

Grafer og bredde-først søgning

Diskret Matematik og Algoritmer
Københavns Universitet, efterår 2021

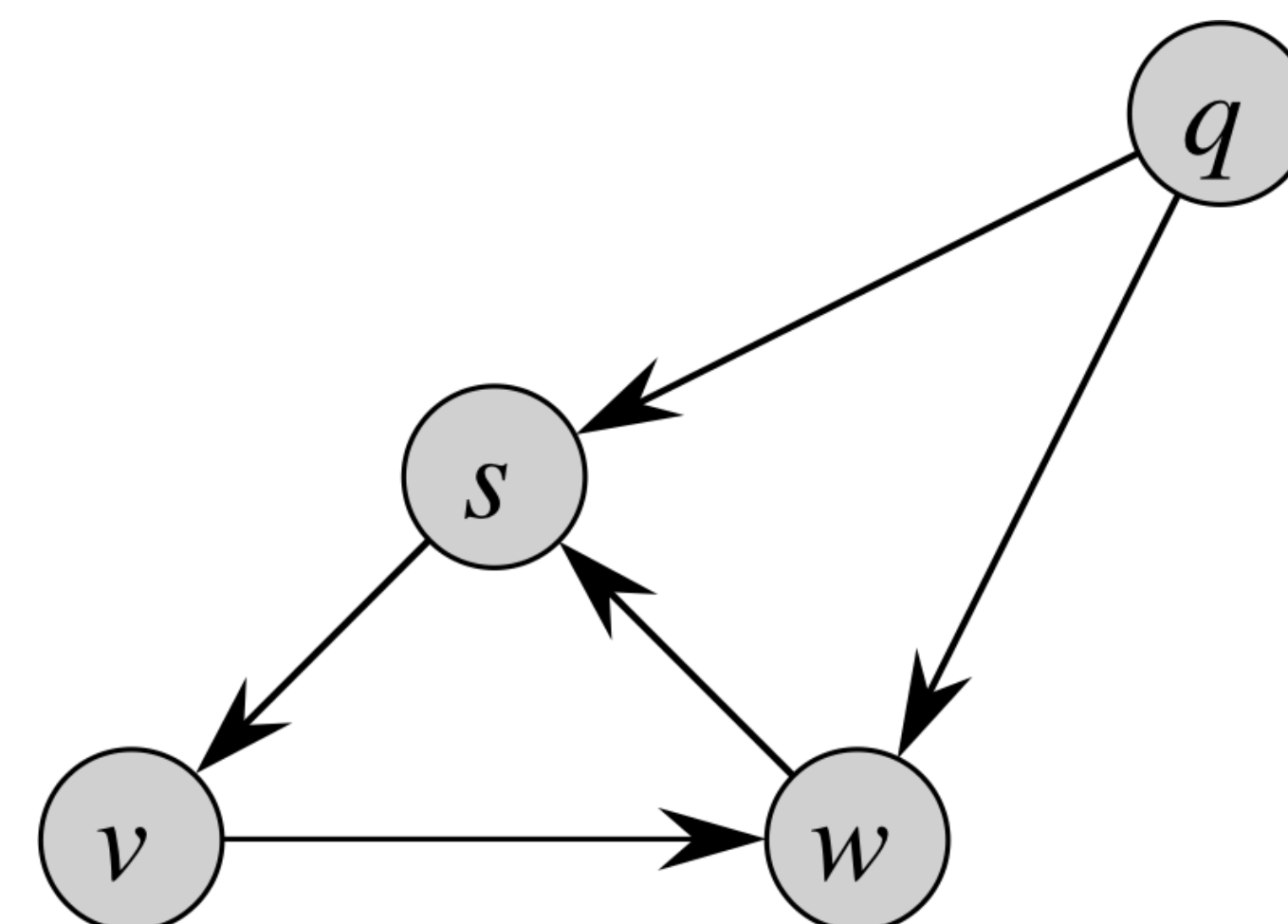
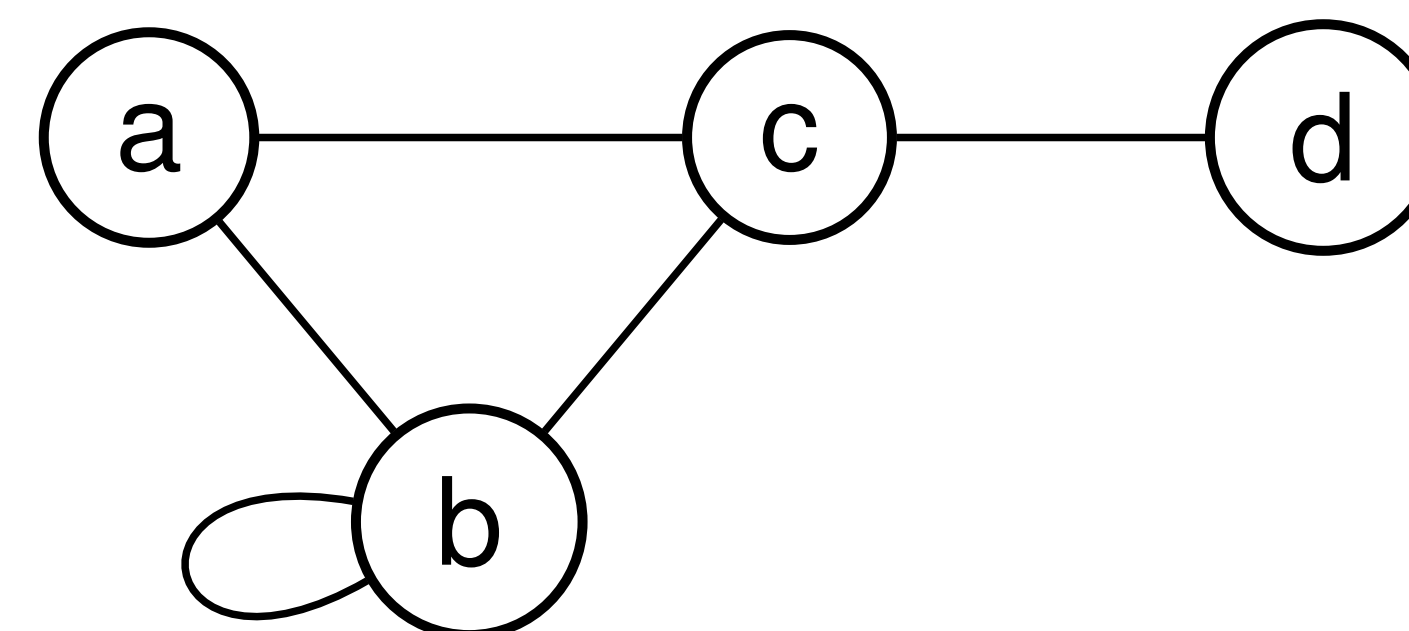
Rasmus Pagh

Slides med lyseblå baggrund er baseret på slides af Kevin Wayne



Begreber fra sidste uge

ENG	DAN
graph	(uorienteret) graf
node, vertex	knude
edge	kant
adjacent, neighbors	naboer
cycle	kreds
path	sti
degree	grad
connected	forbundne
tree	træ
digraph	orienteret graf



Hvad sker der?

