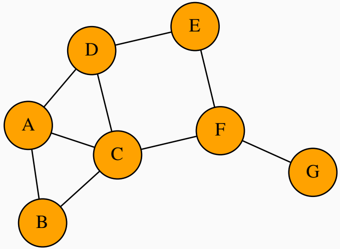
Korteste stier

IN2010 – Algoritmer og Datastrukturer

Uke 39, 2020

Institutt for Informatikk

Repetisjon: Stier og veier



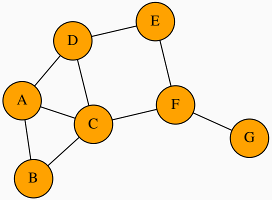
- Sti: En sekvens av noder i grafen forbundet av kanter, der ingen *noder* gjentas. A-B-C-D
- Vei: En sekvens av noder i grafen forbundet av kanter, der ingen *kanter* gjentas. A-B-C-A-D

Hver sti er en vei!

Korteste stier og veier

Hver vei fra en node til en annen inneholder en sti mellom de samme nodene.

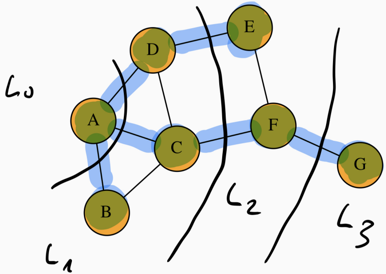
Hvis en node gjentas i en vei, tilsvarer dette å ha gått i en sykel



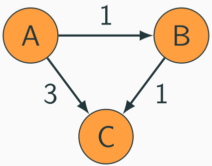
- Vei: A-B-C-A-D
- innholder en sti fra A til D. A-D
- så hver kortest vei kommer til å være en sti!
- så spørsmålet om korteste veier mellom to noder er det samme som korteste stier!

Uvektede grafer: BFS

Når grafen er uvektet, kan vi bruke BFS til å finne korteste stier



Vektede grafer



Nå skal vi se på vektede, rettede grafer. Vi skal se på to algoritmer:

- Dijkstra: kanter med positive vekt
- Bellman-Ford: kanter med både positive og negativ vekt

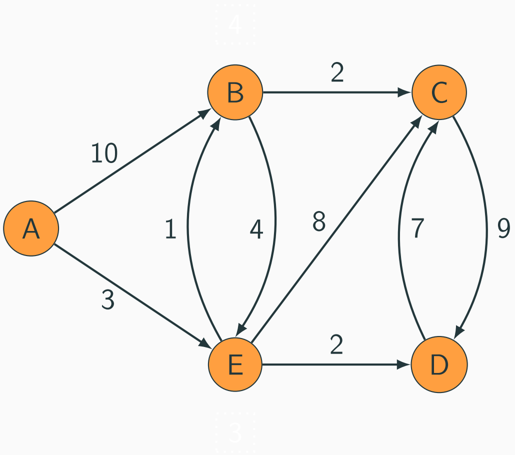
Dijkstra algoritme

Input: Graf G og en startnode s

Ideen: BFS, som tar høyde for kantenes vekt. Vi traverserer grafen fra s som i BFS, men hvor vi bruker en prioritetskø istedenfor en FIFO kø.

- Prioritetskøen skal holde noder som elementer, og bruke en "estimert avstand" D til sammenligning
- D skal inneholde den korteste veien vi har funnet så langt
- s får $D[s] = 0$
- Hver gang vi fjerner et element u fra prioritetskøen, ser vi på naboene v til u og ser om vi har funnet en kortere vei til v, oppdaterer evt. $D[v]$.

Dijkstra algoritme: eksempel



Dijkstra algoritme: Pseudokode

Algorithm 1: Dijktras algoritme

```
1  Procedure Dijkstra(G, s)
2      initialize Q as empty heap
3      for each vertex u in G do
4          D[u] = ∞
5          Q.add(u, D[u])
6      D[s] = 0
7      while Q not empty do
8          v = Q.removeMin()
9          for edge (v, t) in G do
10             if D[v] + w((v, t)) < D[t] then
11                 D[t] = D[v] + w(v, t)
12                 change value of t in Q to D[t]
13     return D
```

Analyse avhengig av implementasjonen. Antar:

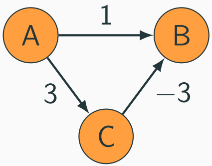
- Bruker naboliste representasjon av G
- binær heap for Q

- 3 (for): går gjennom alle noder i G og legger til heap: $O(|V| \log(|V|))$
- 7 (while): en iterasjon per node i G: $O(|V|)$ iterasjoner.
- 8 (heap-remove): hver remove tar $O(log|V|)$ tid, dermed $O(|V| \log(|V|)$
- 9 (for): en løkke over naboene, som er $deg(v)$ per iterasjon. Tilsammen $O(|E|)$ iterasjoner, med oppdatering i Q i $O(\log(|V|))$. Totalt $O(|E| \log(|V|)$.

Til sammen: $O((|V| + |E|) \log(|V|))$ som er $O(|E| \log(|V|))$

Negative kanter

- grådig fungerer ikke alltid!



- med negative kanter må vi besøke den samme noden og revurdere avstanden flere ganger!
- Men hvor ofte?

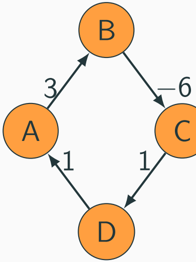
Bellman-Ford: Negative sykler

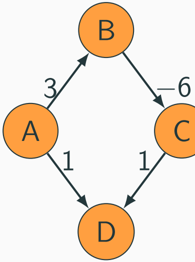
- finner kortest sti i grafer med negative kanter
- hvis grafen inneholder en negativ sykel (der summen av vektene er negativ), *finnes* det ingen kortest sti i den
- Bellman Ford finner kortest sti, eller oppdager negative sykler ved noen smarte triks

Bellman-Ford: Idéen

- den lengste stien som ikke er en sykel inneholder $|V| - 1$ kanter.
- bruker ikke prioritetskø, oppdaterer heller estimert avstand D for alle noder $|V| - 1$ ganger.
- hvis det finnes en node hvor estimert avstand fortsatt blir mindre etter det, inneholder G en negativ sykel.

Bellman-Ford: Eksempel





Bellman-Ford: Pseudokode

Algorithm 2: Bellman-Ford algoritme

```
1  Procedure BellmanFord(G, s)
2      for each vertex u in G do
3          D[u] = ∞
4      D[s] = 0
5      for i from 1 to |V| − 1 do
6          for edge (u, v) in G do
7              if D[u] + w((u, v)) < D[v] then
8                  D[v] = D[u] + w(u, v)
9      for edge (u, v) in G do
10         if D[u] + w((u, v)) < D[v] then
11             return "G has a negative cycle"
12     return D
```

Analyse:

- 2 (for): løkke over antall noder, $O(|V|)$
- 5,6 (for): nøstede for løkker. Den ytterste går $|V| - 1$ ganger, den innerste $|E|$ ganger. Totalt $O(|V| \cdot ||E|)$
- 9 (for): løkke over alle noder, $O(|E|)$

Totalt: $O(|V| \cdot |E|)$