

Korteste veje

Diskret Matematik og Formelle Sprog
Københavns Universitet, marts 2023

Rasmus Pagh

Slides med lyseblå baggrund er baseret på slides af Kevin Wayne



Hvad sker der?

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

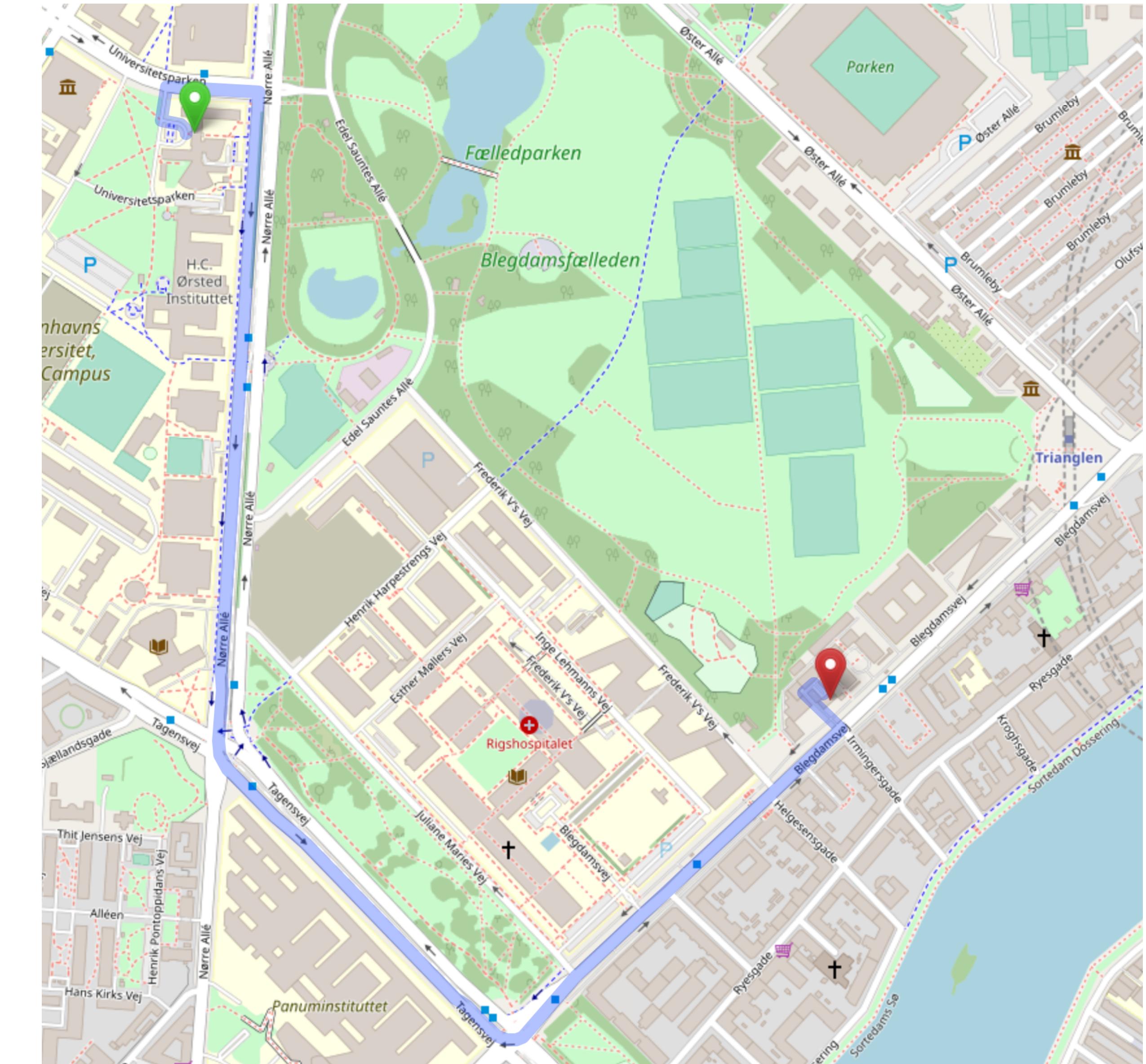
Plan for ugen

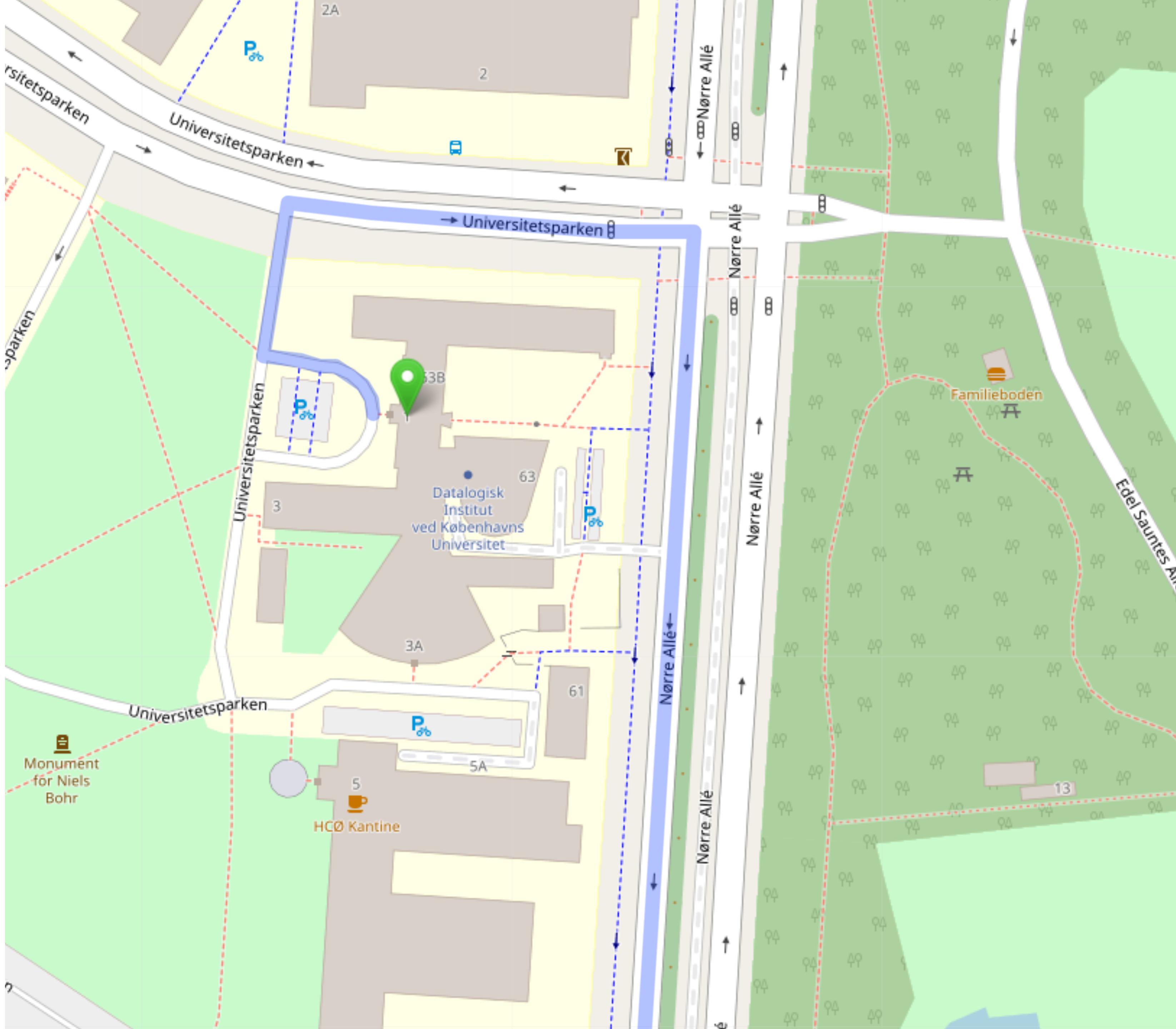
Onsdag Repræsentation af grafer (Noter om grafer / CLRS 20.1), bredde-først søgning
(Noter om grafer / CLRS 20.2)

