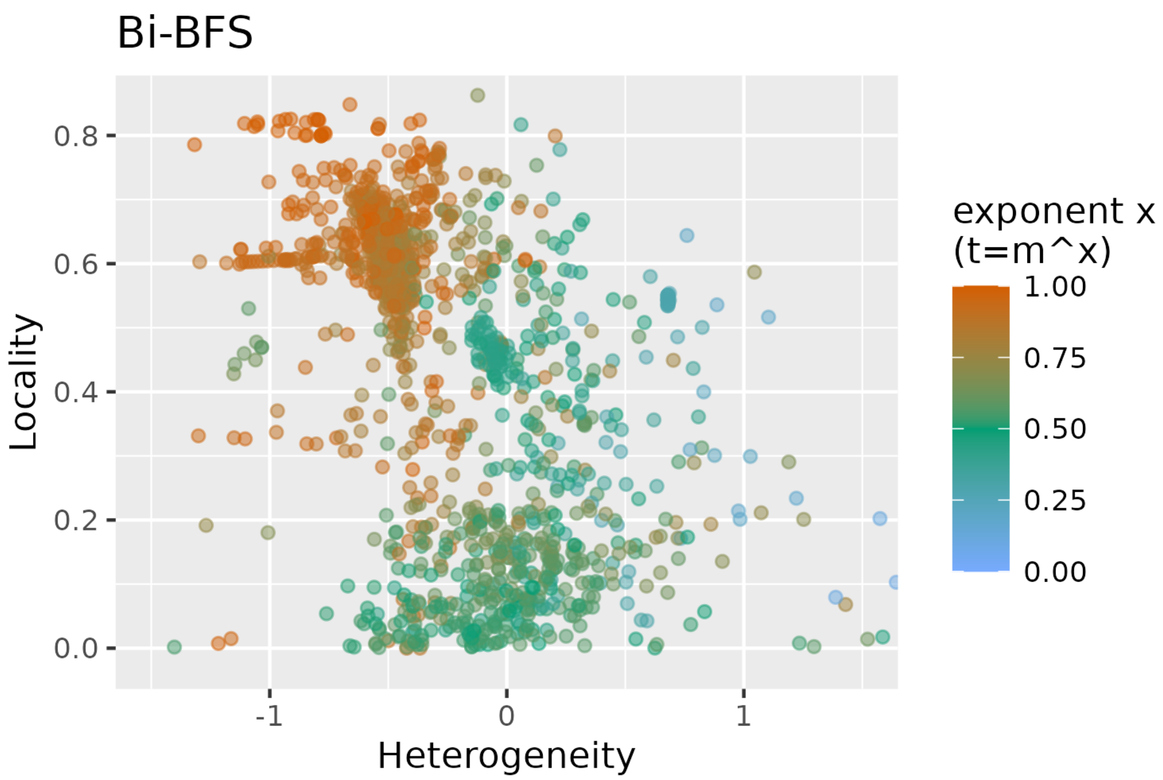
Beating the Worst Case

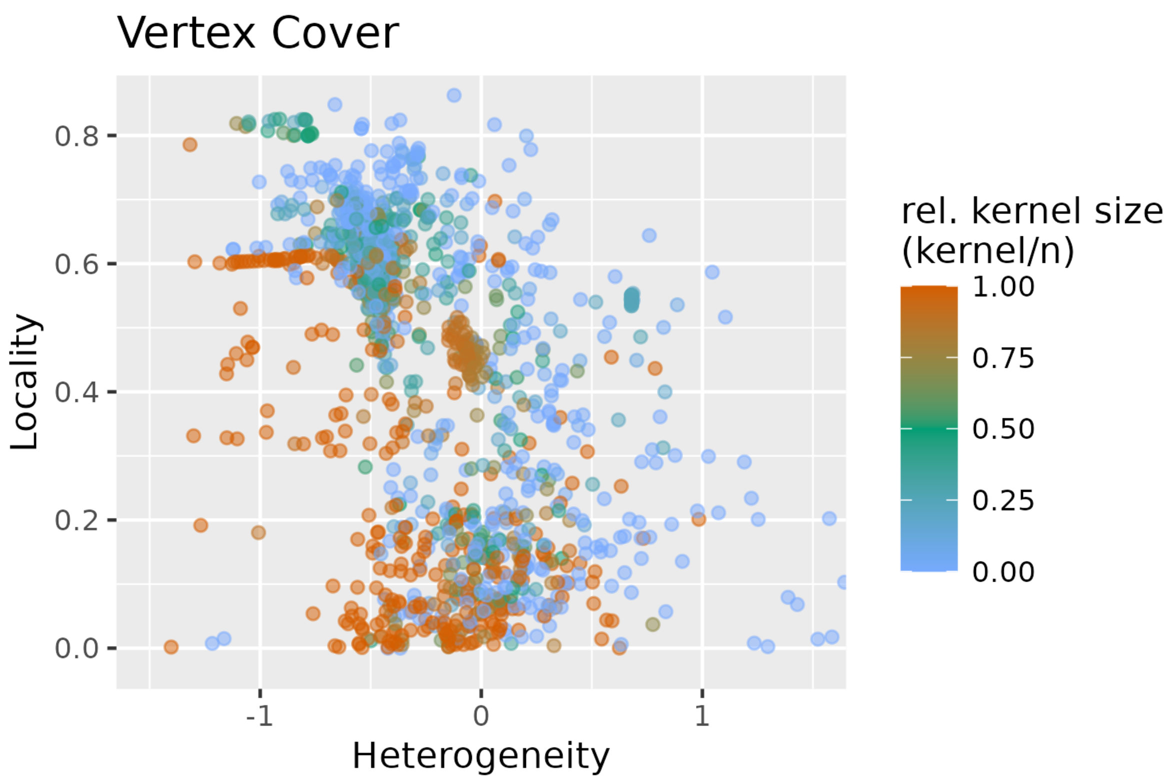
Practical Course – 7th meeting

Jean-Pierre, Marcus

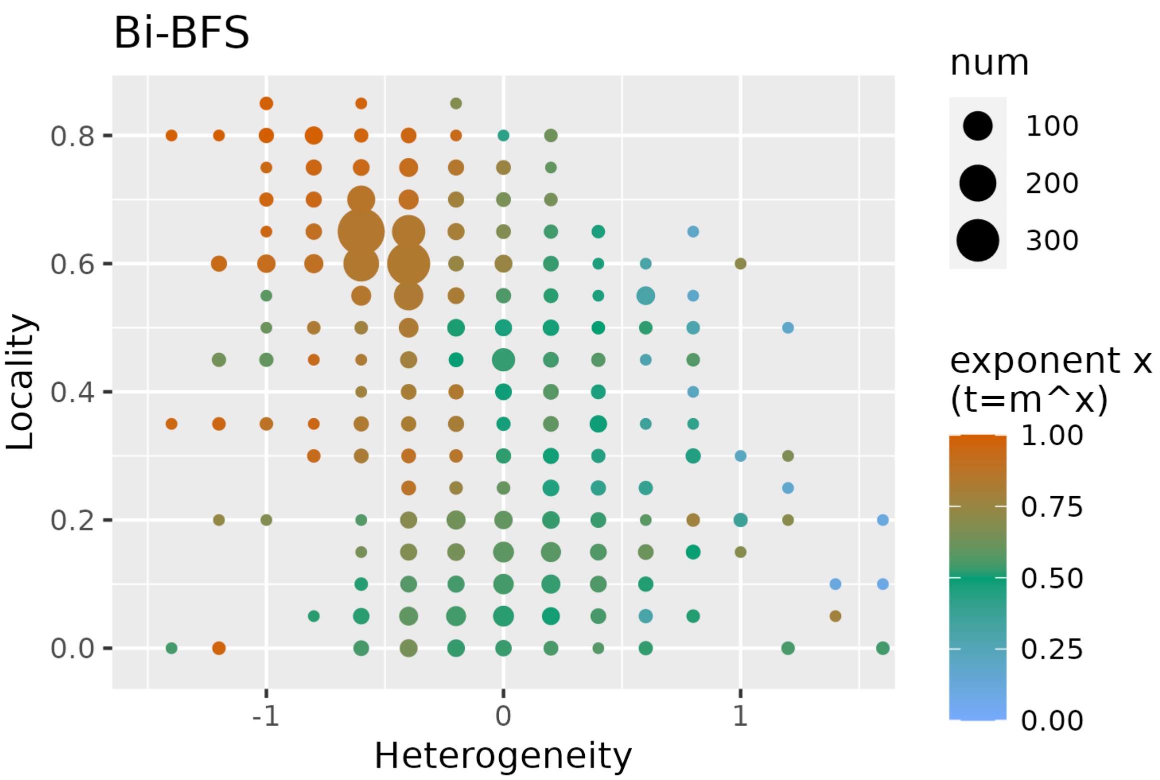
Exercise Sheet 3

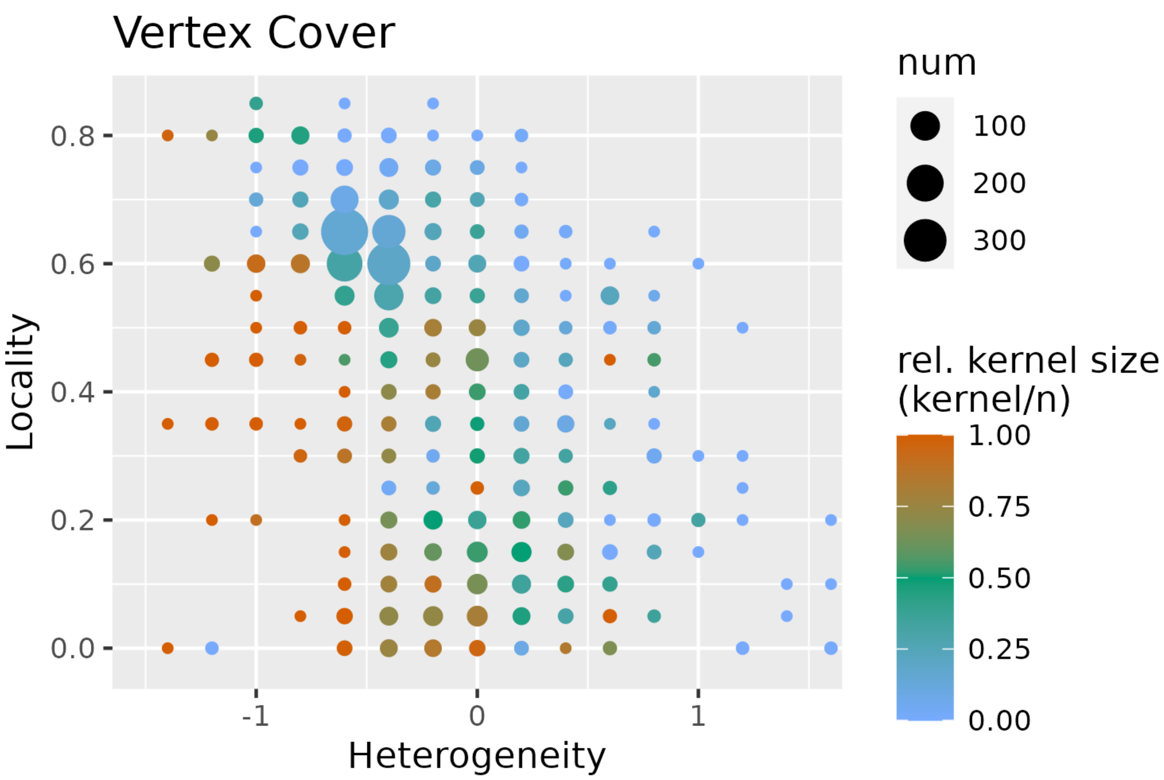
Solution Exercise Sheet 3



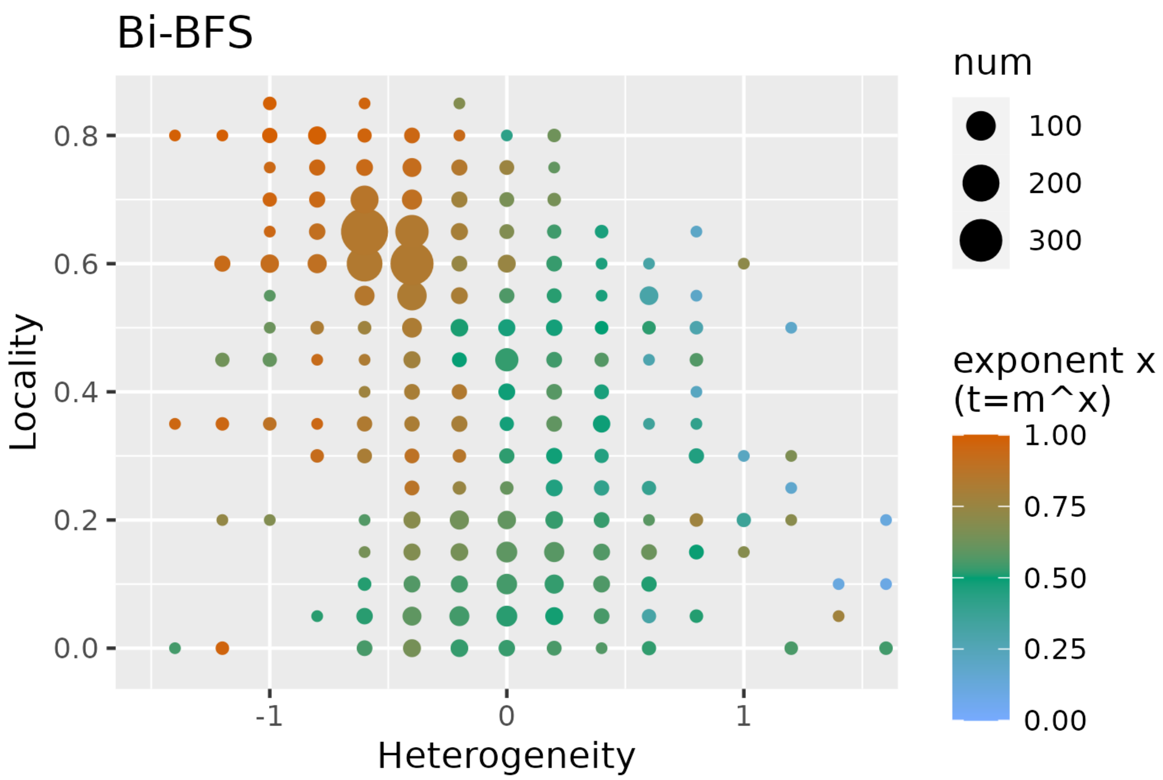


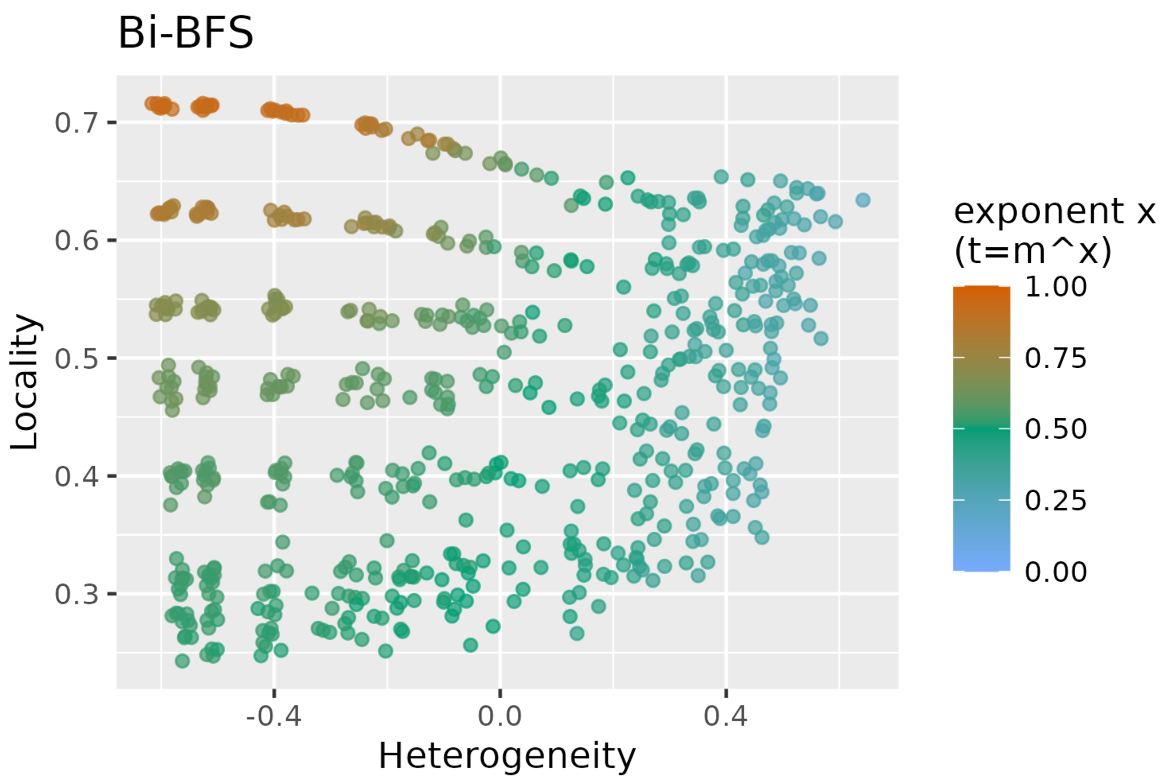