Litteratur

- *Noter om grafer* (spring evt. over materialet om dybde først søgning)
- CLRS sektion 22.1 og 22.2 (dækker samme emner som noten, men med flere detaljer)
- CLRS 24.0, 24.3 (dækker samme emner som noten, men med flere detaljer)

Mål for ugen

- Kendskab til terminologi for grafer
- Kendskab til repræsentationer af grafer (tætte og tynde)
- Forståelse af bredde-først søgning og Dijkstra's algoritme

Plan for ugen

- Mandag: Repræsentation af grafer (Noter om grafer / CLRS 22.1), bredde-først søgning (Noter om grafer / CLRS 22.2)
- Tirsdag: Korteste vej, Dijkstra's algoritme (Noter om grafer / CLRS 24.0, 24.3)
- Fredag: Ingen forelæsning, juleferie!

Motiverende case

- Data: En liste af n LinkedIn kontakter
[(Rasmus, Ravi) , (Rasmus, Peter) , (Ravi, Andrei) , (Andrei, Jeff) , ...]
- Spørgsmål:
Hvor lang er den korteste
forbindelse i LinkedIn-grafen
fra Rasmus til Mark Zuckerberg?



Mark Zuckerberg

• 3rd+

CEO at Facebook

Palo Alto, CA

Connect

Ravi

Research

Mountain

500+ c



Mes

Bjarne Stroustrup 2nd

Technical Fellow and Managing Director at Morgan Stanley

New York, New York, United States · [Contact info](#)

500+ connections



5 mutual connections: Peter Sestoft, Adam Buchsbaum, ...

Connect

 **Message**

More

Anvendelser af grafer

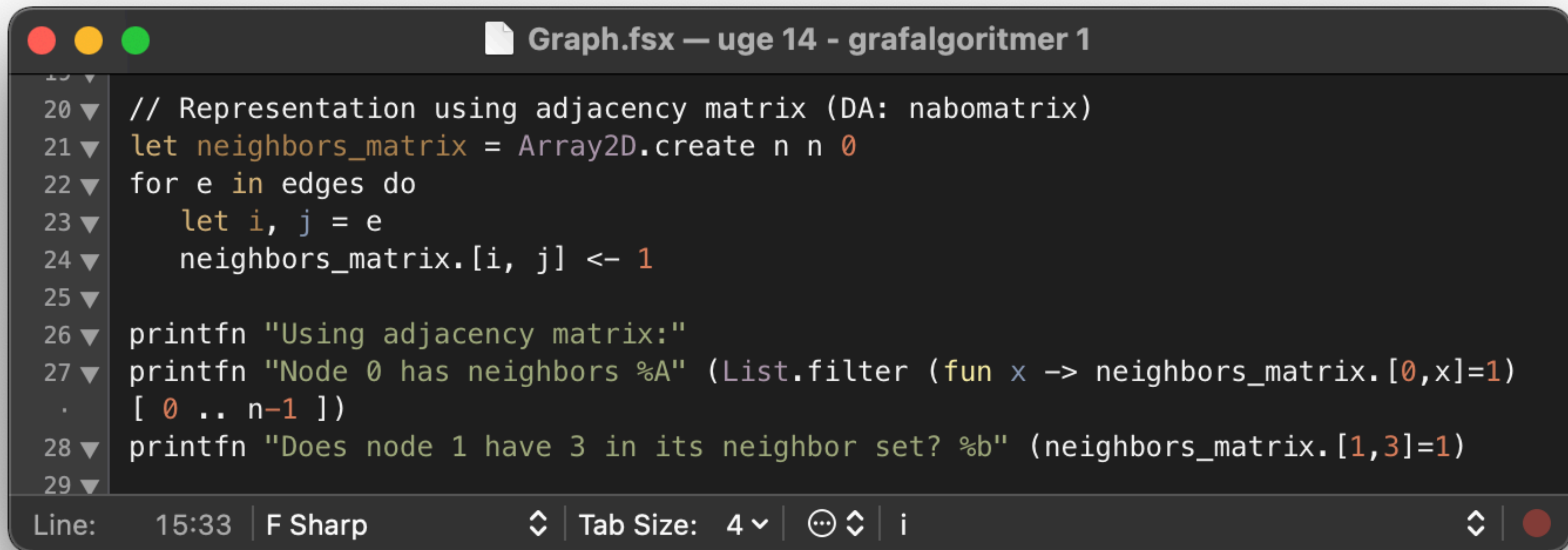
graf	Knude	kant
communication	telephone, computer	fiber optic cable
circuit	gate, register, processor	wire
mechanical	joint	rod, beam, spring
financial	stock, currency	transactions
transportation	street intersection, airport	highway, airway route
internet	class C network	connection
game	board position	legal move
social relationship	person, actor	friendship, movie cast
neural network	neuron	synapse
protein network	protein	protein-protein interaction
molecule	atom	bond

Repræsentation 1: Nabolister

```
Graph.fsx — uge 14 - grafalgoritmer 1
2 ▼
3 ▼ // Graph from slide 2 in lecture, nodes a, b, c, d numbered 0, 1, 2, 3
4 ▼ let edges = [ (0,1); (0,2); (1,1); (1,2); (2,3); (1,0); (2,0); (2,1); (3,2) ]
5 ▼ let n = 4 // number of nodes
6 ▼
7 ▼ // Representation using adjacency lists (DA: nabolister)
8 ▼ let neighbors_list = Array.create n (new List<int>())
9 ▼ for i in 0 .. n-1 do
10 ▼     neighbors_list.[i] <- new List<int>()
11 ▼ for e in edges do
12 ▼     let i, j = e
13 ▼     neighbors_list.[i].Add(j) |> ignore
14 ▼
15 ▼ printfn "Using adjacency lists:"
16 ▼ printfn "Node 0 has neighbors %A" (neighbors_list.[0])
17 ▼ printfn "Does node 1 have 3 in its neighbor set? %b" (neighbors_list.[1].Contains(3))
18 ▼
```