Mandag Korteste vej, Dijkstra's algoritme (Noter om grafer / CLRS 22.0, 22.3)

Motiverende case

- Data: Afstande og hastighedsbegrænsninger i et vejnet (fx OpenStreetMaps)
- Spørgsmål: Hvad er den korteste vej fra DIKU til Niels Bohr Instituttet?
- Idé
Modellér vejnettet som en graf:
 - Knuder er positioner, fx vejkryds
 - Kanter er vejstykker mellem knuder
 - Hver kant har en “vægt”, svarende til hvor lang tid, det tager at bevæge sig langs den.

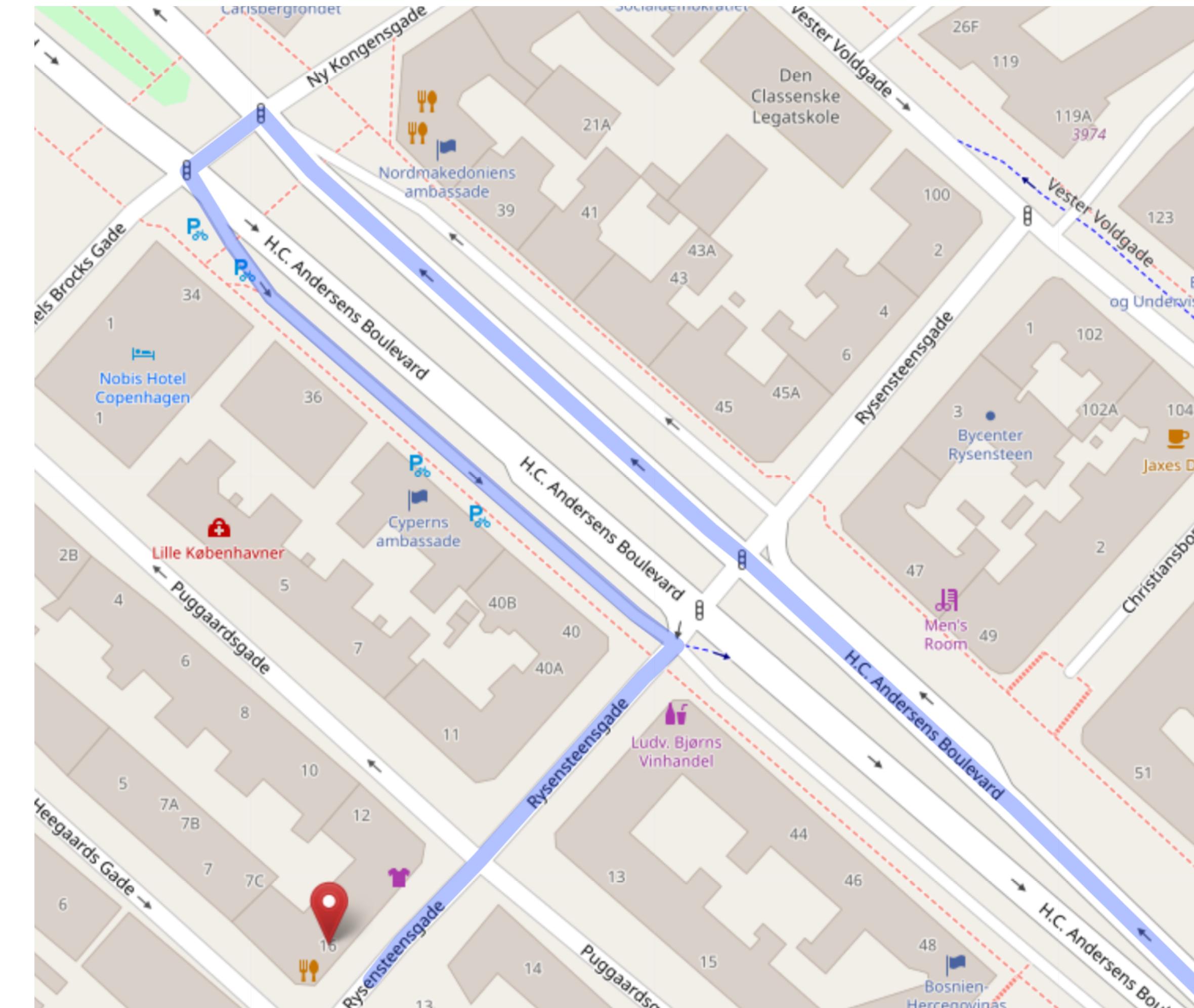




Modellering med grafer

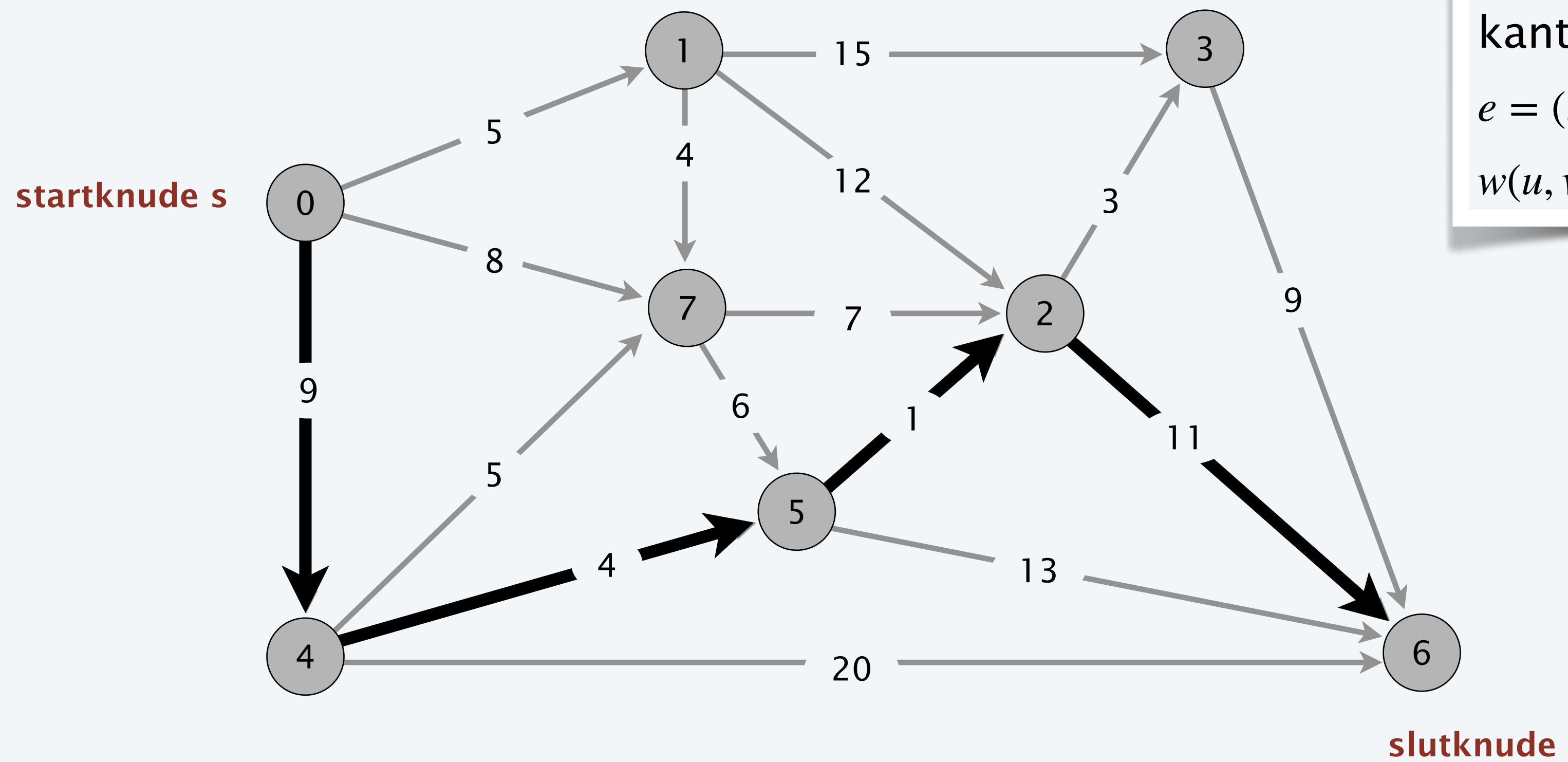
- I nogle vejkryds er venstresving forbudt.

Hvordan kan vi modellere det som en graf?



s-t korteste vej problemet

Problem. Givet en orienteret graf $G = (V, E)$, kantlængder $\ell_e \geq 0$, en startknude $s \in V$, og en slutknude $t \in V$, find en korteste orienteret sti fra s til t .