Solution Exercise Sheet 3



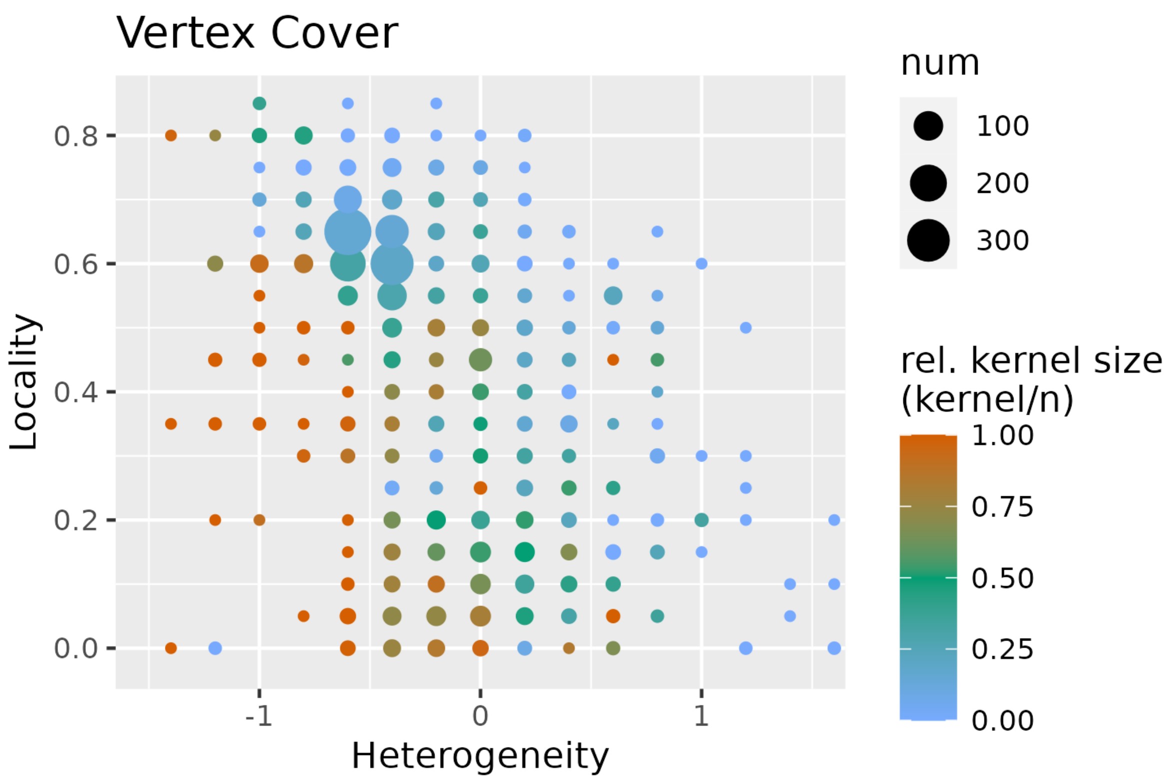


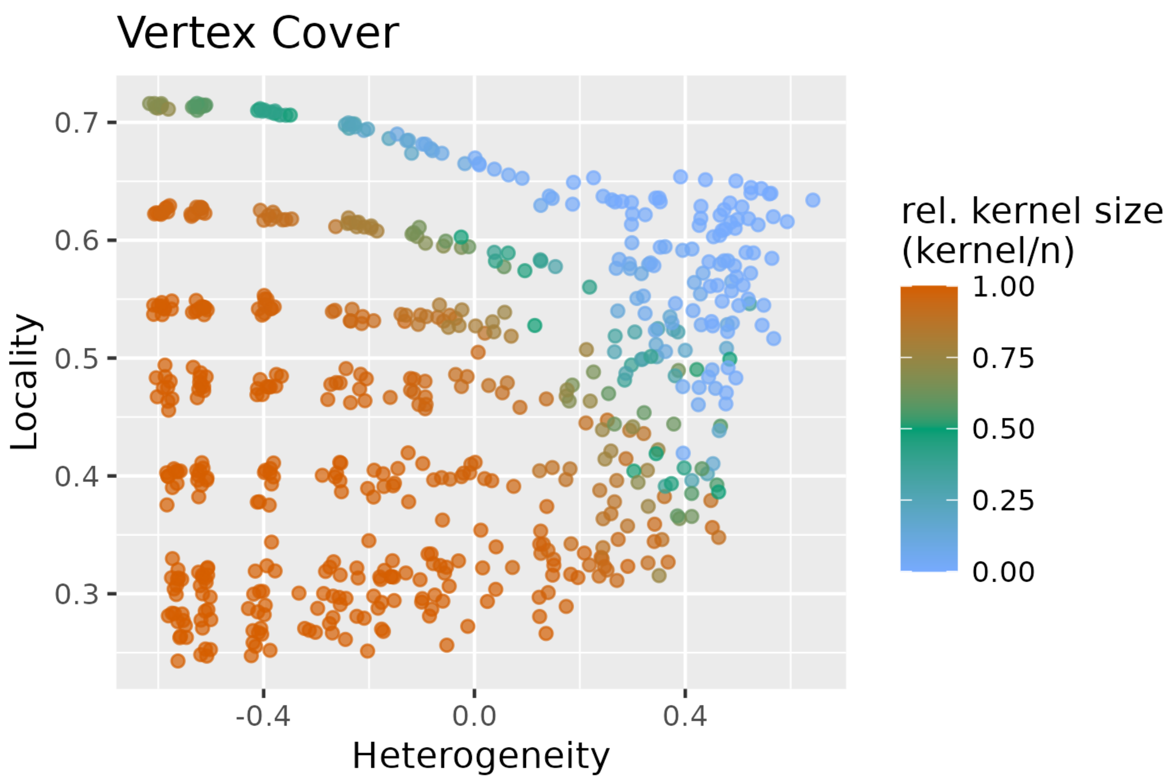
Solution Exercise Sheet 3



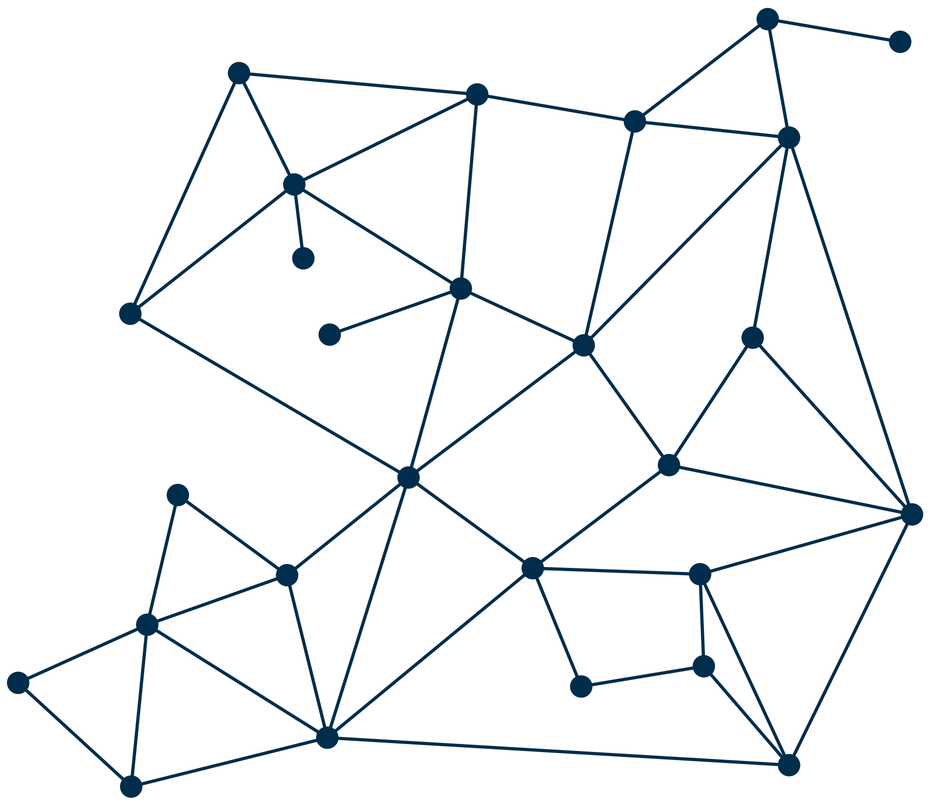


Solution Exercise Sheet 3

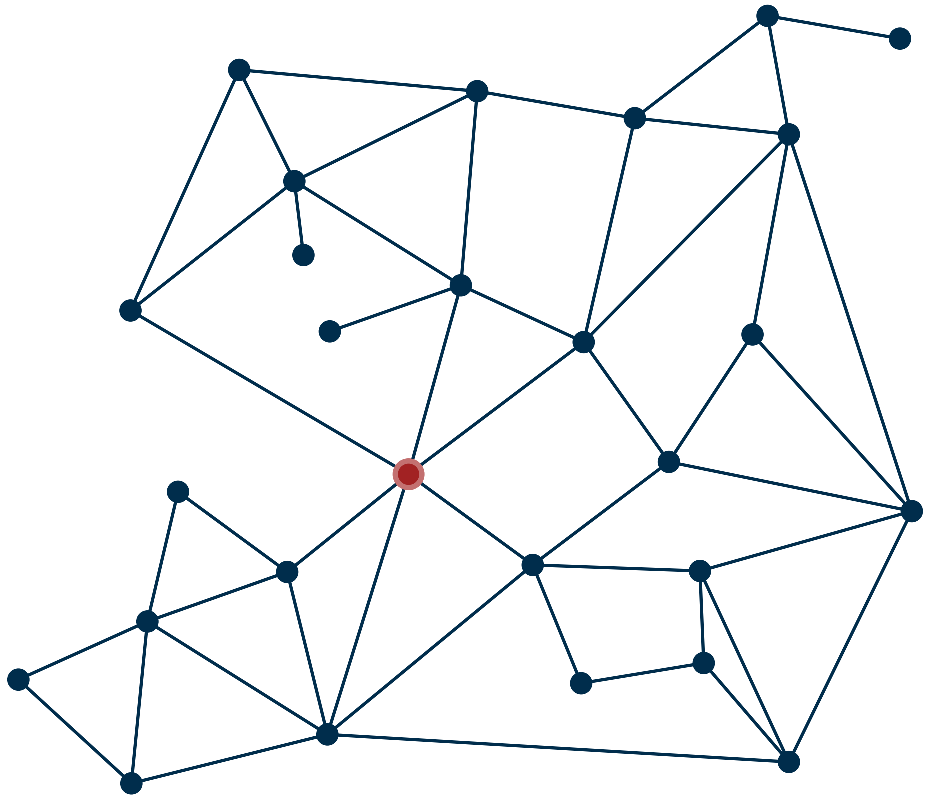




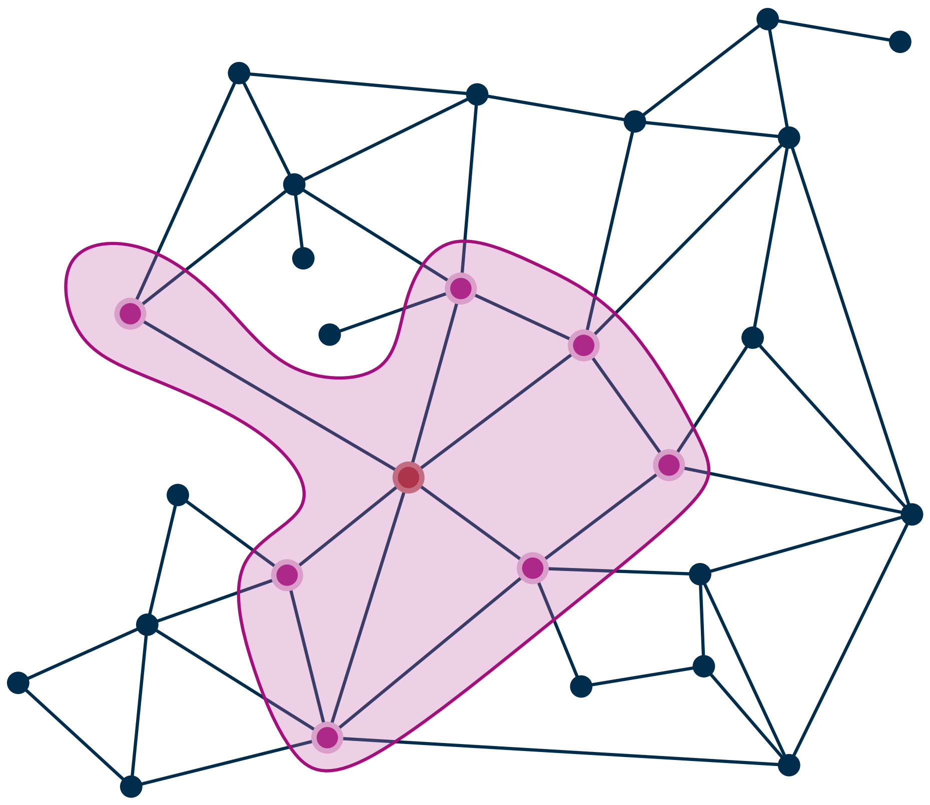
Eccentricity and Diameter



