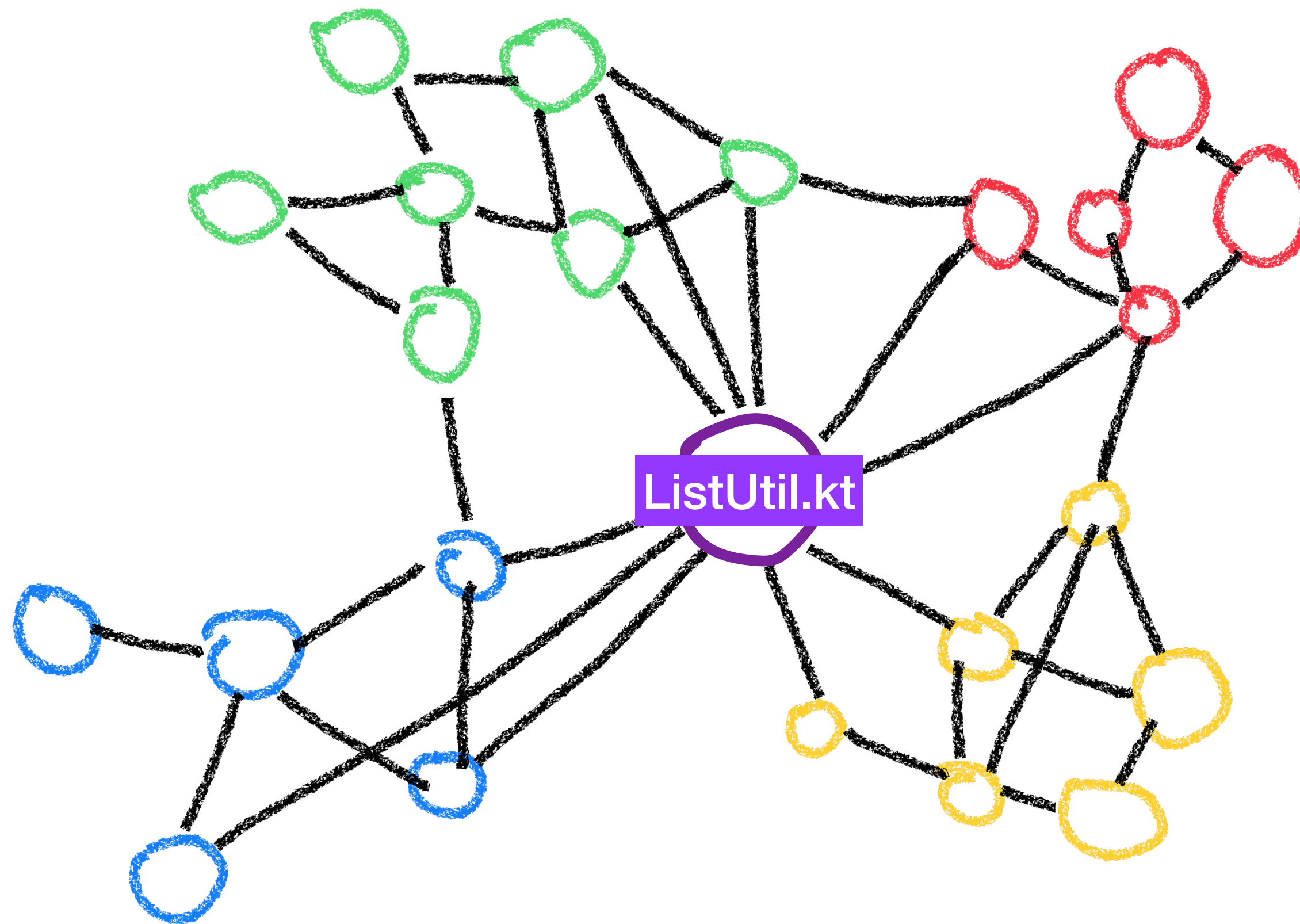
# How to filter "stars"?