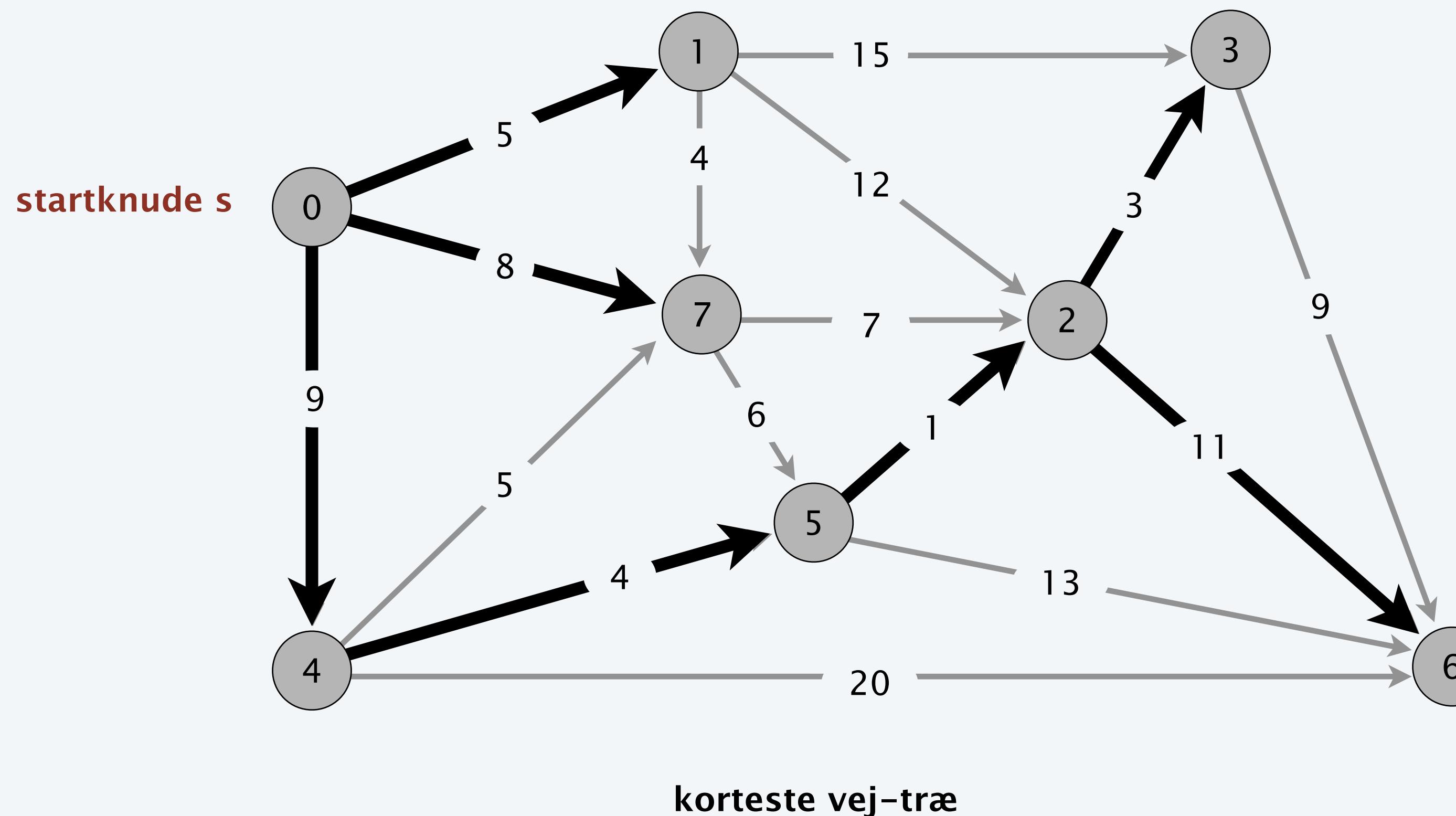


CLRS notation:
kantlængden ℓ_e for
 $e = (u, v)$ skrives som
 $w(u, v)$, kaldes for "vægt"

$$\text{Længde af sti} = 9 + 4 + 1 + 11 = 25$$

s-t korteste vej-træ problemet

Problem. Givet en orienteret graf $G = (V, E)$, kantlængder $\ell_e \geq 0$, en startknude $s \in V$, find en korteste orienteret sti fra s til **alle** andre knuder.



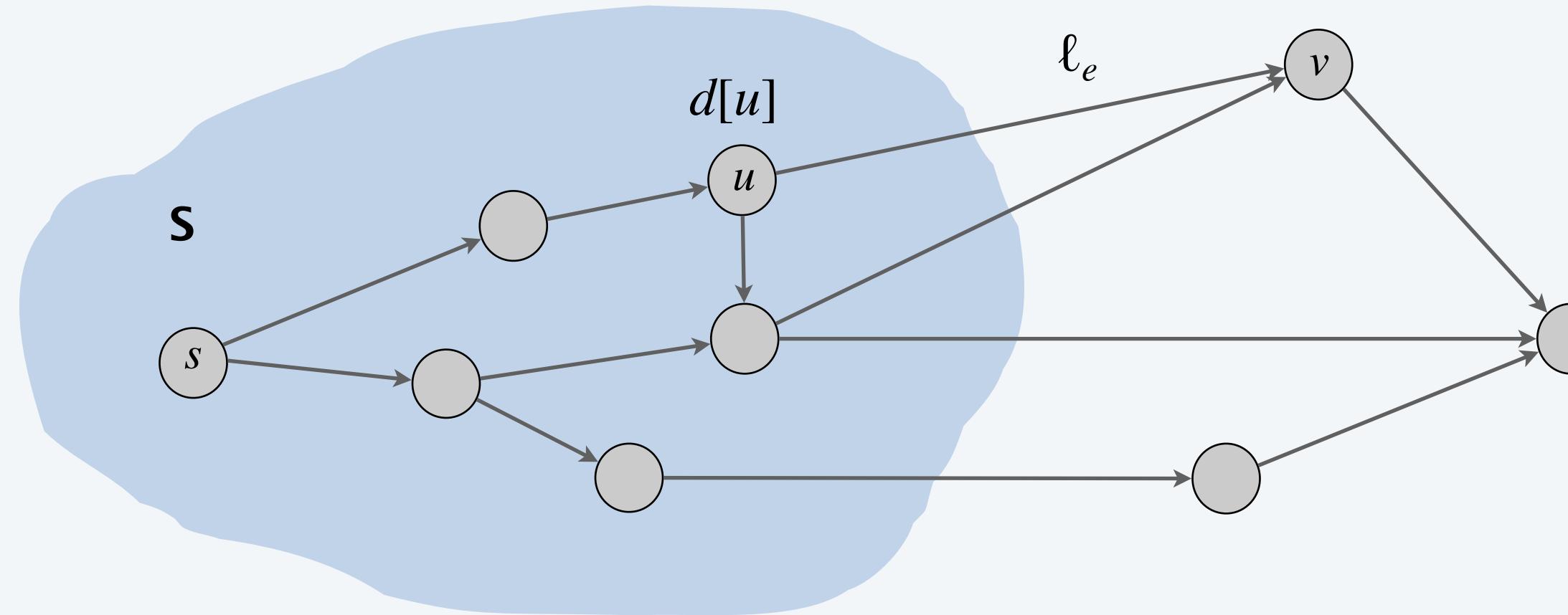
Dijkstra's algorithm (s-t korteste vej-træ problemet)

Grådig algoritme. Vedligehold en mængde S af knuder hvor algoritmen har fundet længden $d[u]$ af af den korteste sti $s \rightsquigarrow u$ for hver $u \in S$.