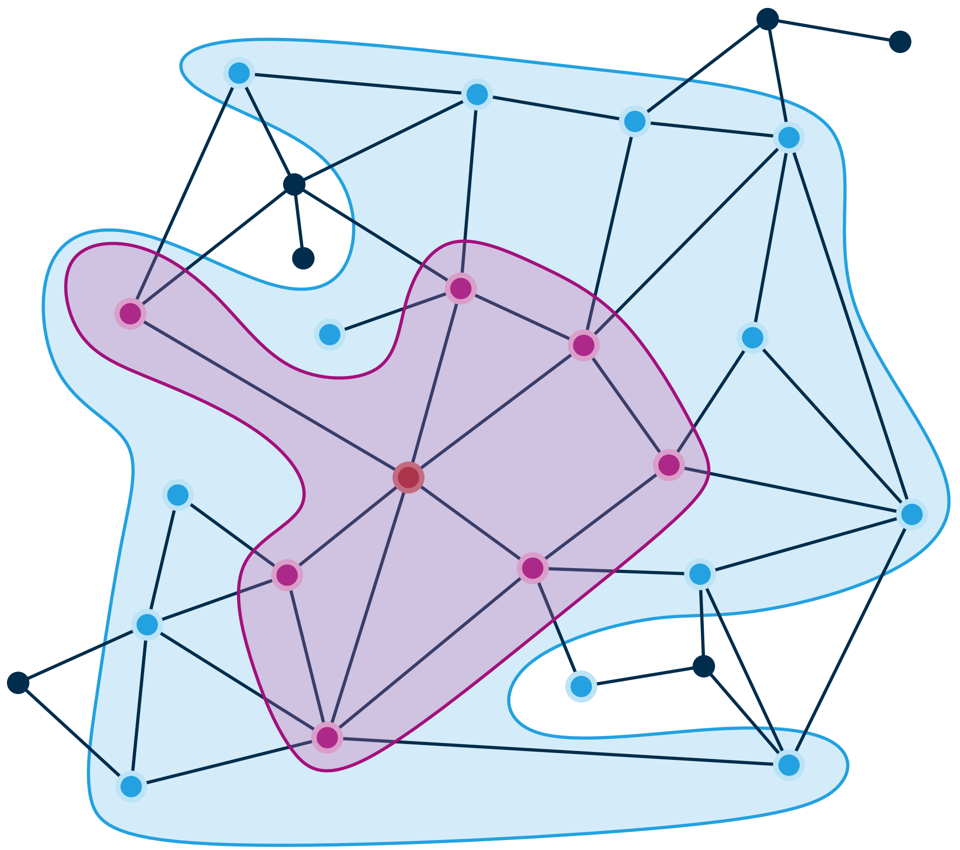
Eccentricity and Diameter



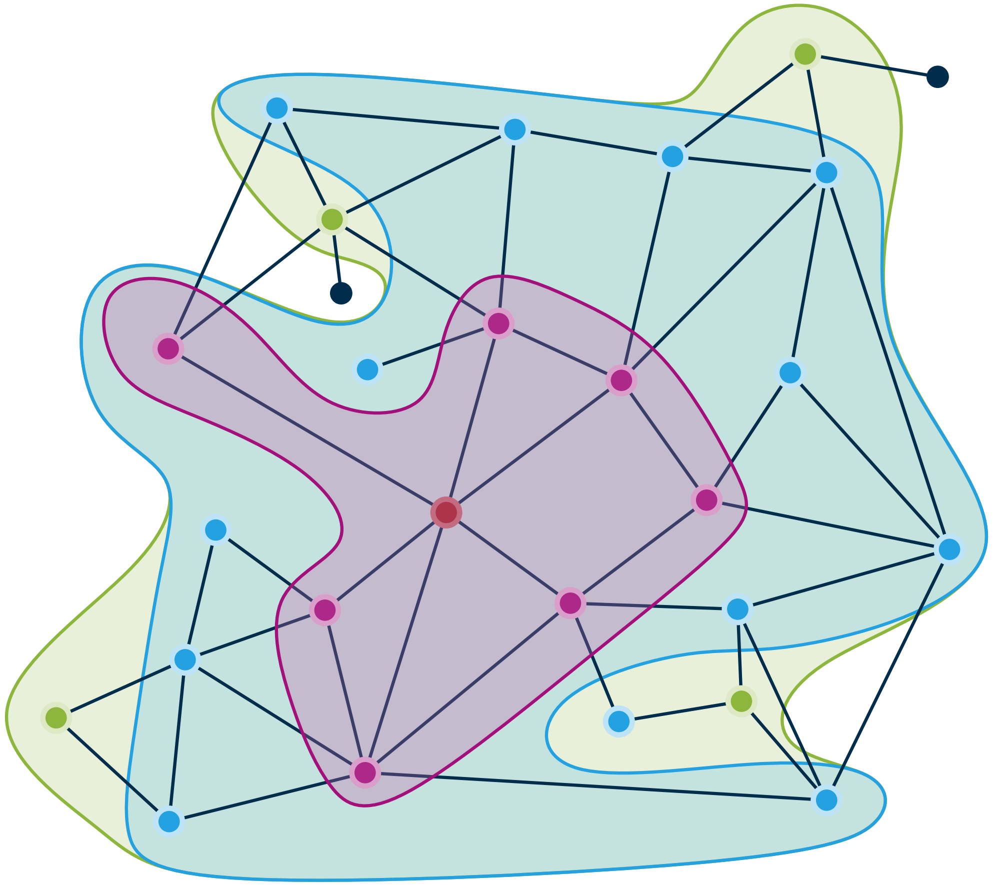
Eccentricity and Diameter



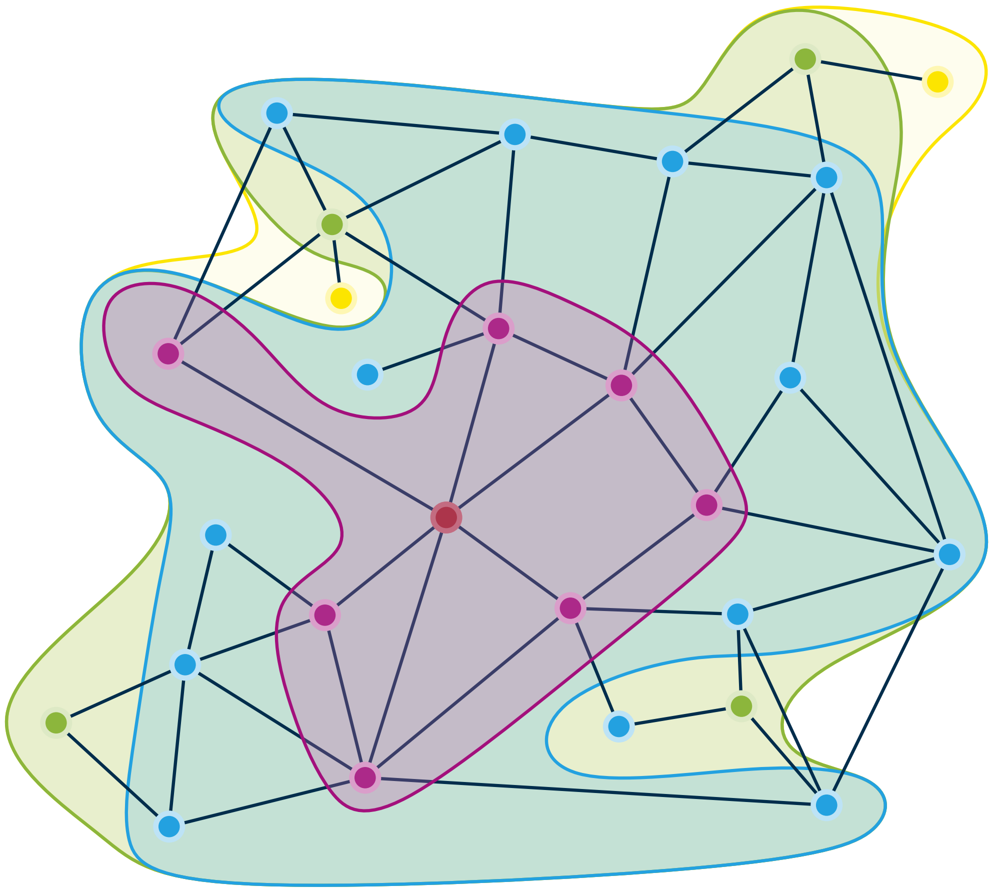
Eccentricity and Diameter



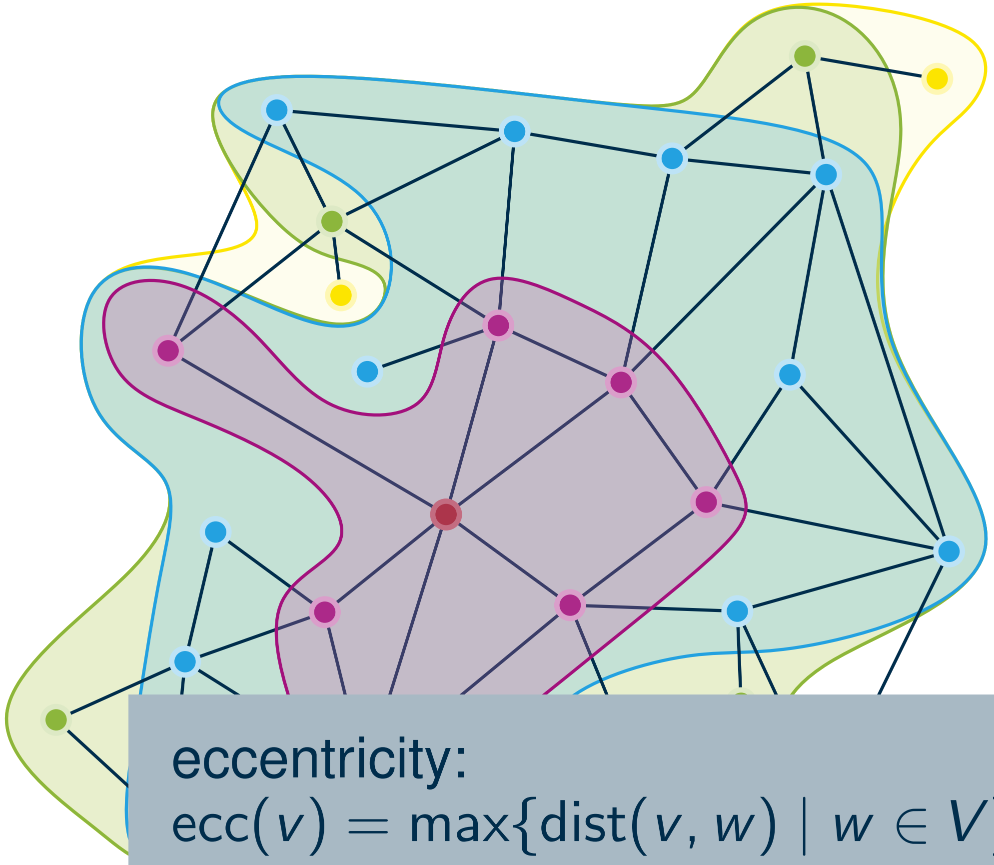
Eccentricity and Diameter



Eccentricity and Diameter

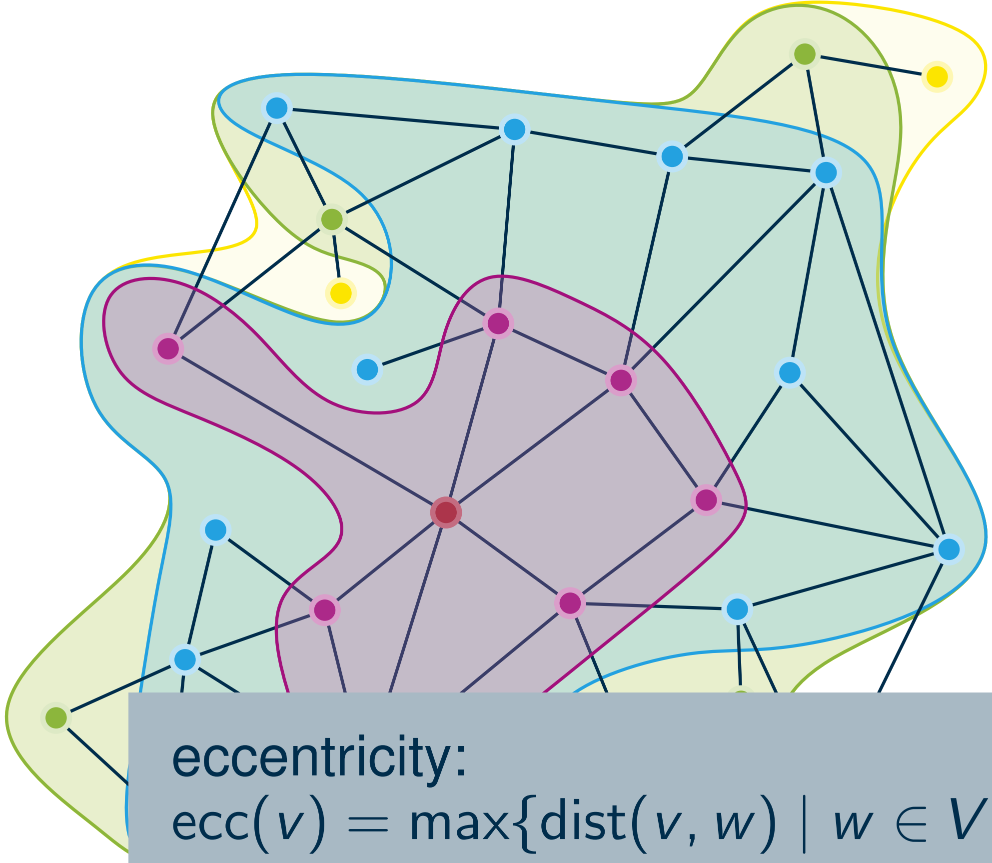


Eccentricity and Diameter