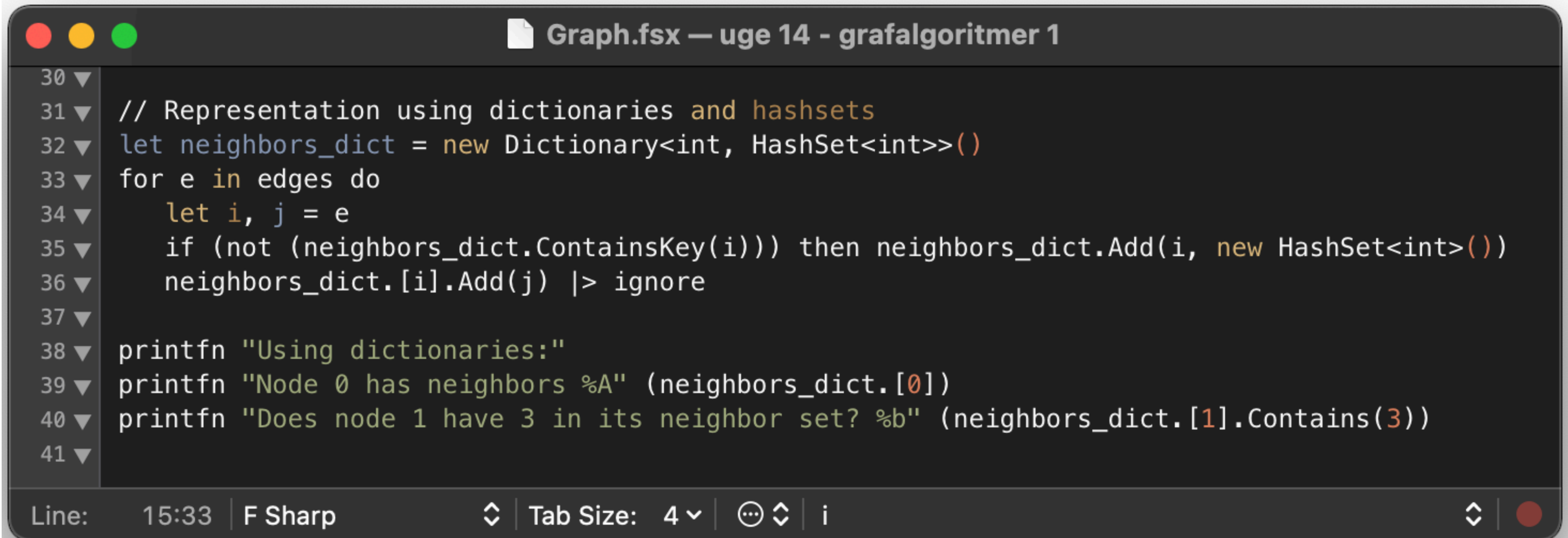
Line: 15:33 | F Sharp | Tab Size: 4 | i

Repræsentation 2: Nabomatrixer



```
19 ▼  
20 ▼ // Representation using adjacency matrix (DA: nabomatrix)  
21 ▼ let neighbors_matrix = Array2D.create n n 0  
22 ▼ for e in edges do  
23 ▼     let i, j = e  
24 ▼     neighbors_matrix.[i, j] <- 1  
25 ▼  
26 ▼ printfn "Using adjacency matrix:"  
27 ▼ printfn "Node 0 has neighbors %A" (List.filter (fun x -> neighbors_matrix.[0,x]=1)  
28 ▼     [ 0 .. n-1 ])  
29 ▼ printfn "Does node 1 have 3 in its neighbor set? %b" (neighbors_matrix.[1,3]=1)  
Line: 15:33 | F Sharp | Tab Size: 4 | i
```


Repræsentation 3: Hashtabeller i hashtabeller



```
30 ▼  
31 ▼ // Representation using dictionaries and hashsets  
32 ▼ let neighbors_dict = new Dictionary<int, HashSet<int>>()  
33 ▼ for e in edges do  
34 ▼     let i, j = e  
35 ▼     if (not (neighbors_dict.ContainsKey(i))) then neighbors_dict.Add(i, new HashSet<int>())  
36 ▼     neighbors_dict.[i].Add(j) |> ignore  
37 ▼  
38 ▼ printfn "Using dictionaries:"  
39 ▼ printfn "Node 0 has neighbors %A" (neighbors_dict.[0])  
40 ▼ printfn "Does node 1 have 3 in its neighbor set? %b" (neighbors_dict.[1].Contains(3))  
41 ▼
```

Line: 15:33 | F Sharp

Tab Size: 4

Øvelse

- Betragt en graf med n knuder og m kanter
- For hver af de tre repræsentationer, hvad er store-O kompleksiteten af følgende:
 1. Tid for at finde alle naboer til knude i
 2. Tid for at afgøre om i og j er naboer
 3. Tid for at lave en liste med alle kanter
 4. Plads til datastrukturen