- Initialisér $S \leftarrow \{ s \}$, $d[s] \leftarrow 0$.
- Vælg hele tiden den knude $v \notin S$ der giver kortest mulige vej med én ny kant:

$$\sigma(v) = \min_{e = (u, v) : u \in S} d[u] + \ell_e$$

længden af en korteste vej fra s til en knude u , fulgt af kanten $e = (u, v)$



Dijkstra's algorithm (s-t korteste vej-træ problemet)

Grådig algoritme. Vedligehold en mængde S af knuder hvor algoritmen har fundet længden $d[u]$ af af den korteste sti $s \rightsquigarrow u$ for hver $u \in S$.



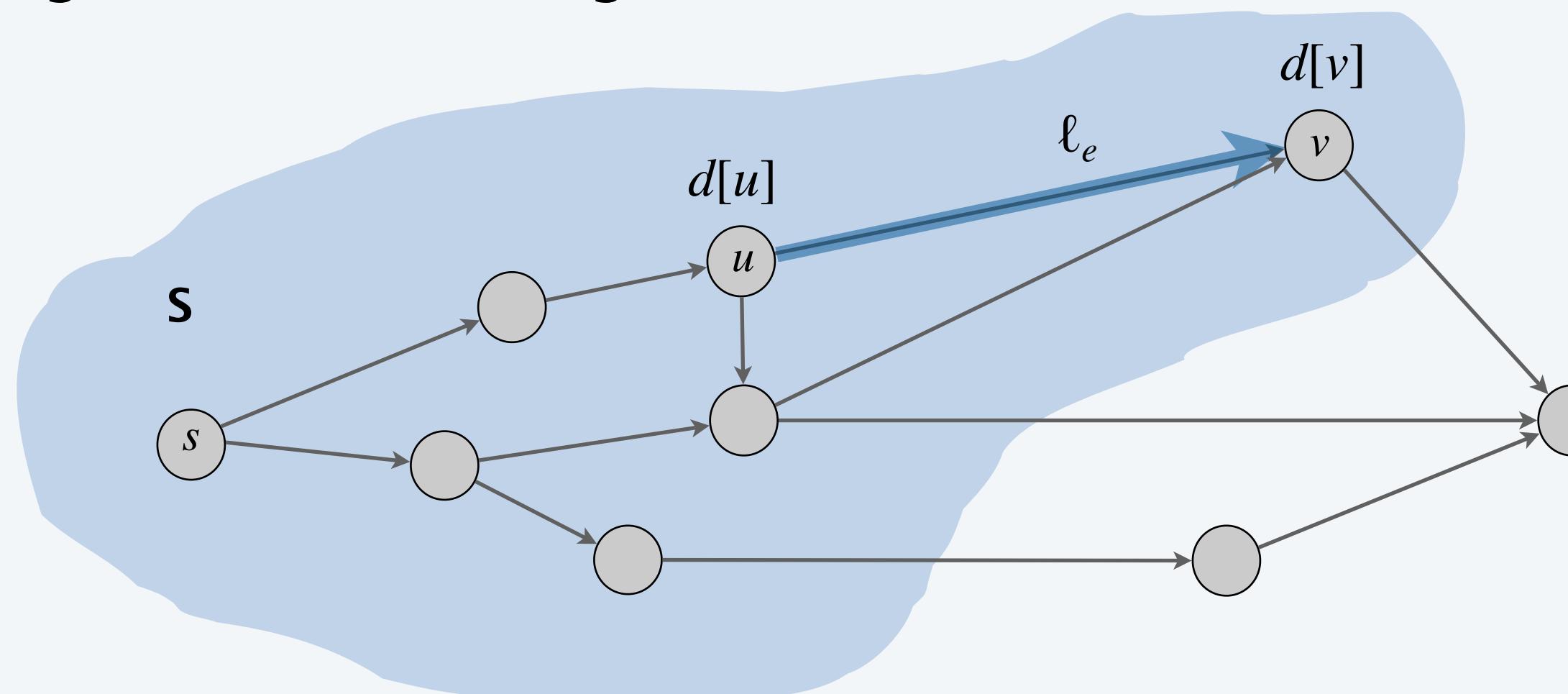
- Initialisér $S \leftarrow \{s\}$, $d[s] \leftarrow 0$.
- Vælg hele tiden den knude $v \notin S$ der giver kortest mulige vej med én ny kant:

$$\sigma(v) = \min_{e = (u,v) : u \in S} d[u] + \ell_e$$

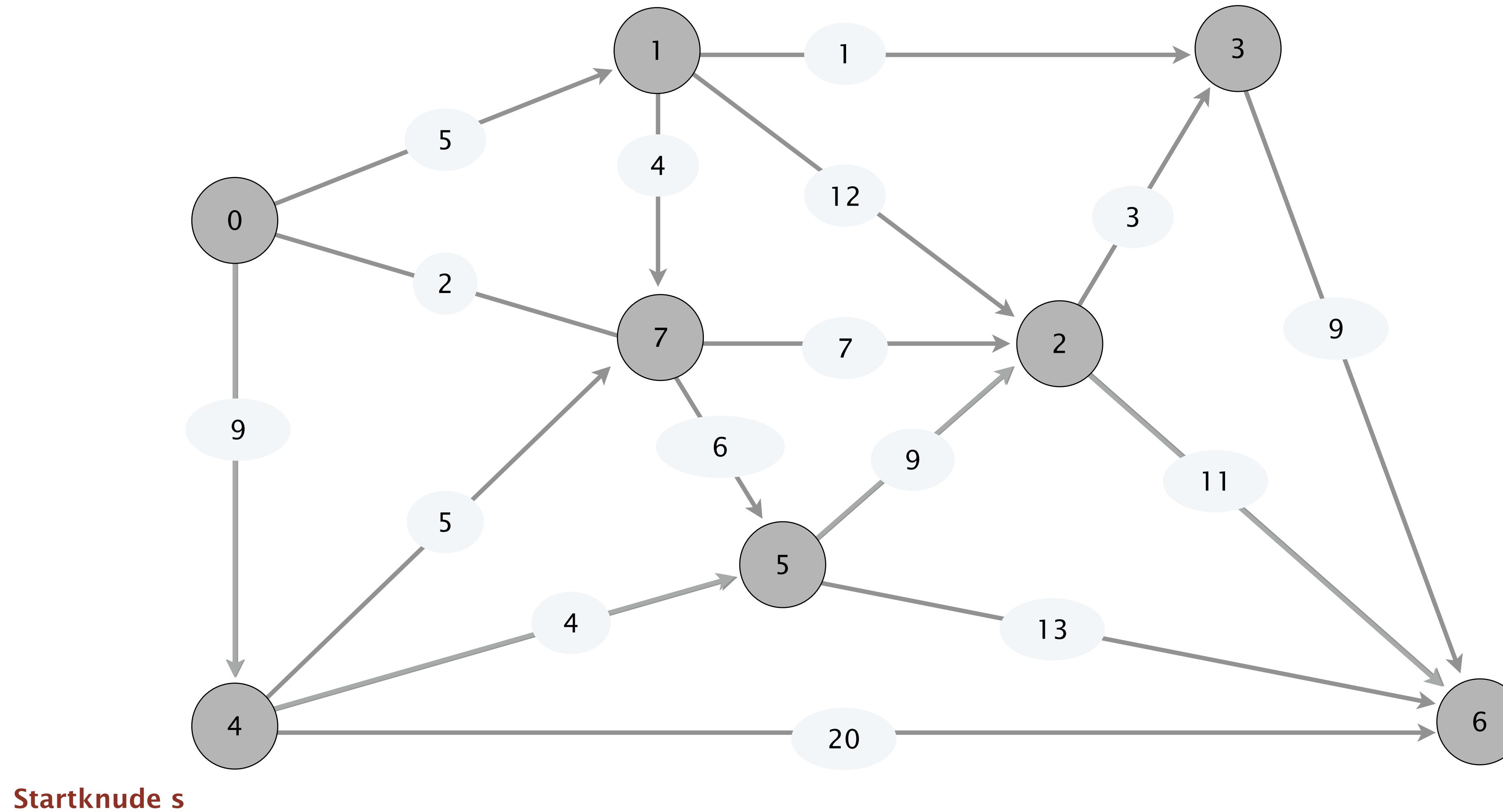
tilføj v til S , og sæt $d[v] \leftarrow \sigma(v)$.

længden af en korteste vej fra s til en knude u , fulgt af kanten $e = (u, v)$

- For at genskabe stien, registrér kanten e hvor minimum opnås: $pred[v] \leftarrow e$



Eksempel på Dijkstra's algoritme



Dijkstra's algoritme: korrekthedsbevis

Invariant. For hver knude $u \in S$ er $d[u] =$ længden af den korteste sti $s \rightsquigarrow u$.

Bevis. [induktion på $|S|$]

Base case: $|S| = 1$ er triviel eftersom $S = \{ s \}$ og $d[s] = 0$.