eccentricity:
$\mathrm{ecc}(v) = \max\{\mathrm{dist}(v, w) \mid w \in V\}$

Eccentricity and Diameter

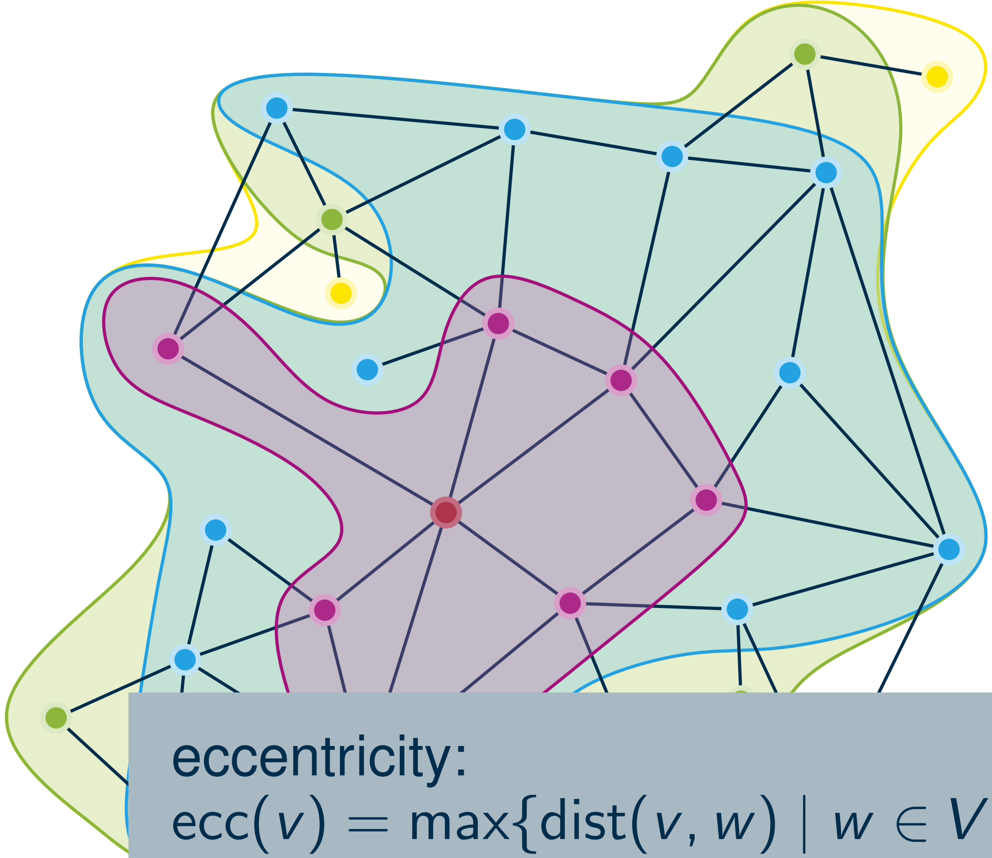


eccentricity:
$\mathrm{ecc}(v) = \max\{\mathrm{dist}(v, w) \mid w \in V\}$

diameter:
$\mathrm{diam}(G) = \max\{\mathrm{ecc}(v) \mid v \in V\}$

Eccentricity and Diameter

How to compute the diameter of a graph?