- DI & Navigation
- core modules



# How to filter "stars"?

- Degree



# How to filter "stars"?

Jon M. Kleinberg, 1999

- Authorities

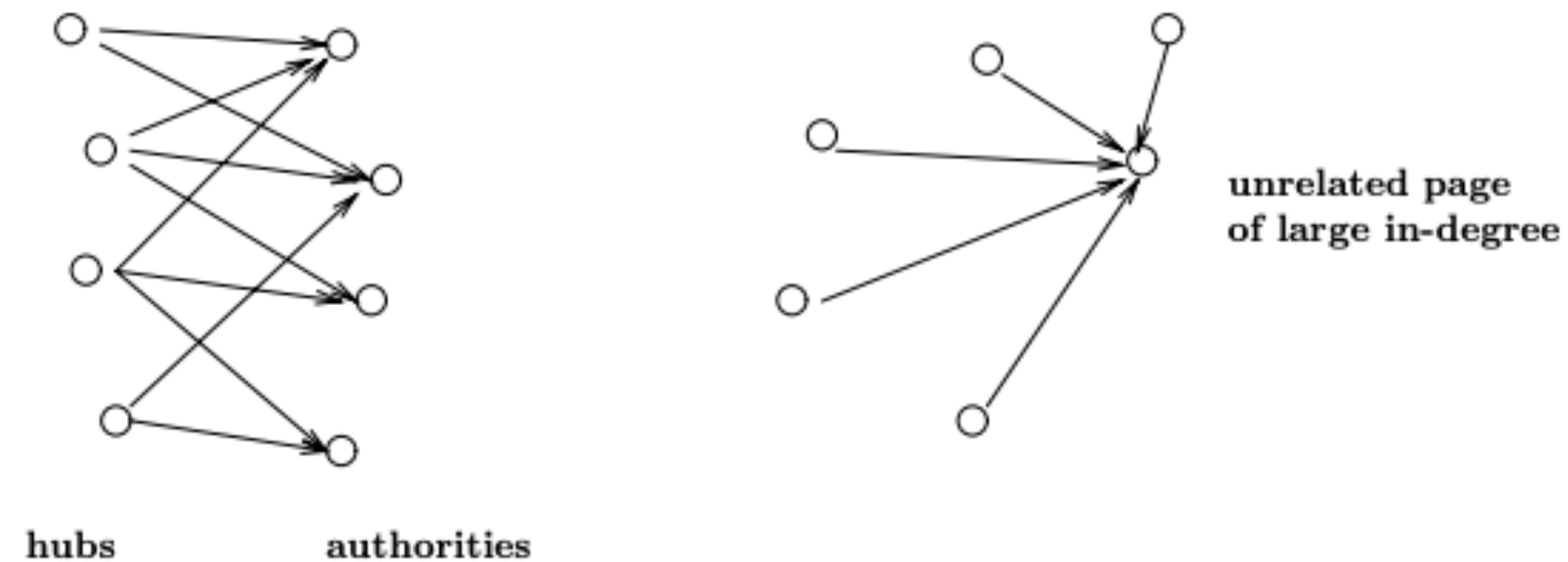


Figure 2: A densely linked set of hubs and authorities.



# Separate core from the composition-root

- >p95 in-Degree = core
- >p95 out-Degree = composition-root
- both = needs rewrite