Induktionshypotese: Antag påstanden er sand for $|S| = i$.

- Lad v være den næste knude, der føjes til S , og lad (u, v) være den sidste kant.
- En korteste vej $s \rightsquigarrow u$ plus (u, v) er en sti $s \rightsquigarrow v$ af længde $\sigma(v)$.
- For en **vilkårlig** anden sti $s \rightsquigarrow v$ (P) skal vises at længden ikke er under $\sigma(v)$.
- Lad $e = (x, y)$ være den første kant i P som forlader S ,
og lad P' være delstien fra s til x .
- P har allerede længde $\geq \sigma(v)$ da den når knude y :

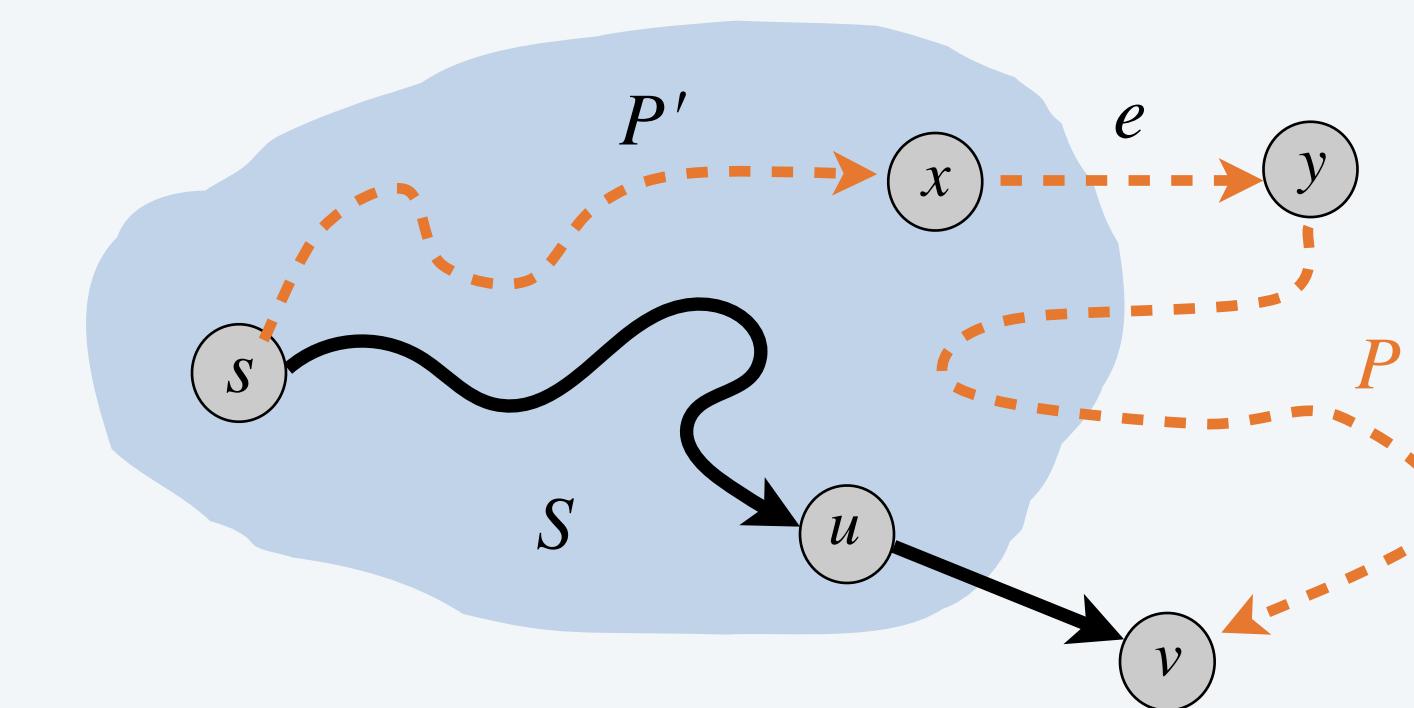
$$\ell(P) \geq \ell(P') + \ell_e \geq d[x] + \ell_e \geq \sigma(y) \geq \sigma(v) \quad ■$$