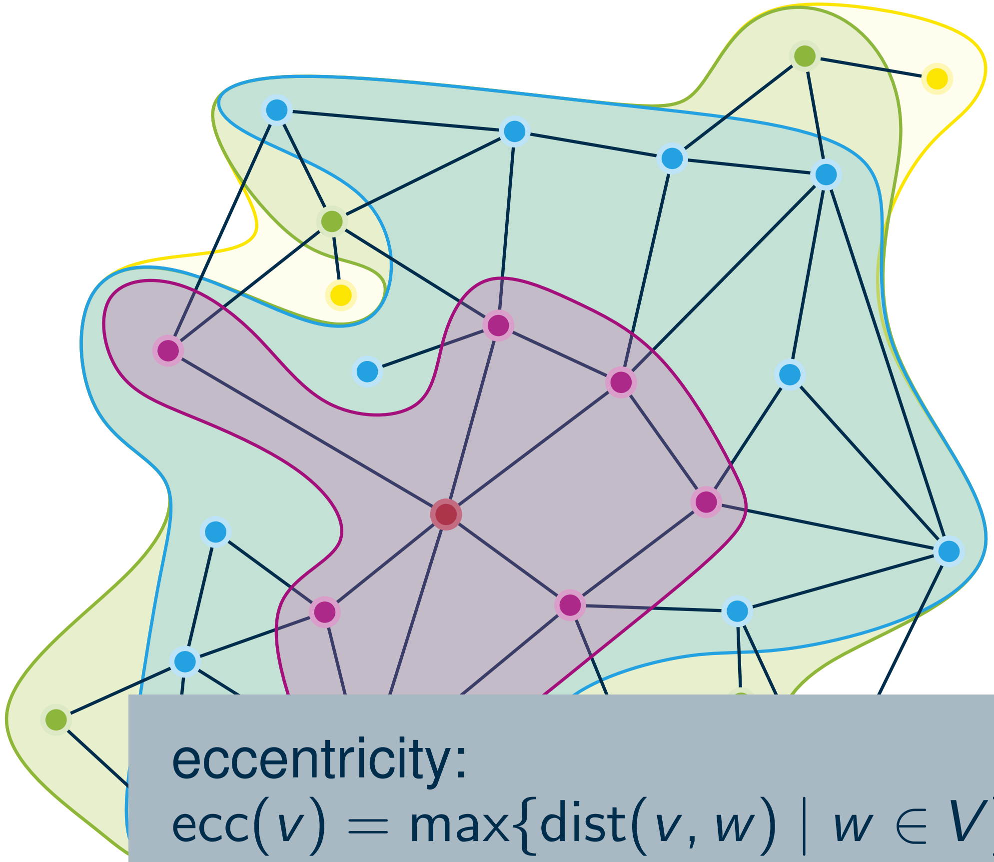
eccentricity:
$\mathrm{ecc}(v) = \max\{\mathrm{dist}(v, w) \mid w \in V\}$

diameter:
$\mathrm{diam}(G) = \max\{\mathrm{ecc}(v) \mid v \in V\}$

Eccentricity and Diameter

How to compute the diameter of a graph?

- naive: n times BFS



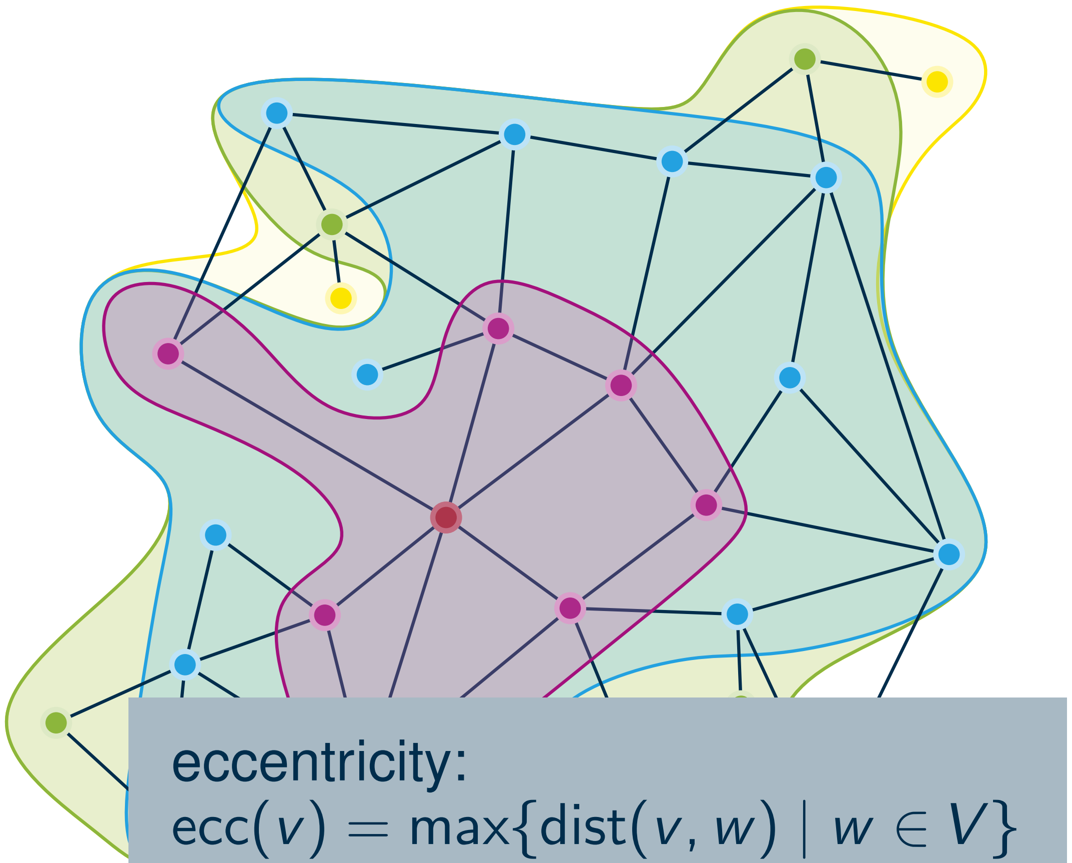
eccentricity:
$\mathrm{ecc}(v) = \max\{\mathrm{dist}(v, w) \mid w \in V\}$

diameter:
$\mathrm{diam}(G) = \max\{\mathrm{ecc}(v) \mid v \in V\}$

Eccentricity and Diameter

How to compute the diameter of a graph?

- naive: n times BFS
- can we do better?



eccentricity:
$\mathrm{ecc}(v) = \max\{\mathrm{dist}(v, w) \mid w \in V\}$

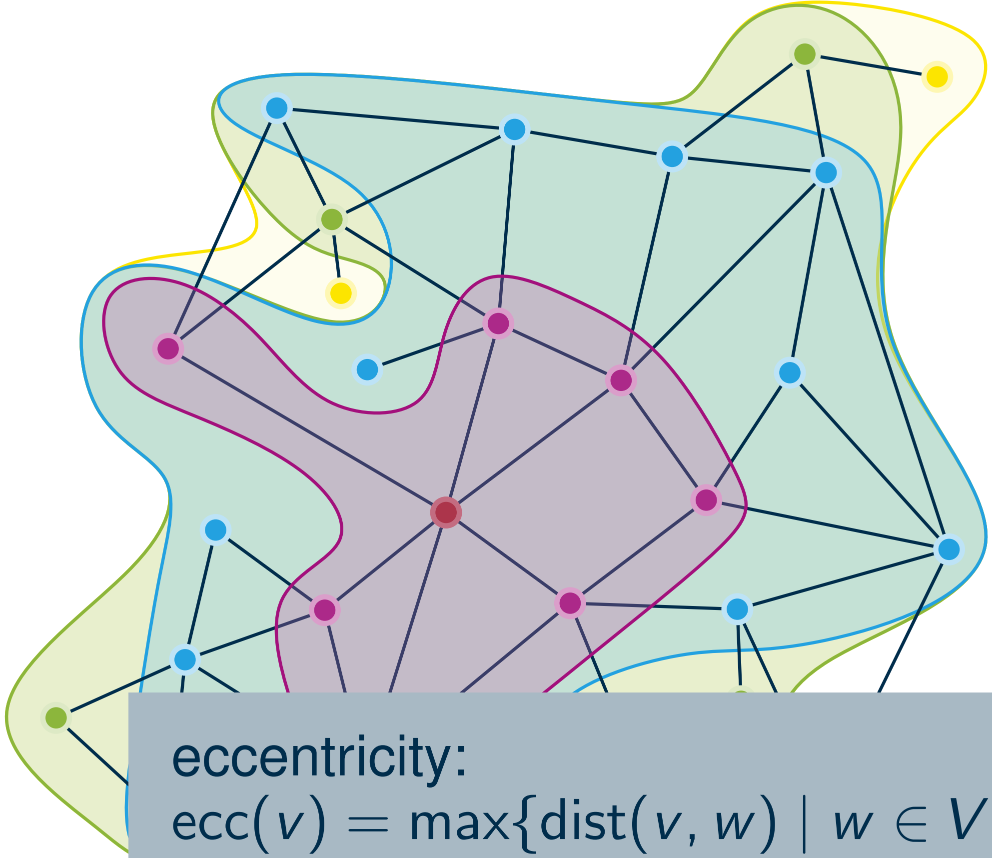
diameter:
$\mathrm{diam}(G) = \max\{\mathrm{ecc}(v) \mid v \in V\}$

Eccentricity and Diameter

How to compute the diameter of a graph?