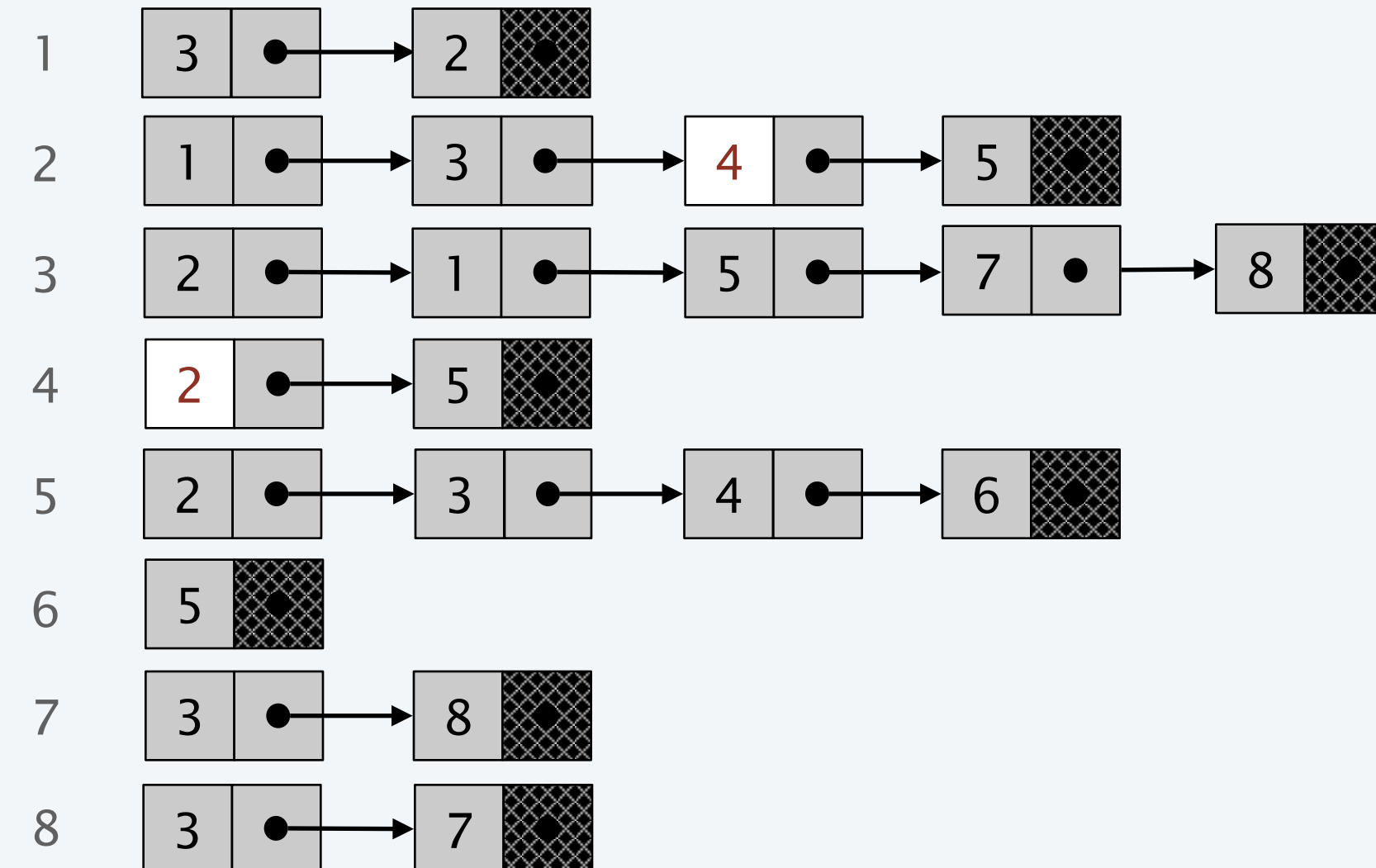
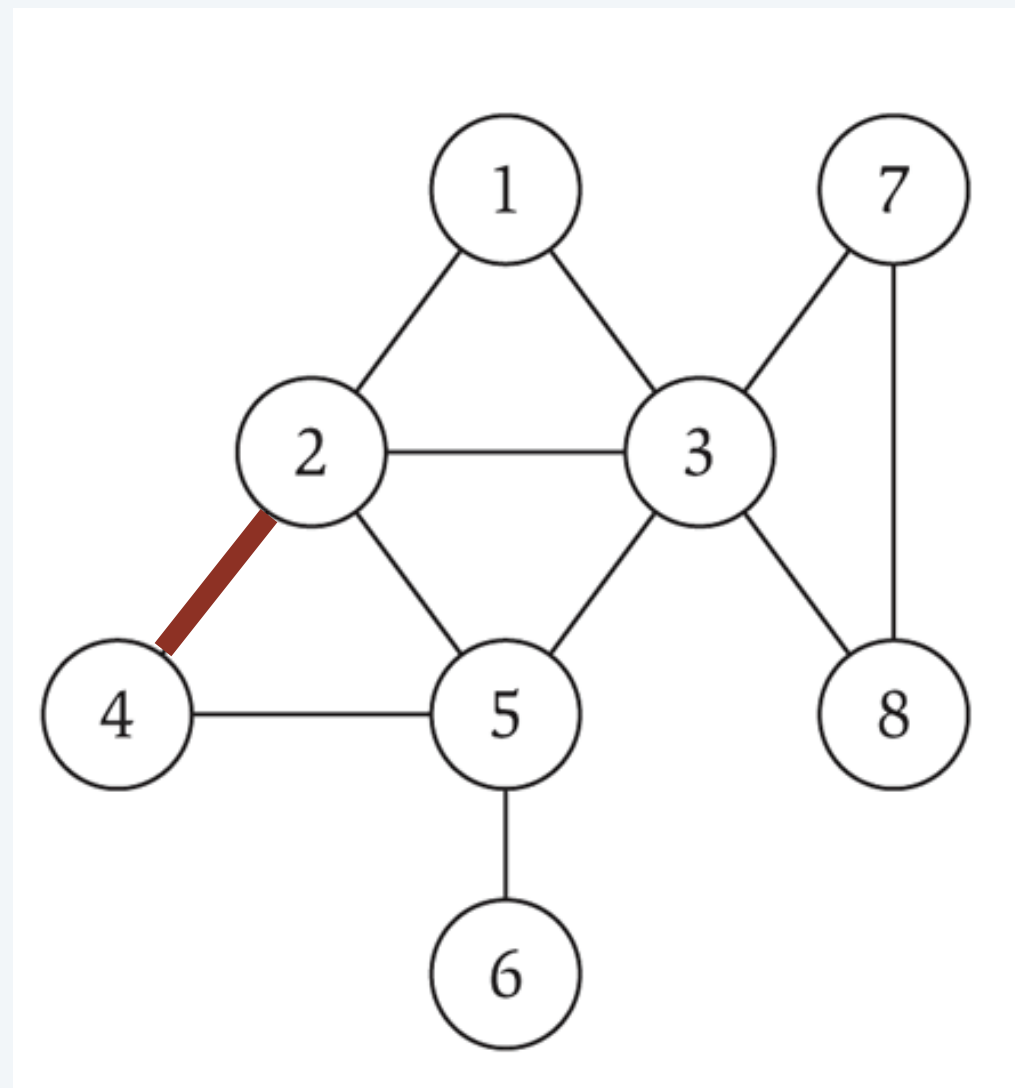
	Naboliste	Nabomatrix	Hashtabel
1			
2			
3			
4			

Opsummering: Nabolister

Nabolister. Knude-indekseret tabel af lister.