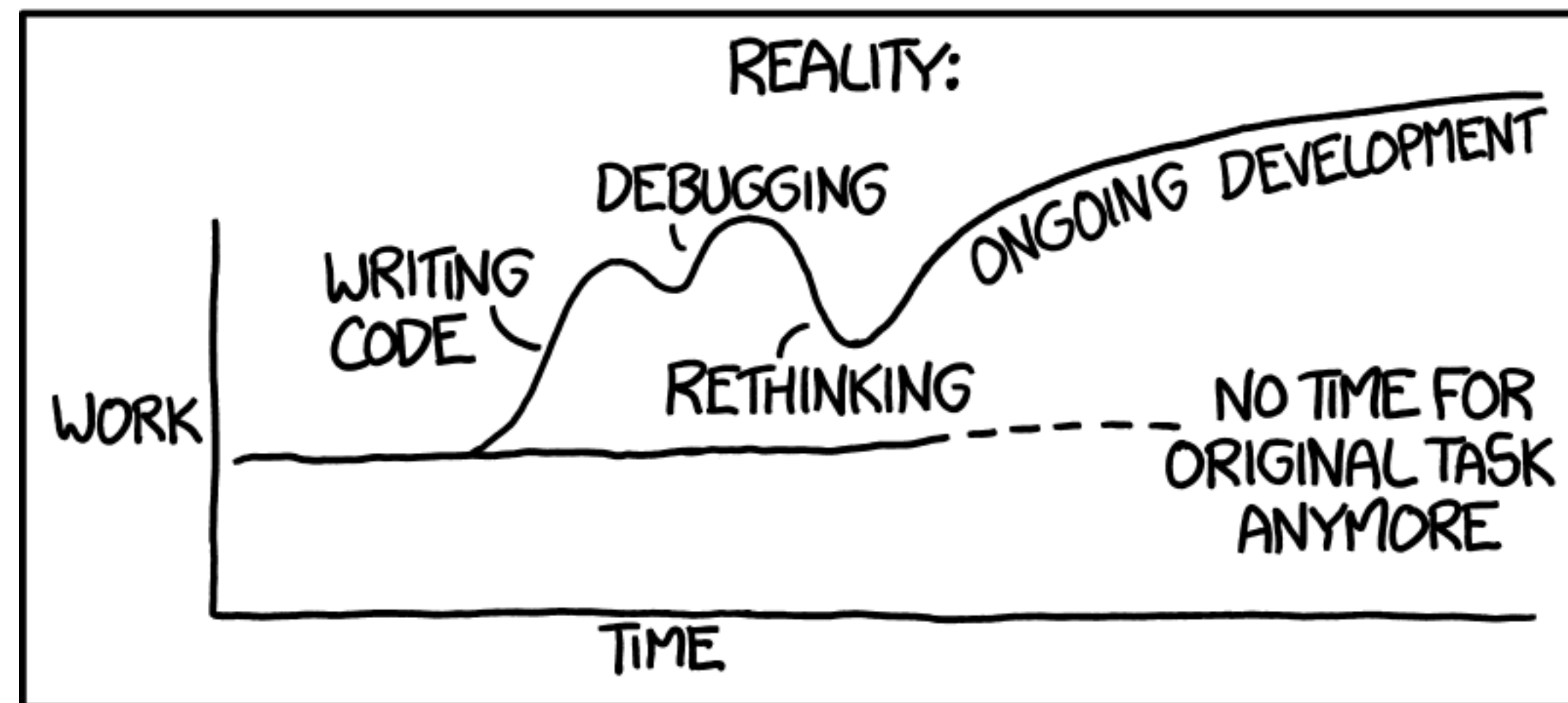
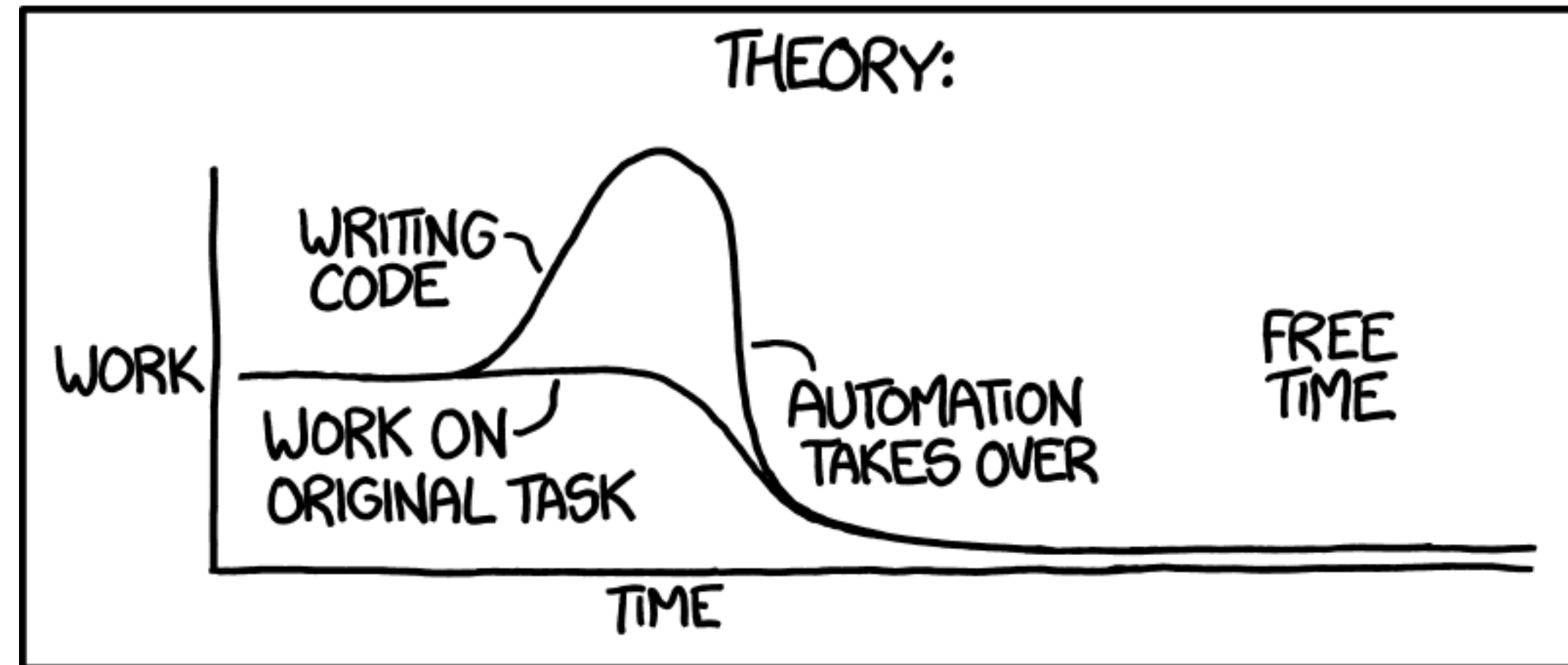
# MVPv2