- naive: n times BFS
- can we do better?
  - no: (probably)

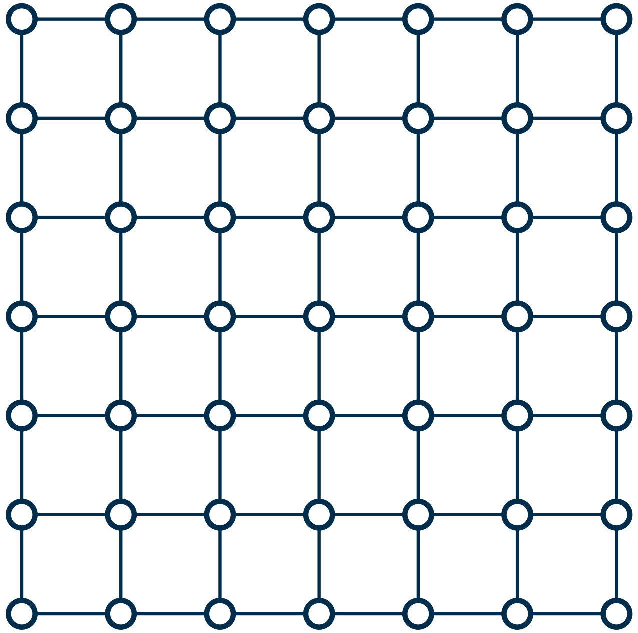
> “Any $O(m^{2-\varepsilon})$ time algorithm that distinguishes whether the diameter of a given undirected unweighted graph is 2 or at least 3 would imply an improved CNF-SAT algorithm. [Liam, Williams – STOC’13] ”

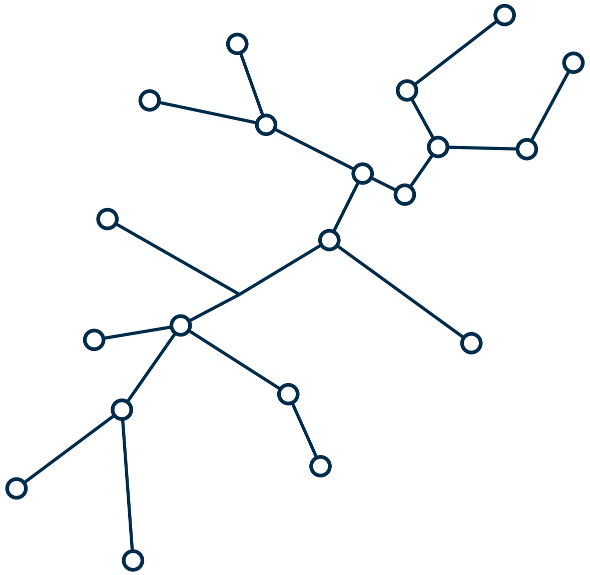


eccentricity:
$\mathrm{ecc}(v) = \max\{\mathrm{dist}(v, w) \mid w \in V\}$

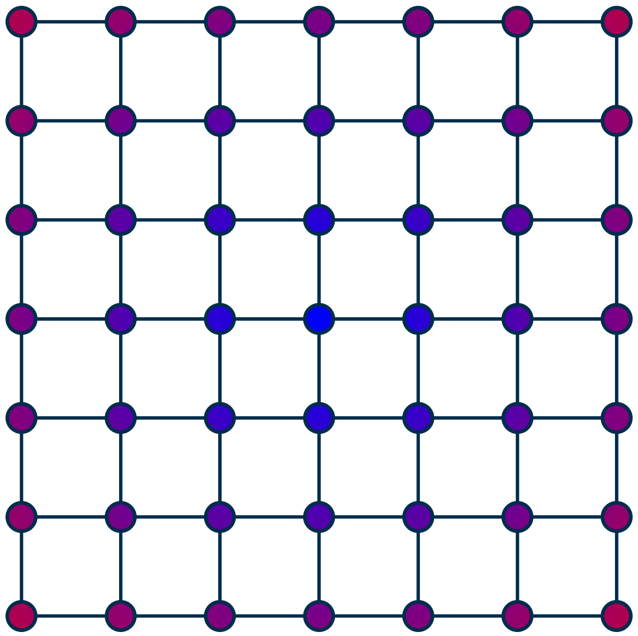
diameter:
$\mathrm{diam}(G) = \max\{\mathrm{ecc}(v) \mid v \in V\}$

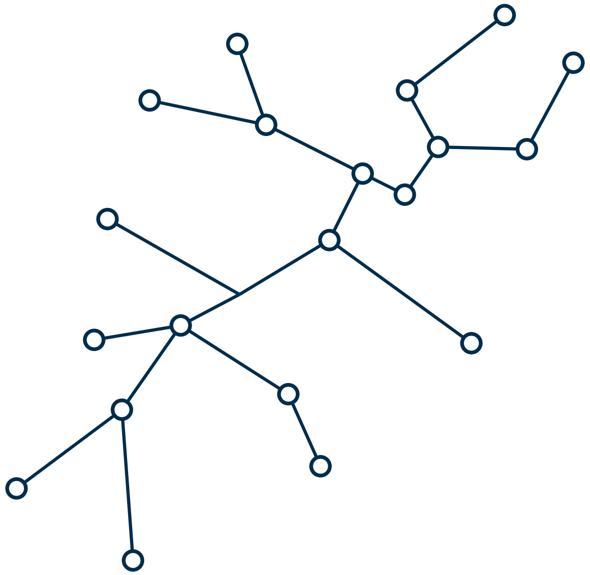
Easy inputs?



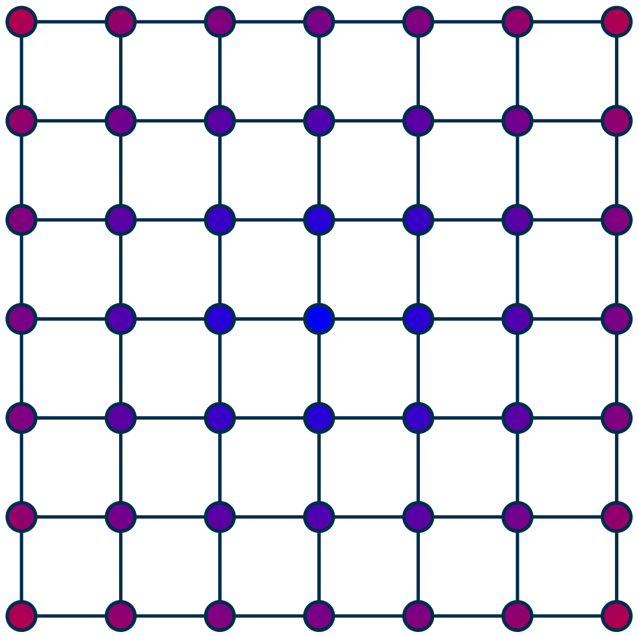


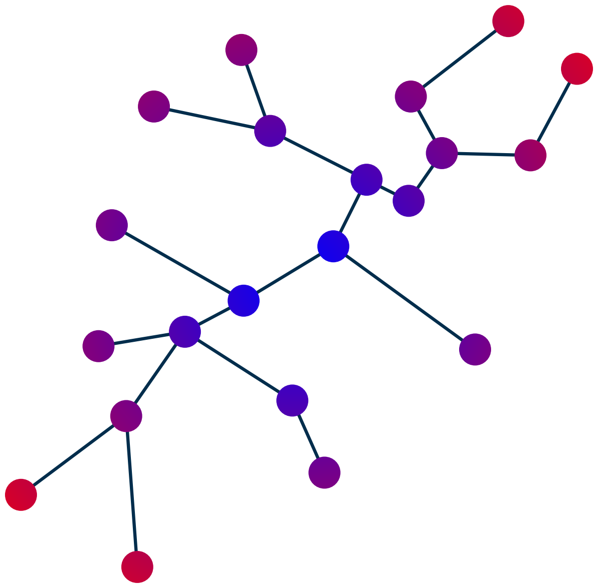
Easy inputs?



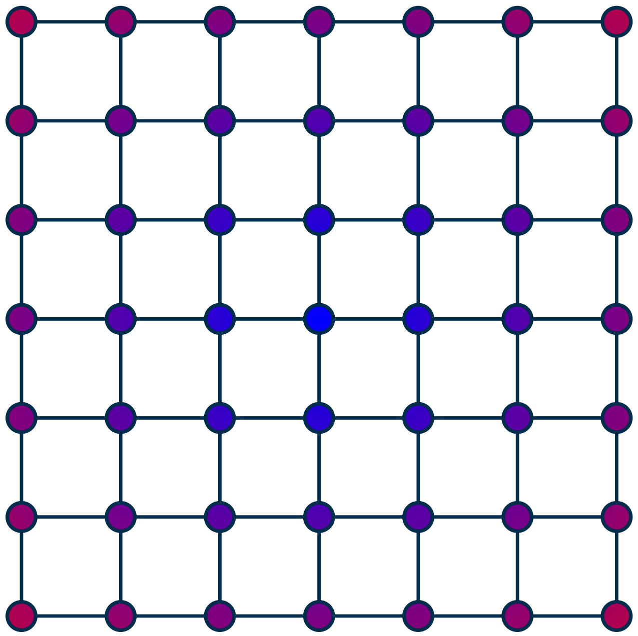


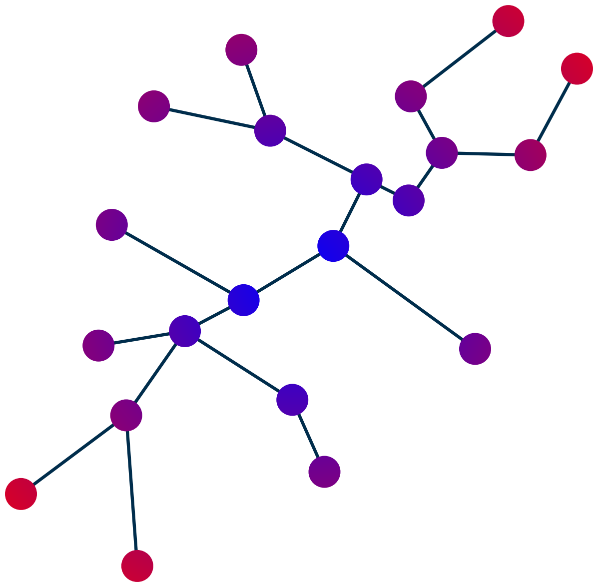
Easy inputs?





Easy inputs?





Idea: only run BFS from peripheral vertices

iFUB algorithm

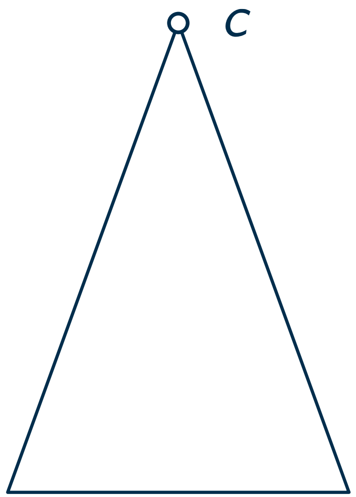
(iterative fringe upper bound, [Crescenzi et al. 2013])