- To liste-elementer for hver kant (ikke orienteret).
- Plads $\Theta(m + n)$.
- At undersøge om (u, v) er en kant tager $O(\text{grad}(u))$ tid.
- At finde alle kanter tager $\Theta(m + n)$ tid.

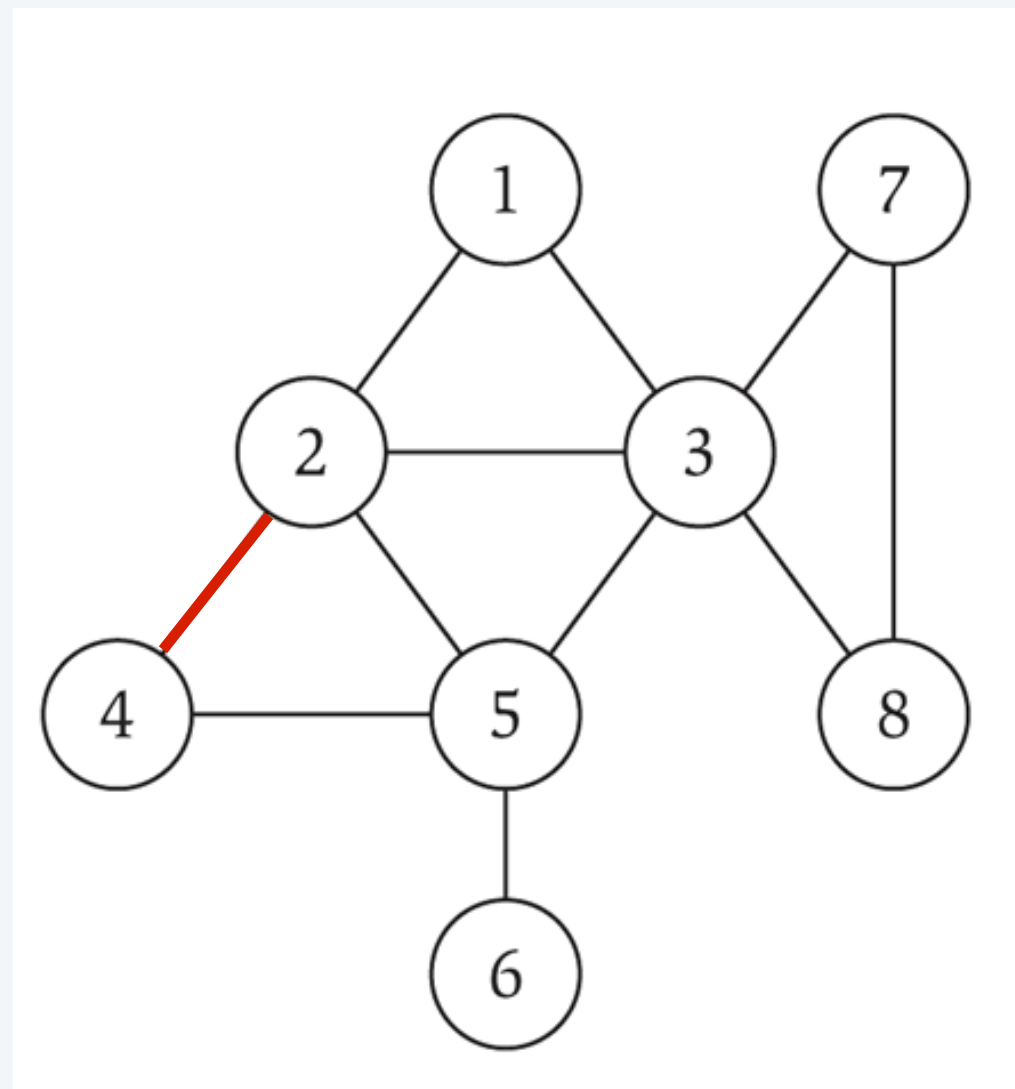
grad = antal naboer til u



Opsummering: Nabomatrix

Nabomatrix. n -gange- n matrix hvor $A_{uv} = 1$ hvis (u, v) er en kant, $A_{uv} = 0$ ellers.

- To 1ere for hver kant (ikke orienteret).
- Plads proportional med n^2 .
- At undersøge om (u, v) er en kant tager $\Theta(1)$ tid.
- At finde alle kanter tager $\Theta(n^2)$ tid.