- Pros:
  - Reduced time to find modules to a minimum
- Cons:
  - Tickets needs to be kept up-to-date



# How to monitor modularisation?

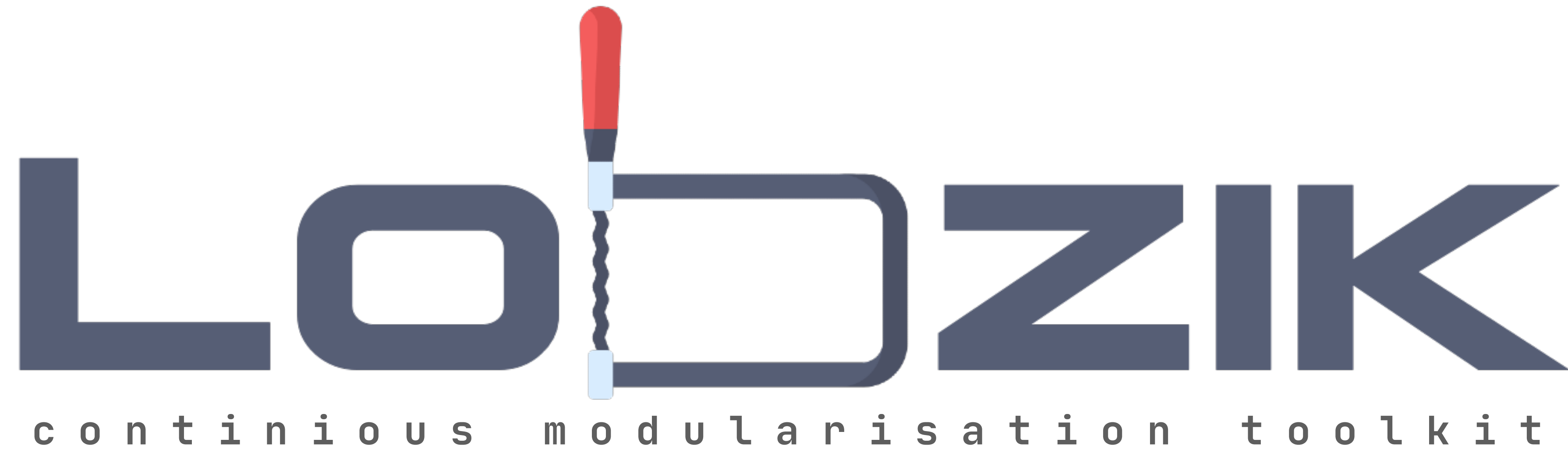
"I SPEND A LOT OF TIME ON THIS TASK.  
I SHOULD WRITE A PROGRAM AUTOMATING IT!"