iFUB Algorithm

- select *central* vertex c

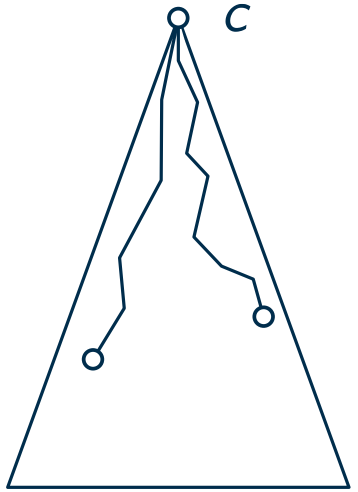
iFUB Algorithm

- select *central* vertex c



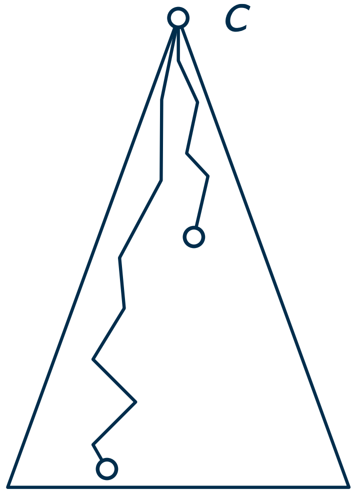
iFUB Algorithm

- select *central* vertex c



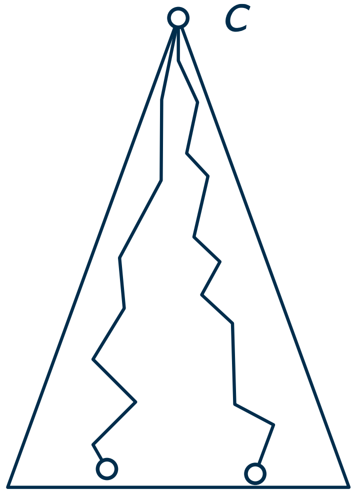
iFUB Algorithm

- select *central* vertex c



iFUB Algorithm

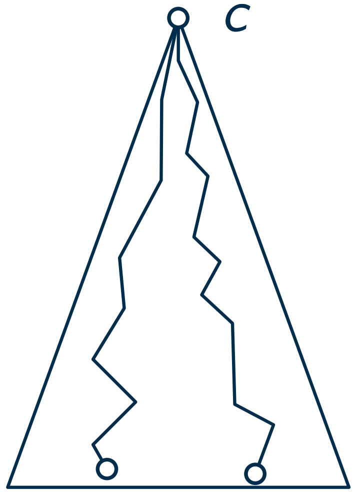
- select *central* vertex c



iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

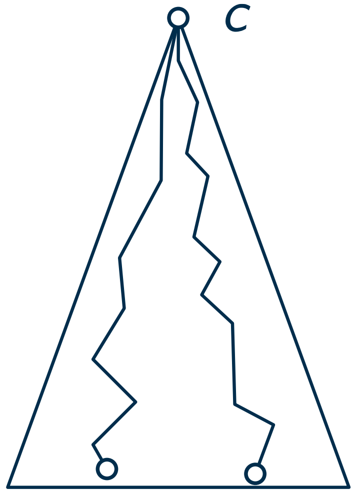


iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w

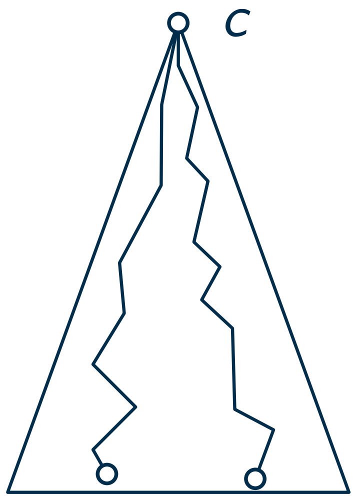


iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c

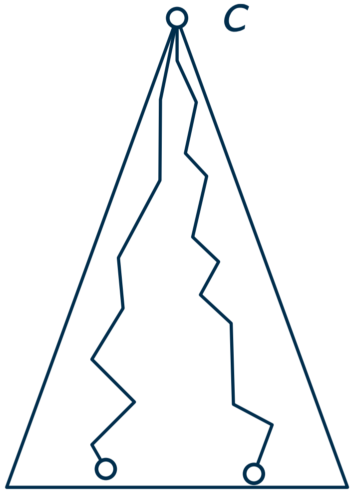


iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:

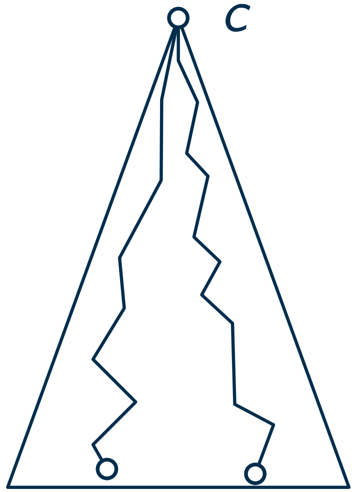


iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter

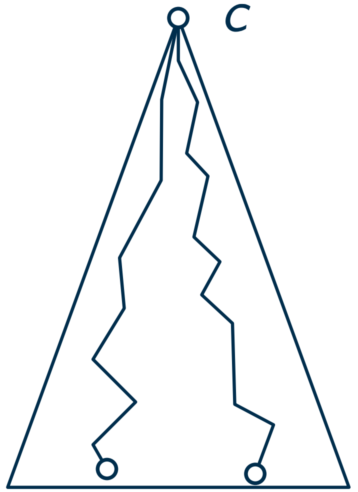


iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$

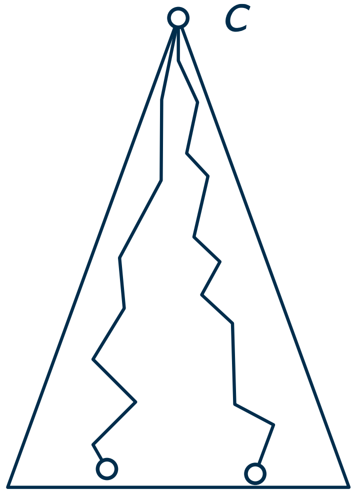


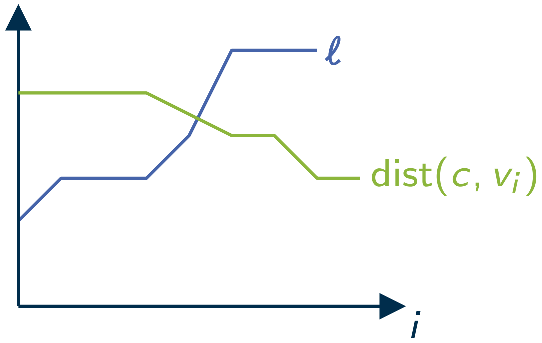
iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

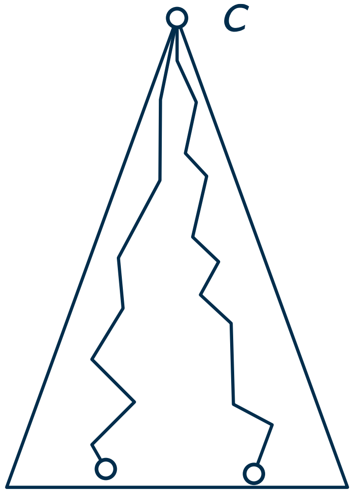
- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$





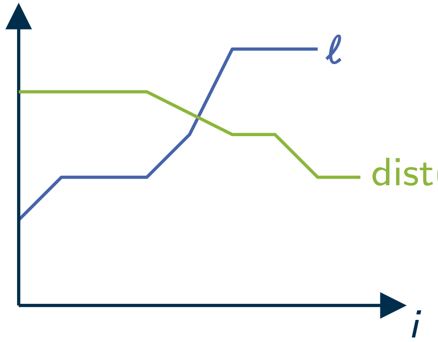
iFUB Algorithm

- select *central* vertex c



Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$

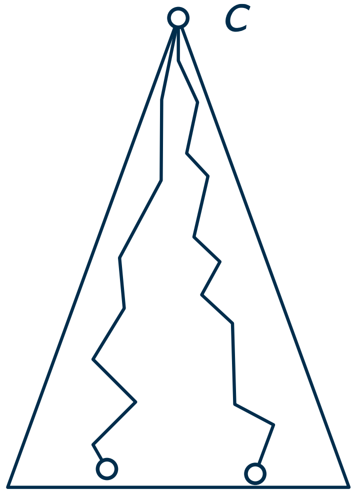


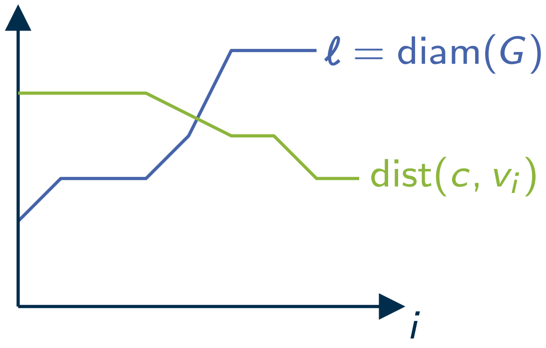
iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



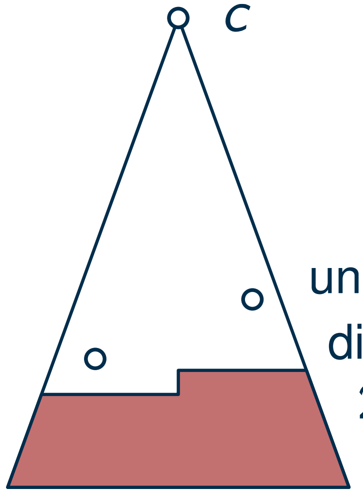


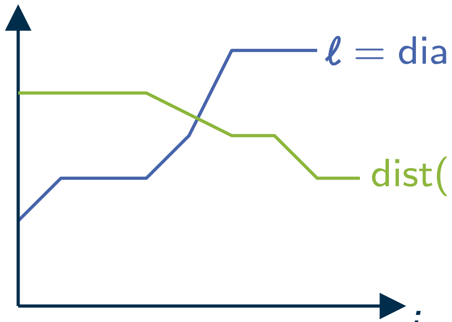
iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



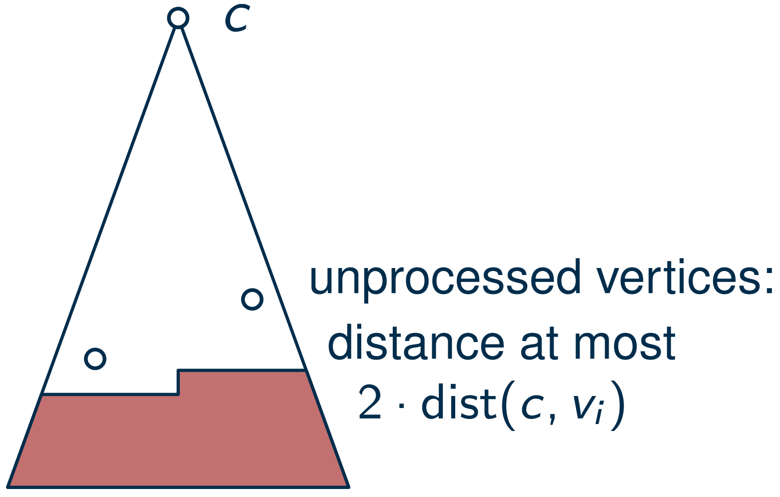


iFUB Algorithm

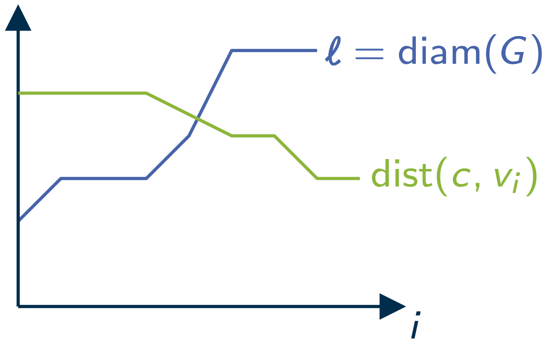
- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



$\Rightarrow 2 \cdot \mathrm{dist}(c, v_i) \leq \ell$ already implies $\ell = \mathrm{diam}(G)$

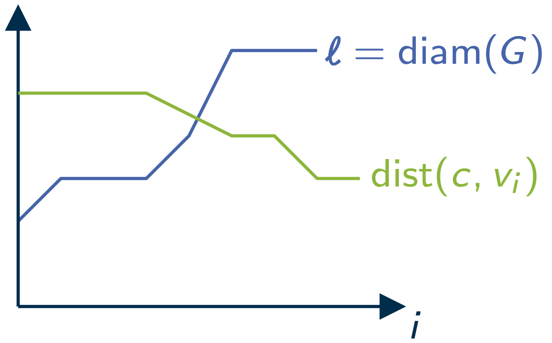


iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



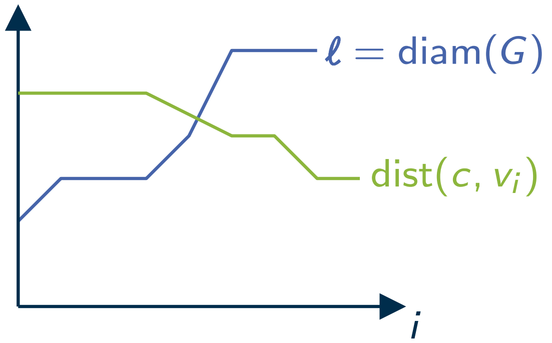
Question: How to choose c?

iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



Question: How to choose c?