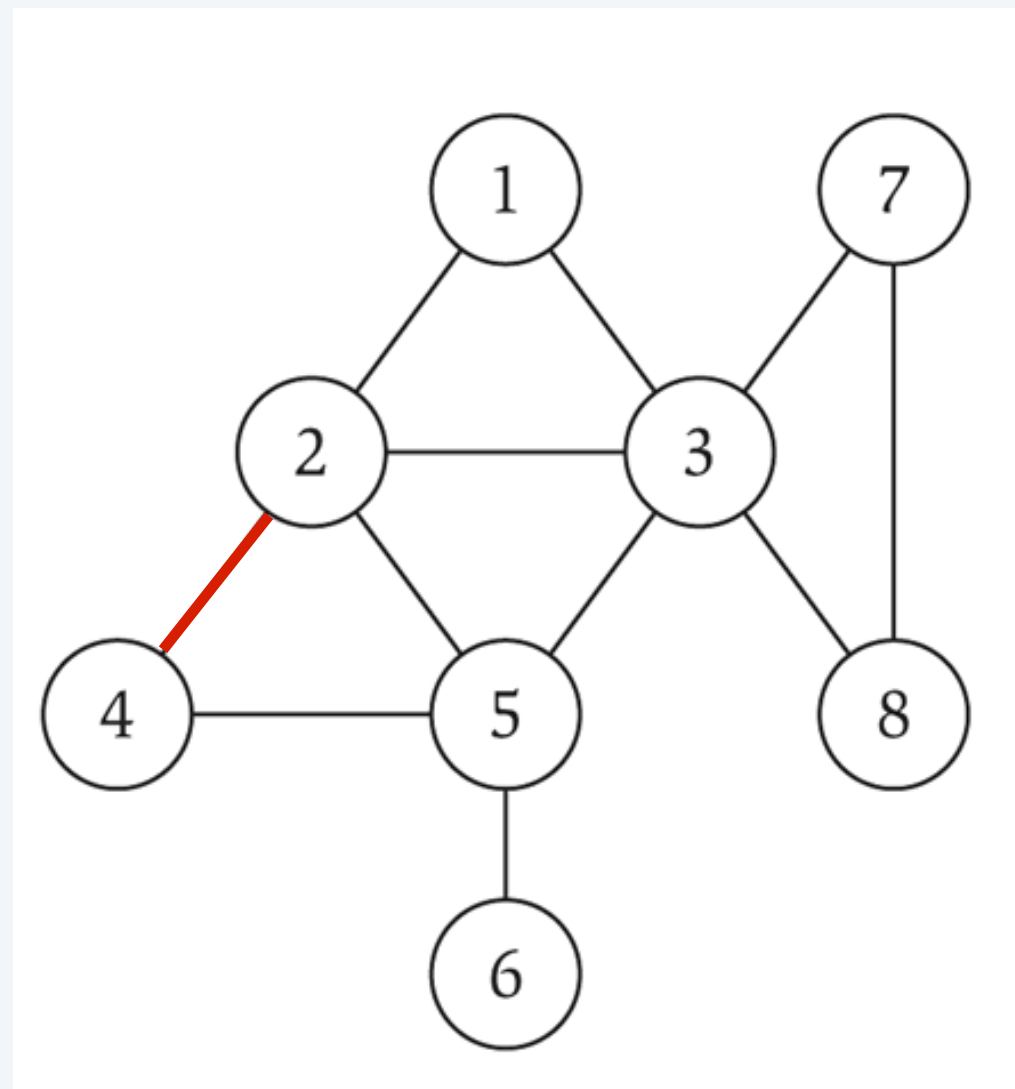


	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

Opsummering: Hashtabeller i hashtabeller

Ordbog A hvor $A[u]$ er et hashset, der indeholder v hvis og kun hvis (u, v) er en kant.

- To nøgler og to elementer i hashsets for hver kant (ikke orienteret).
- Plads $\Theta(m + n)$.
- At undersøge om (u, v) er en kant tager $\Theta(1)$ tid.
- At finde alle kanter tager $\Theta(n+m)$ tid.



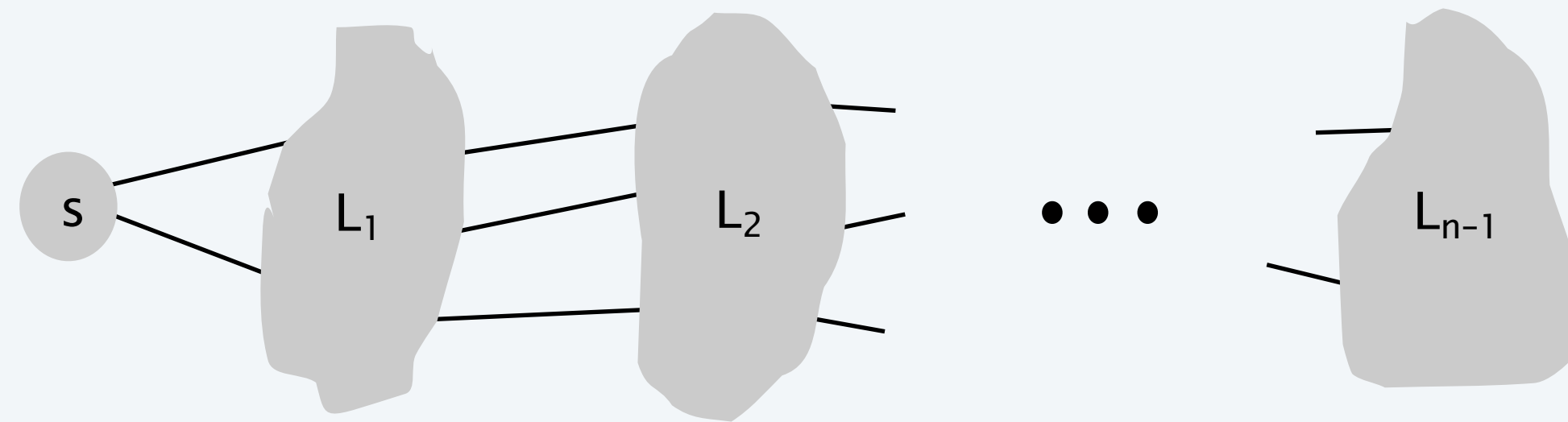
Bredde-først søgning (BFS)

Mål. Find korteste vej fra en knude s til alle andre knuder i grafen (hvor den findes)

BFS intuition. Udforsk grafen med udgangspunkt i s , læg nye knuder til i ét “lag” ad gangen.