# Modularization Toolkit

- Vertical modularisation = Authority + Degree
- Horizontal modularisation = Louvain
  - Scoring = Modularity / Conductance





# Lobzik MVP

- Gradle plugin for scanning codebase
- Gephi Toolkit for algorithms and visualisation svg
- HTML report with modularisation analysis

# Naming problem

ActionFromNotification	ExtKt	CallsFragmentEndOfOutgoingBindi
App2AppAgent	FloatingViewModel	CallsFragmentEndOfTalkingBinding
AudioSource	FloatingWindow	CallsFragmentIncomingBinding
Authorization	FullScreenVideo	CallsFragmentNeedPermissionBind
AuthorizationMachineKt	HistoryCallEffHandler	CallsFragmentTalkingOutgoingBind
AuthRequest	IncomingCallFragment	CallsHiddenCallBinding
BackgroundViewModel	JsonToCallEventConverterKt	CallsLayoutBckgBinding
BaseVideoFragment	LastCallInfo	CallsLayoutPhotoWindowBinding
BaseViewBindingFragment	LoginEvent	CallsMutedBlockBinding
CallAction	LogInResult	CallsNavigationEffectHandler
CallActivity	MutedBlock	CallsSelfVideoBinding
CallActivityArgs	MutedEntity	CallsVideoView
CallData	OneTimeTokenRequest	CallViewModel
CallDataKt	OutgoingCallFragment	CallViewModelKt



# Naming problem

ActionFromNotification

App2AppAgent

AudioSource

Authorization

AuthorizationMachineKt

AuthRequest

BackgroundViewModel

BaseVideoFragment

BaseViewBindingFragment

CallAction

CallActivity

CallActivityArgs

CallData

CallDataKt

ExtKt

FloatingViewModel

FloatingWindow

FullScreenVideo

HistoryCallEffHandler

IncomingCallFragment

JsonToCallEventConverterKt

LastCallInfo

LoginEvent

LogInResult

MutedBlock

MutedEntity

OneTimeTokenRequest

OutgoingCallFragment

CallsFragmentEndOfOutgoingBinding

CallsFragmentEndOfTalkingBinding

CallsFragmentIncomingBinding

CallsFragmentNeedPermissionBinding

CallsFragmentTalkingOutgoingBinding

CallsHiddenCallBinding

CallsLayoutBckgBinding

CallsLayoutPhotoWindowBinding

CallsMutedBlockBinding

CallsNavigationEffectHandler

CallsSelfVideoBinding

CallsVideoView

CallViewModel

CallViewModelKt

# TF-IDF summarisation

$$w_{x,y} = \text{tf}_{x,y} \times \log \left( \frac{N}{\text{df}_x} \right)$$

**TF-IDF**

Term  $x$  within document  $y$

$\text{tf}_{x,y}$  = frequency of  $x$  in  $y$

$\text{df}_x$  = number of documents containing  $x$

$N$  = total number of documents



# v1.0

- Pros:
  - Automatically scans project
  - Detects potential modules
  - Scores existing modules
- Cons:
  - ~~doesn't make coffee~~
  - Still needs an engineer to validate findings