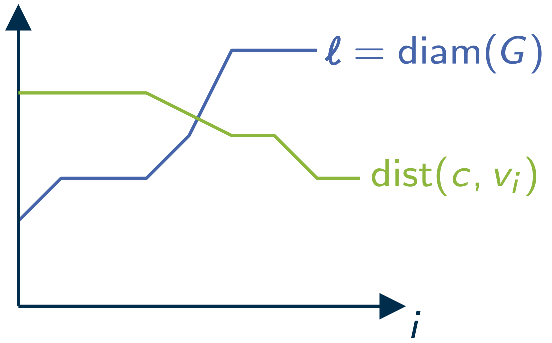
- heterogeneous graph: maximum degree

iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



Question: How to choose c?

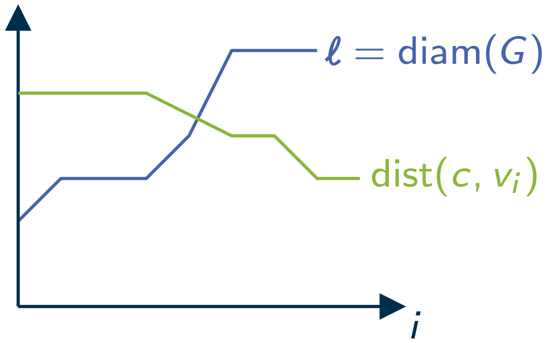
- heterogeneous graph: maximum degree
- otherwise: *2-sweep*

iFUB Algorithm

- select *central* vertex c

Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \dots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \dots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



Question: How to choose c?

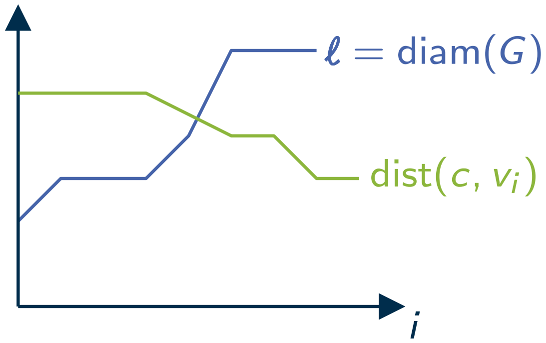
- heterogeneous graph: maximum degree
- otherwise: *2-sweep*
  - choose v arbitrarily, w most distant from v, w' most distant from w

iFUB Algorithm

- select *central* vertex c

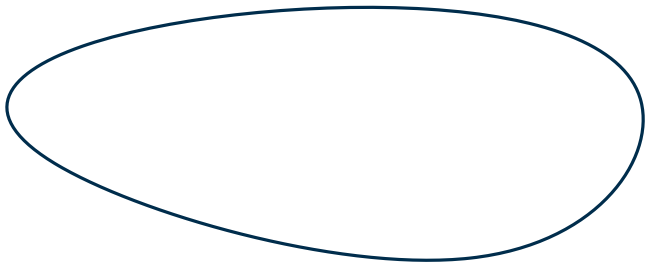
Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



Question: How to choose c?

- heterogeneous graph: maximum degree
- otherwise: *2-sweep*
  - choose v arbitrarily, w most distant from v, w' most distant from w

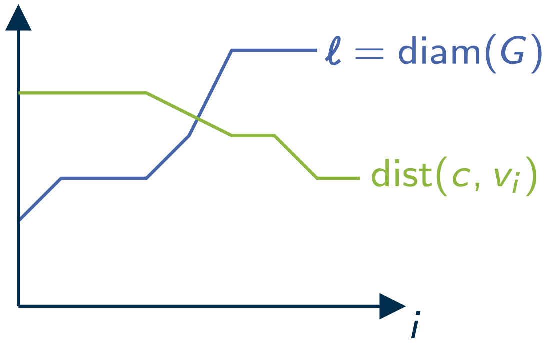


iFUB Algorithm

- select *central* vertex c

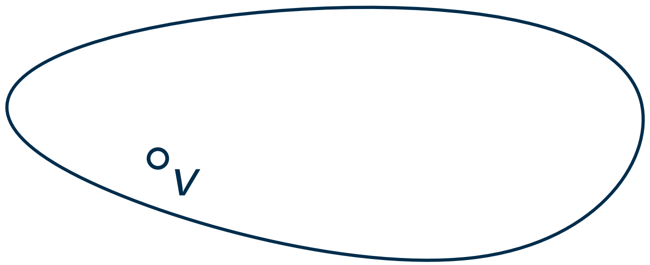
Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



Question: How to choose c?

- heterogeneous graph: maximum degree
- otherwise: *2-sweep*
  - choose v arbitrarily, w most distant from v, w' most distant from w

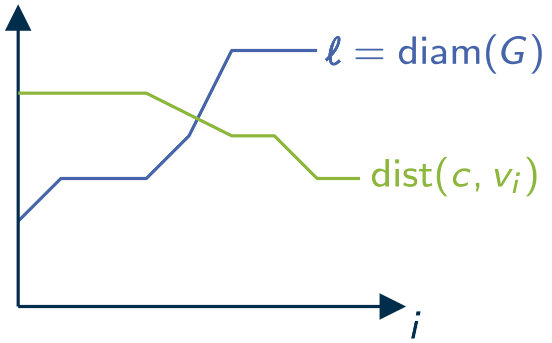


iFUB Algorithm

- select *central* vertex c

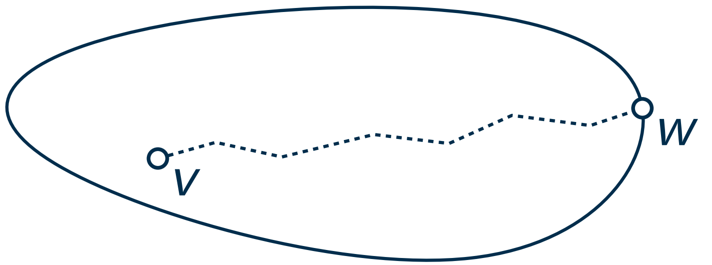
Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



Question: How to choose c?

- heterogeneous graph: maximum degree
- otherwise: *2-sweep*
  - choose v arbitrarily, w most distant from v, w' most distant from w

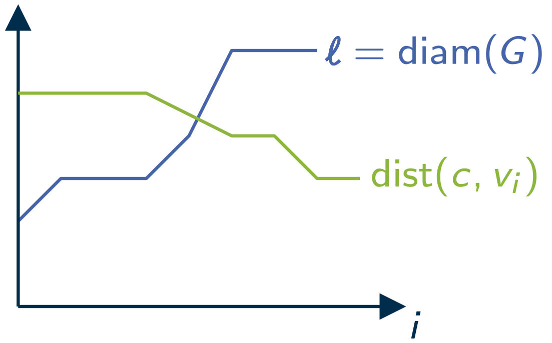


iFUB Algorithm

- select *central* vertex c

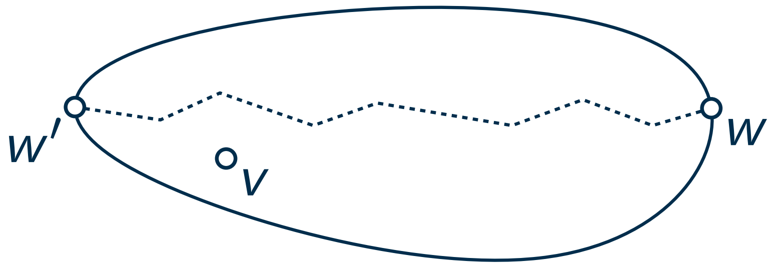
Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



Question: How to choose c?

- heterogeneous graph: maximum degree
- otherwise: *2-sweep*
  - choose v arbitrarily, w most distant from v, w' most distant from w

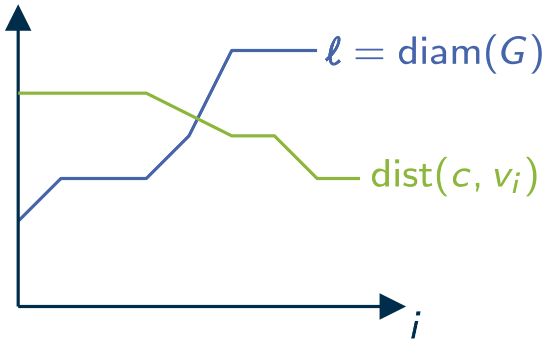


iFUB Algorithm

- select *central* vertex c

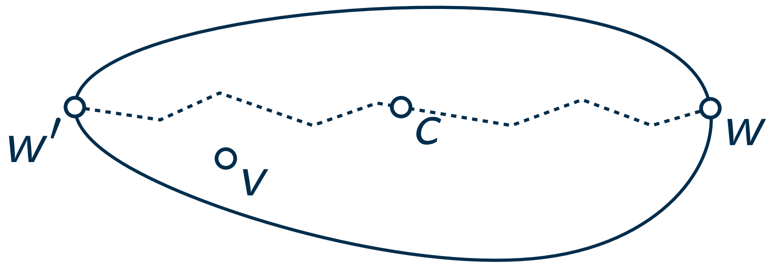
Observation: there is a diametrical vertex w with distance at least $\mathrm{diam}(G)/2$ from c.

- plan: run BFS until we find w
- let $v_1, \ldots, v_n$ be vertices in descending distance from c
- after running BFS from v_i:
  - $\max\{\mathrm{ecc}(v_1), \ldots, \mathrm{ecc}(v_i)\}$ gives lower bound ℓ on diameter
  - assume $2 \cdot \mathrm{dist}(c, v_i) < \ell$, then $\mathrm{dist}(c, v_i) < \mathrm{diam}(G)/2$
  - w already found, $\ell = \mathrm{diam}(G)$



Question: How to choose c?

- heterogeneous graph: maximum degree
- otherwise: *2-sweep*
  - choose v arbitrarily, w most distant from v, w' most distant from w
  - choose c in middle of shortest path between w and w'



Exercise Sheet 4 & Project

Exercise Sheet 3

- Optimize / clean up code and workflow
- Study the performance of iFUB on realistic inputs
- Time frame: only *one* week

Exercise Sheet 4 & Project

Exercise Sheet 3

- Optimize / clean up code and workflow
- Study the performance of iFUB on realistic inputs
- Time frame: only *one* week

Presentations?

- no fancy slides required, just briefly prepare to answer:
  1. What are your findings about iFUB?
  2. How does your pipeline look like?

Exercise Sheet 4 & Project

Exercise Sheet 3

- Optimize / clean up code and workflow
- Study the performance of iFUB on realistic inputs
- Time frame: only *one* week

Presentations?

- no fancy slides required, just briefly prepare to answer:
  1. What are your findings about iFUB?
  2. How does your pipeline look like?

Afterwards: Project (6 weeks)

- Goal: investigate and answer a research question
- Inspiration:
  - On the External Validity of Average-Case Analyses of Graph Algorithms [B., F. 2022]
  - Deterministic Performance Guarantees for Bidirectional BFS on Real-World Networks [B., W. 2022]
  - Understand / improve algorithms (e.g., diameter, hard problems, . . .)