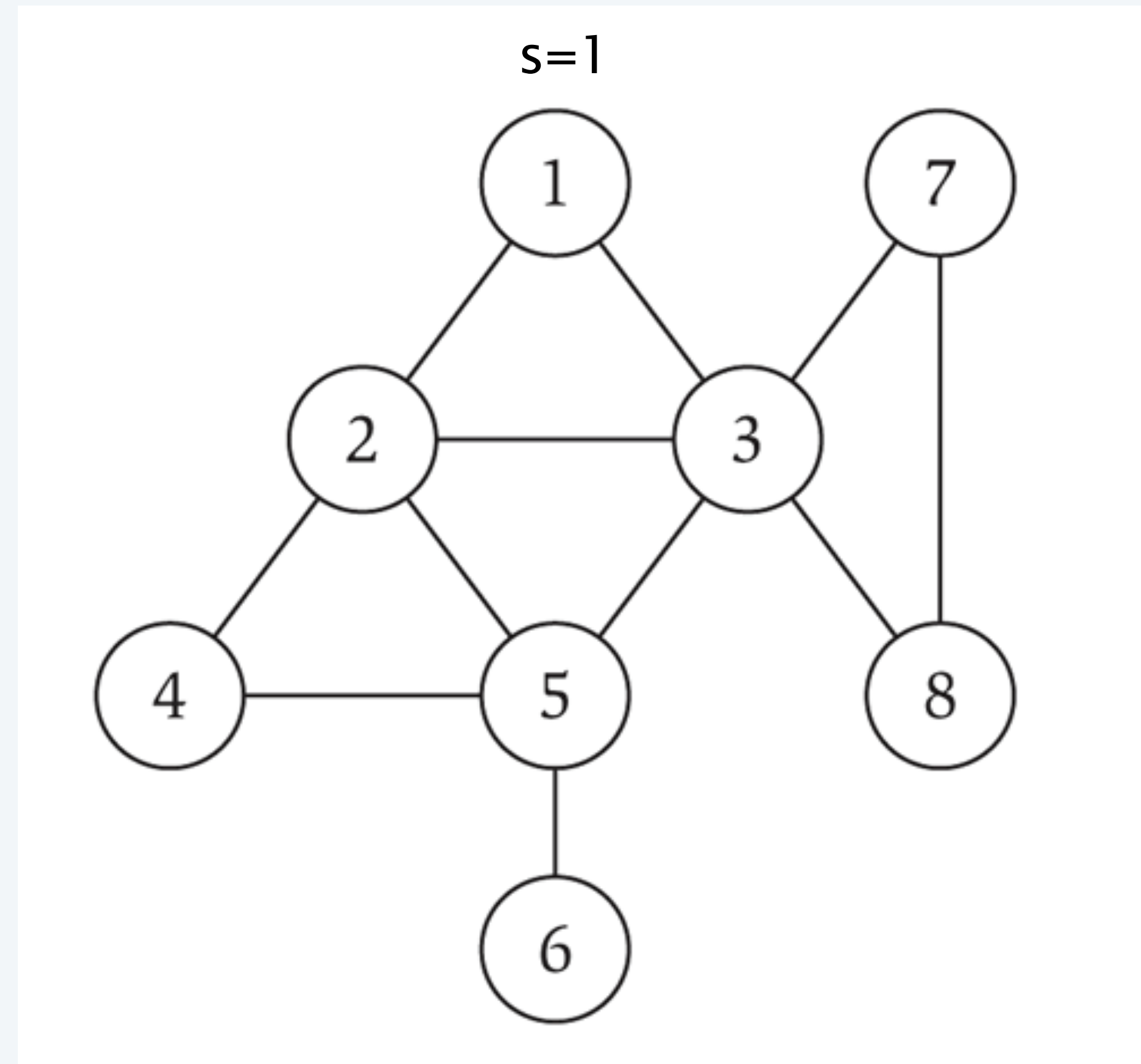
BFS algoritme.

- $L_0 = \{ s \}$.
- L_1 = alle naboer til L_0 .
- L_2 = alle knuder der ikke tilhører L_0 eller L_1 , og som har en kant til en knude i L_1 .
- L_{i+1} = alle knuder der ikke tilhører et tidligere lag, og som har en kan til en knude i L_i .



Teorem. For hvert i består L_i af alle knuder der har afstand præcis i fra s . Der er en sti fra s til t hvis og kun t forekommer i et af lagene.

BFS eksempler



Animation: Bredde-først søgning i en labyrint

BFS pseudocode

```
BFS( $V, E, s$ )  
  for each  $u \in V - \{s\}$   
     $u.d = \infty$   
   $s.d = 0$   
   $Q = \emptyset$   
  ENQUEUE( $Q, s$ )  
  while  $Q \neq \emptyset$   
     $u = \text{DEQUEUE}(Q)$   
    for each  $v \in G.\text{Adj}[u]$   
      if  $v.d == \infty$   
         $v.d = u.d + 1$   
        ENQUEUE( $Q, v$ )
```

BFS analyse

Teorem. BFS kører i tid $O(m + n)$ hvis grafen er i naboliste representation.

Bevis.

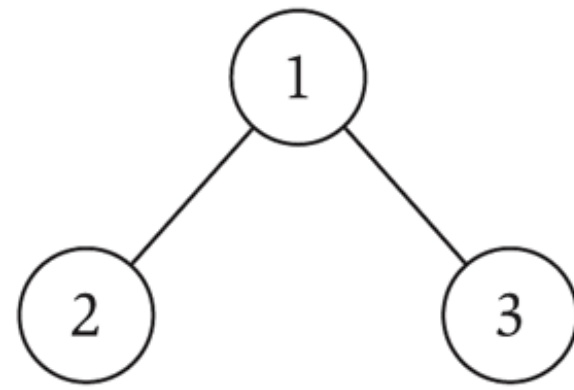
- Nemt at vise $O(n^2)$ køretid:
 - der er højst n nabolister $L[i]$
 - hver knude tilføjes køen højst én gang; while-løkken kører $\leq n$ gange
 - når vi betragter knude u , er der $\leq n$ tilstødende kanter (u, v) , og vi bruger tid $O(1)$ for hver af dem
- Kører faktisk i $O(m + n)$ tid:
 - når vi betragter knude u , er der $grad(u)$ tilstødende kanter (u, v)
 - totalt bruger vi tid $\sum_{u \in V} grad(u) = 2m$ på at løbe gennem kanter. ■



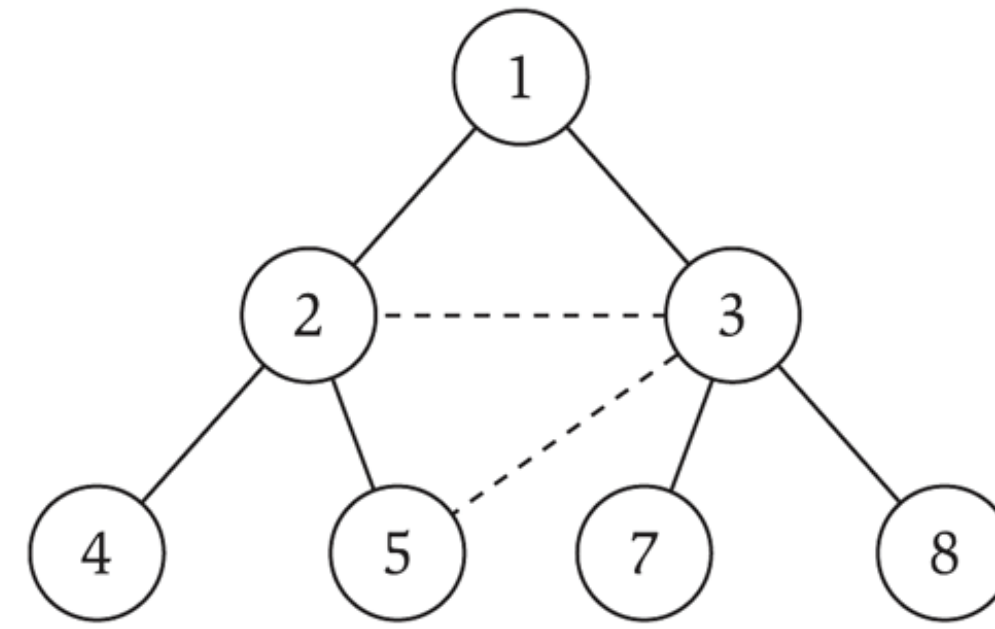
Hver kant (u, v) tælles præcis to gange i summen:
én gang i $grad(u)$ og én gang i $grad(v)$

```
BFS( $V, E, s$ )
  for each  $u \in V - \{s\}$ 
     $u.d = \infty$ 
   $s.d = 0$ 
   $Q = \emptyset$ 
  ENQUEUE( $Q, s$ )
  while  $Q \neq \emptyset$ 
     $u = \text{DEQUEUE}(Q)$ 
    for each  $v \in G.Adj[u]$ 
      if  $v.d == \infty$ 
         $v.d = u.d + 1$ 
        ENQUEUE( $Q, v$ )
```