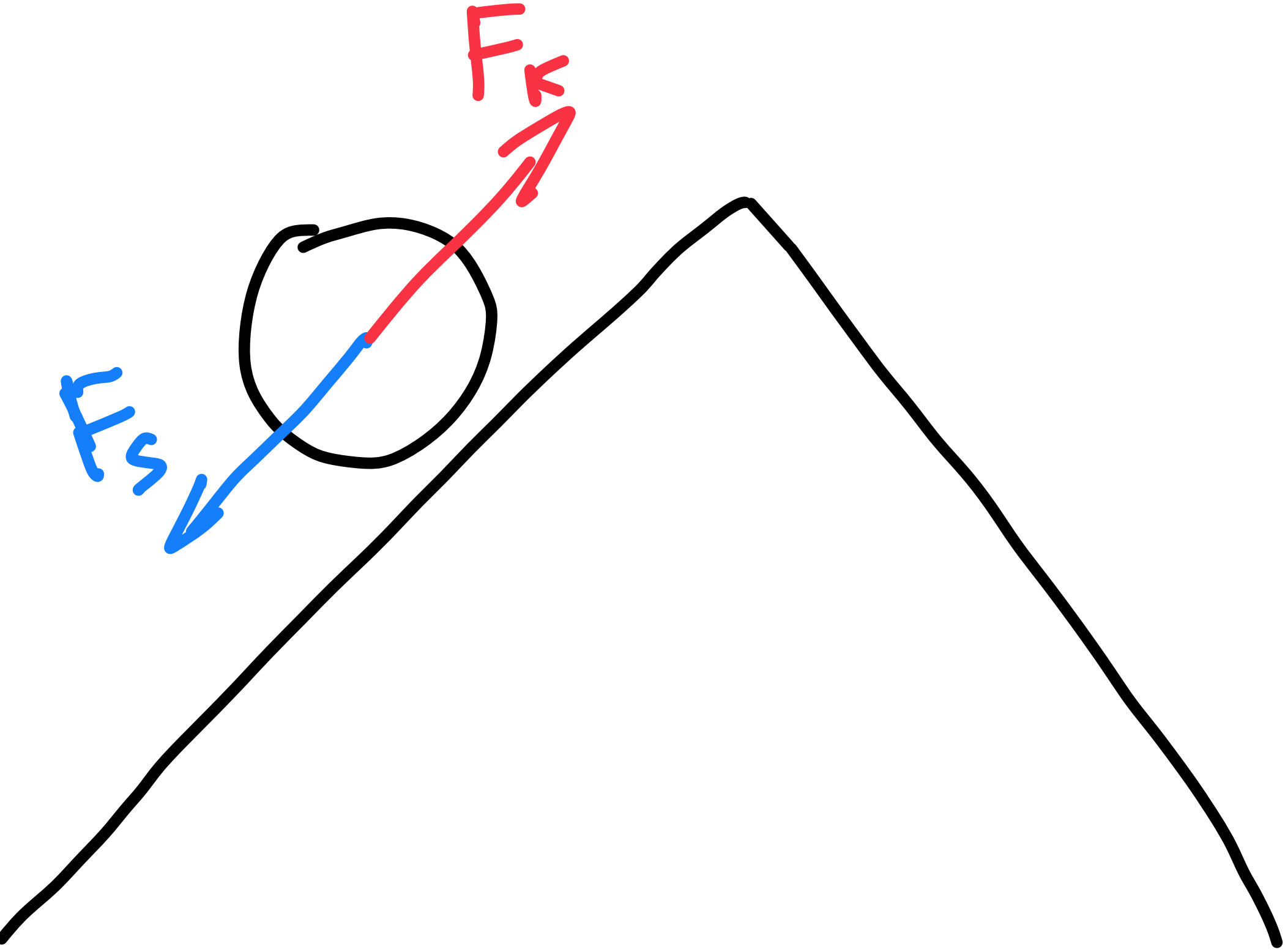
**Stepan Goncharov**  
Grab

## **Absolute modularization**

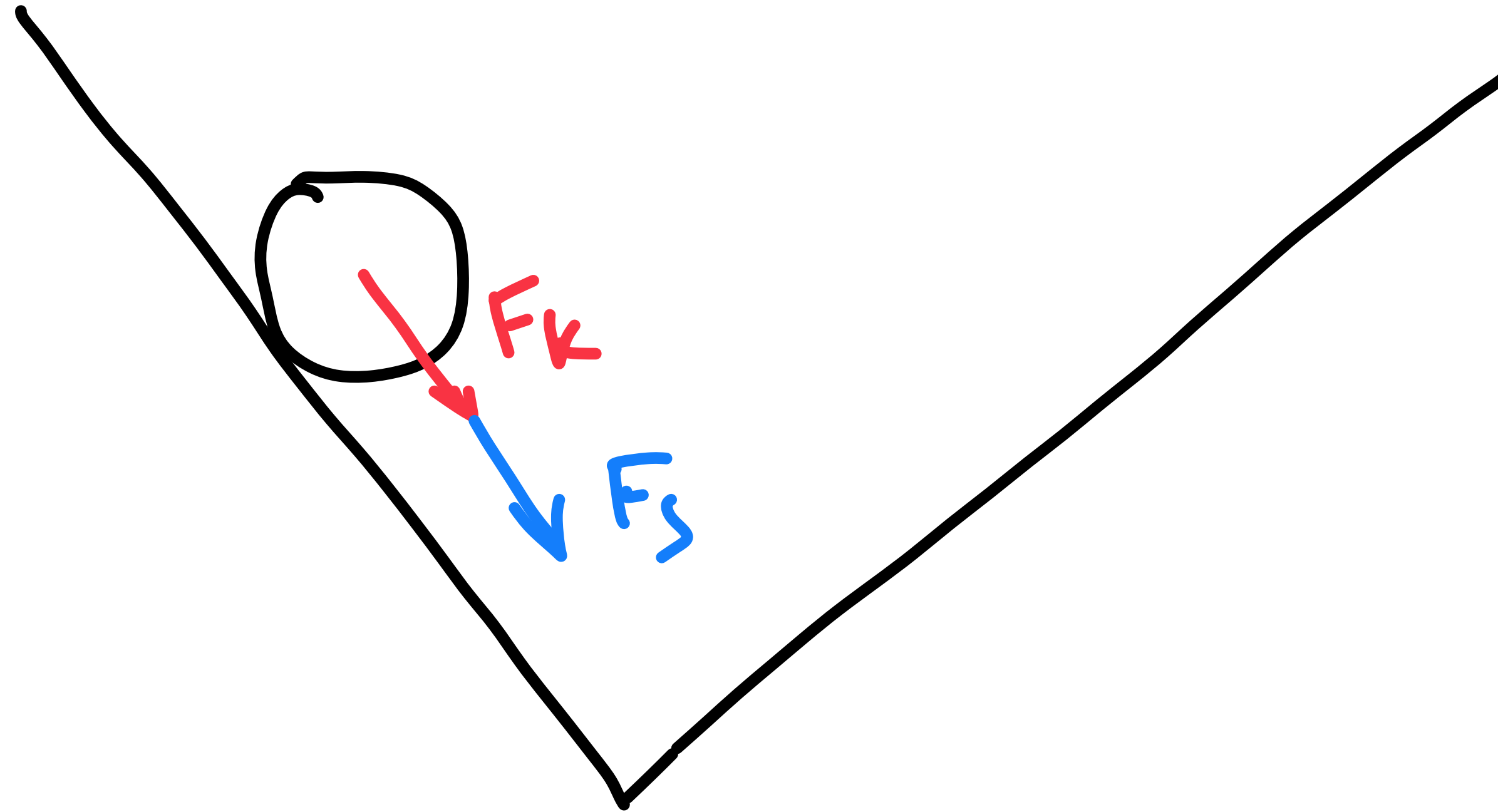




# Mountain of Success



# Pit of Success





# Continuous Modularisation

# Demo



# Try it yourself

- Grab Lobzik at [github.com/Mishkun/lobzik](https://github.com/Mishkun/lobzik)
- Read my blog @izpodshtorki
- Follow me on [twitter.com/TheMishkun](https://twitter.com/TheMishkun)
- Ask questions!