↑
ikke-negative
længder

↑
induktions-
hypotese

↑
definition
af $\sigma(y)$

↑
Dijkstra valgte v
i stedet for y



Dijkstra's algorithm: effektiv implementation

Første optimering. For hver knude $v \notin S$ vedligeholder vi eksplisit $\sigma[v]$ i stedet for at beregne tallet direkte fra definitionen

$$\sigma(v) = \min_{e = (u,v) : u \in S} d[u] + \ell_e$$

- For hvert $v \notin S$: $\sigma(v)$ kan kun aftage i værdi (fordi mængden S udvides).
- Mere specifikt, antag at u føjes til S og der er en kant $e = (u, v)$ som forlader u .
Så er det tilstrækkeligt at opdatere:

$$\sigma[v] \leftarrow \min \{ \sigma[v], \sigma[u] + \ell_e \}$$


husk: for hver $u \in S$ gælder
 $\sigma[u] = d[u] =$ længden af korteste sti $s \rightsquigarrow u$

Anden optimering. Brug en min-prioritetskø (fx en høb) til at vælge en knude udenfor S der minimerer $\sigma[v]$.

CLRS pseudocode for Dijkstra's algoritme

DIJKSTRA(G, w, s)

INIT-SINGLE-SOURCE(G, s)

$S = \emptyset$

for each vertex $u \in G.V$

INSERT(Q, u)

while $Q \neq \emptyset$

$u = \text{EXTRACT-MIN}(Q)$

$S = S \cup \{u\}$

for each vertex $v \in G.Adj[u]$

RELAX(u, v, w)

if $v.d$ changed

DECREASE-KEY($Q, v, v.d$)

INIT-SINGLE-SOURCE(G, s)

for each $v \in G.V$

$v.d = \infty$

$v.\pi = \text{NIL}$

$s.d = 0$

RELAX(u, v, w)

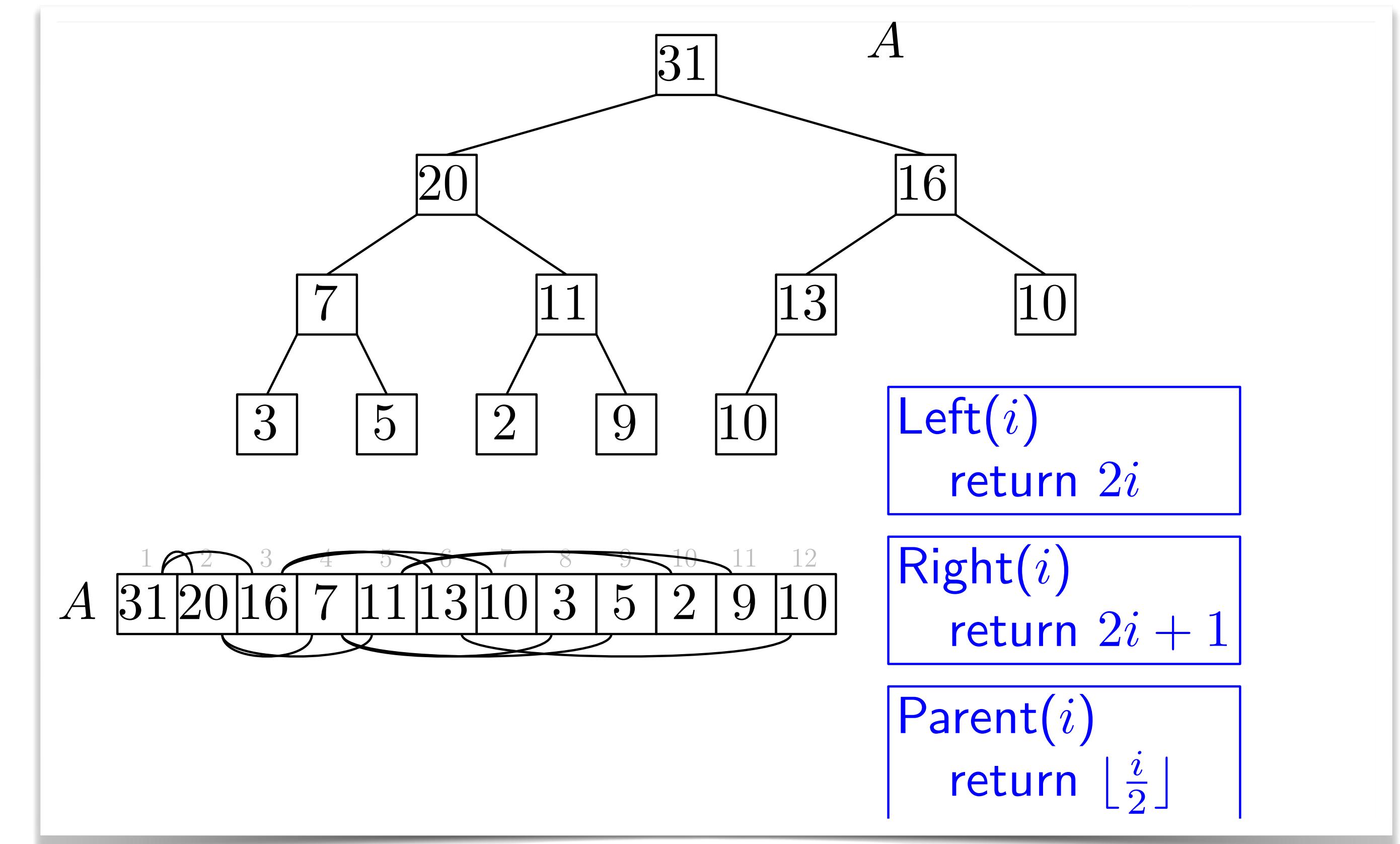
if $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