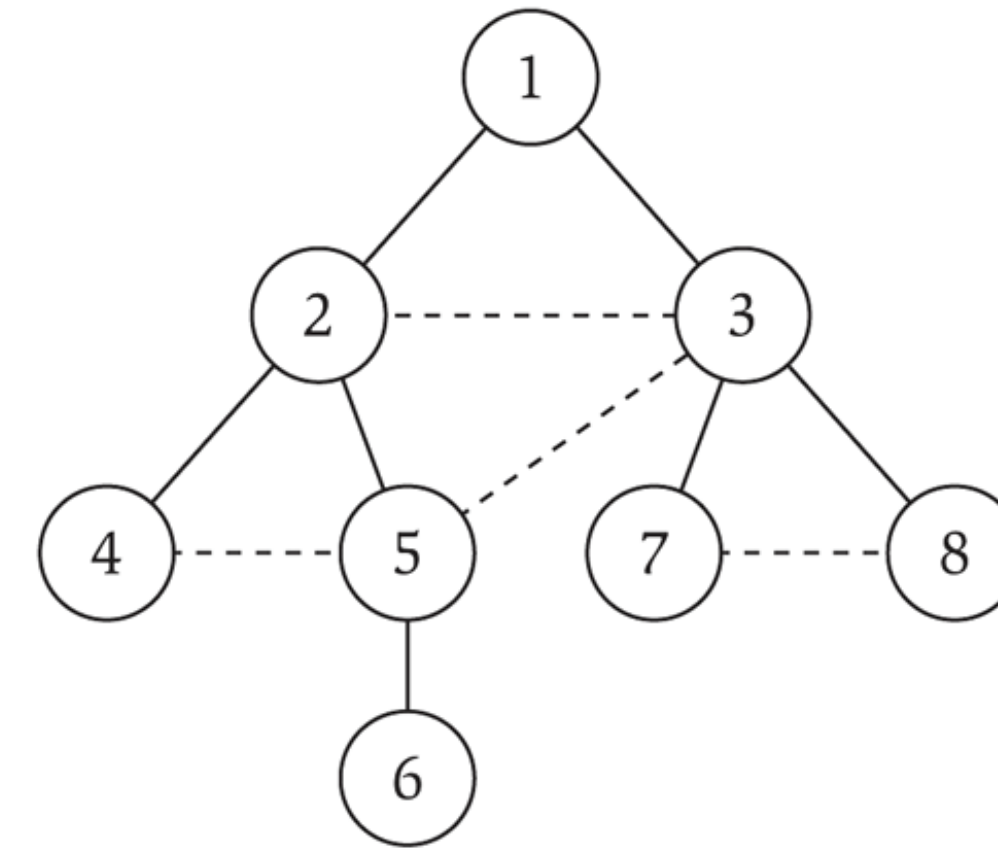

BFS træet



(a)



(b)



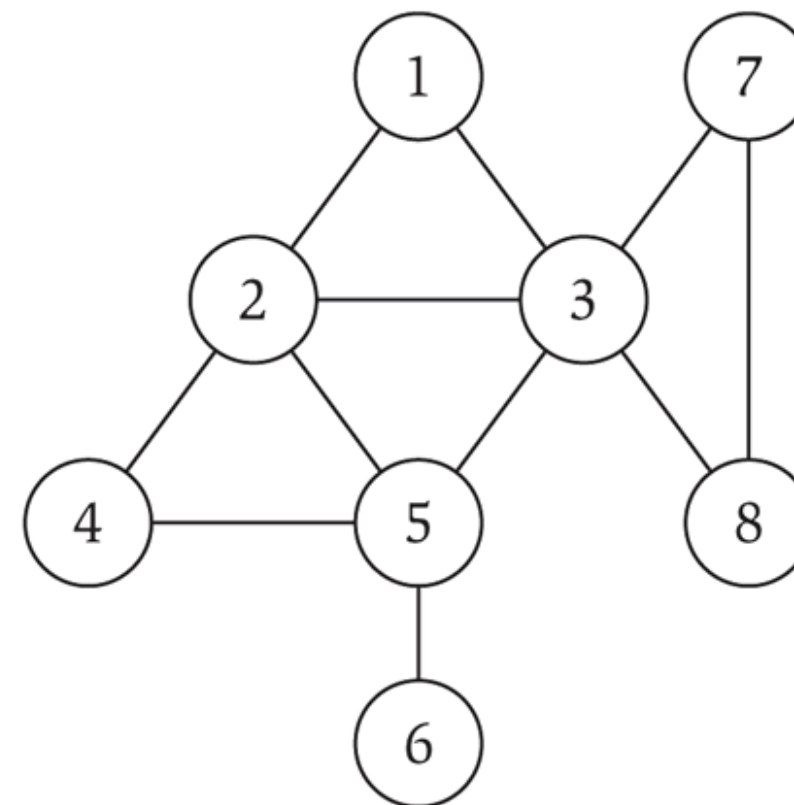
(c)

L_0

L_1

L_2

L_3



Opsummering

- Grafer har flere effektive repræsentationer, med forskellige egenskaber
 - Nabolister er det mest almindelige, hashtabeller et fleksibelt alternativ
- Bredde-først søgning (BFS) finder korteste veje i en graf med n knuder og m kanter i tid $O(n + m)$

I morgen: Korteste veje når kanter har forskellige længder