$v.\pi = u$

Øvelse

- Hvordan kan vi understøtte $\text{Decrease-Key}(Q, v, v \cdot d)$ i en binær min-hob i tid $O(\log n)$?
- Problematik: Hvordan finder vi nøglen v i hoben?
- Det er ok at udvide datastrukturen med mere information, men tiden for operationerne skal bevares



Tidsanalyse for Dijkstra's algoritme

- Graf med n knuder og m orienterede kanter

```

DIJKSTRA( $G, w, s$ )
    INIT-SINGLE-SOURCE( $G, s$ )
     $S = \emptyset$ 
    for each vertex  $u \in G.V$            ←  $O(n)$  tid
        INSERT( $Q, u$ )
    while  $Q \neq \emptyset$ 
         $u = \text{EXTRACT-MIN}(Q)$           ← kaldes højst  $n$  gange
         $S = S \cup \{u\}$ 
        for each vertex  $v \in G.Adj[u]$       ← totalt højst  $m$  iterationer
            RELAX( $u, v, w$ )
            if  $v.d$  changed
                DECREASE-KEY( $Q, v, v.d$ )     ← kaldes højst  $m$  gange
    
```

INIT-SINGLE-SOURCE(G, s)

for each $v \in G.V$

$v.d = \infty$

$v.\pi = \text{NIL}$

$s.d = 0$

← $O(n)$ tid

RELAX(u, v, w)

if $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

$v.\pi = u$

← $O(1)$ tid

- Total tid: $O(n + m)$ plus n gange Extract-Min og m gange Decrease-Key.
- Med en binær hob: $O(n \log n + m \log n)$, eller forsimpleret $O(n + m \log n)$

Dijkstra's algorithm i F#

A screenshot of the ChatGPT web interface. The top bar shows the URL `chat.openai.com/chat`. The main area displays the text "ChatGPT" in large white letters. Below it, there are three small icons: a sun-like icon, a lightning bolt icon, and an exclamation mark icon. A message bubble contains the question "How do I implement Dijkstra's algorithm in F#?". At the bottom, a footer bar reads "ChatGPT Dec 15 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve." On the left side, a sidebar menu includes options like "New Chat", "Light Mode", "OpenAI Discord", "Updates & FAQ", and "Log out".

 Dijkstra.fsx — uge 14 - grafalgoritmer 1

```
4 ▼ let dijkstra (graph : (int * float) list []) (source : int) : (float [] * int []) =
5 ▼     let n = graph.Length                      // number of vertices
6 ▼     let dist = Array.create n infinity        // distances from source to all vertices
7 ▼     let prev = Array.create n (-1)            // previous vertices in the shortest path
8 ▼     let visited = Array.create n false       // whether a vertex has been visited
9 ▼
10 ▼    dist.[source] <- 0.0 // distance to the source vertex is 0
11 ▼    let mutable pq = empty
12 ▼    pq <- insert (source, 0.0) pq
13 ▼
14 ▼    while not (isEmpty pq) do
15 ▼        let u = findMin pq // find the minimum distance vertex
16 ▼        pq <- deleteMin pq // remove it from the queue
17 ▼        if not visited.[u] then
18 ▼            visited.[u] <- true
19 ▼            for (v, w) in graph.[u] do // for each neighbor of u
20 ▼                let alt = dist.[u] + w // compute the distance from u to v
21 ▼                if alt < dist.[v] then // if it is shorter than the current distance
22 ▼                    dist.[v] <- alt // update the distance
23 ▼                    prev.[v] <- u // update the previous vertex
24 ▼                    pq <- insert (v, alt) pq // insert the vertex
25 ▼
26 ▼    (dist, prev) // return array of distances, array of shortest path parents
27 ▼
```

Quiz time!

- Hvad har I lært? Prøv jeres viden af i denne quiz!



En bedre prioritetskø?

Dijkstra køretid. Afhænger af prioritetskø: n INSERT, n DELETE-MIN, $\leq m$ DECREASE-KEY.

- En simpel tabel er optimal for **tætte** grafer. $\longleftarrow \Theta(n^2)$ kanter
- Hobe er meget hurtigere for **tynde** grafer. $\longleftarrow \Theta(n)$ kanter

prioritetskø	INSERT	DELETE-MIN	DECREASE-KEY	total
knude-indekseret tabel ($A[i] = \text{afstand til } i$)	$O(1)$	$O(n)$	$O(1)$	$O(n^2)$
binær høb	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(m \log n)$
Fibonacci høb (Fredman-Tarjan 1984)	$O(1)$	$O(\log n)^\dagger$	$O(1)^\dagger$	$O(m + n \log n)$
heltals prioritetskø (Thorup 2004)	$O(1)$	$O(\log \log n)$	$O(1)$	$O(m + n \log \log n)$

\dagger amortiseret

antager at $m \geq n$

Korteste veje i lineær tid!

Teorem. [Thorup 1999] Der findes en algoritme, der beregner et korteste vej-træ i en uorienteret graf med positive heltalsvægte i tid $O(n+m)$.

NB. Algoritmen udforsker ikke knuderne i afstandsrækkefølge fra s .

Undirected Single Source Shortest Paths with Positive Integer Weights in Linear Time

Mikkel Thorup
AT&T Labs—Research

The single source shortest paths problem (SSSP) is one of the classic problems in algorithmic graph theory: given a positively weighted graph G with a source vertex s , find the shortest path from s to all other vertices in the graph.

Since 1959 all theoretical developments in SSSP for general directed and undirected graphs have been based on Dijkstra's algorithm, visiting the vertices in order of increasing distance from s . Thus, any implementation of Dijkstra's algorithm sorts the vertices according to their distances from s . However, we do not know how to sort in linear time.

Here, a deterministic linear time and linear space algorithm is presented for the undirected single source shortest paths problem with positive integer weights. The algorithm avoids the sorting bottle-neck by building a hierarchical bucketing structure, identifying vertex pairs that may be visited in any order.



Opsummering

- Vi kan effektivt finde korteste veje i (orienterede) grafer med ikke-negative vægte ved brug af Dijkstra's algoritme
- Mange optimeringsproblemer kan formuleres som korteste vej i en graf
- Flere detaljer i CLRS 22.0 og 22.3