

Mindste udspændende træ

- Introduktion
- Repræsentation af vægtede grafer
- Egenskaber for mindste udspændende træer
- Prims algoritme
- Kruskals algoritme

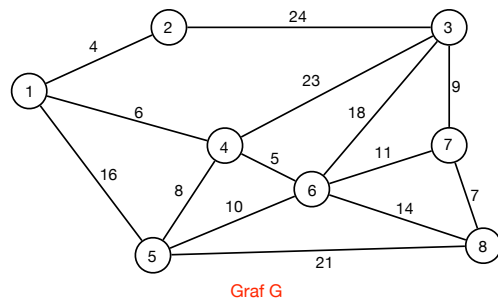
Philip Bille

Mindste udspændende træ

- Introduktion
- Repræsentation af vægtede grafer
- Egenskaber for mindste udspændende træer
- Prims algoritme
- Kruskals algoritme

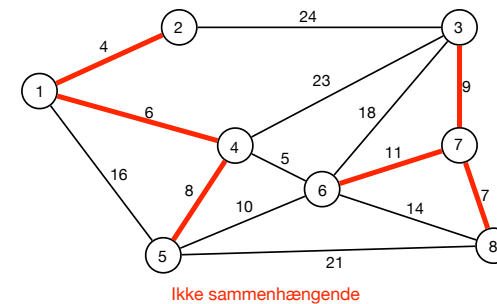
Introduktion

- **Vægtede grafer.** Vægt $w(e)$ på hver kant e i G .
- **Udspændende træ.** Delgraf T af G over alle knuder der er **sammenhængende** og **acyklisk**.
- **Mindste udspændende træ (MST).** Udspændende træ af minimal samlet vægt.



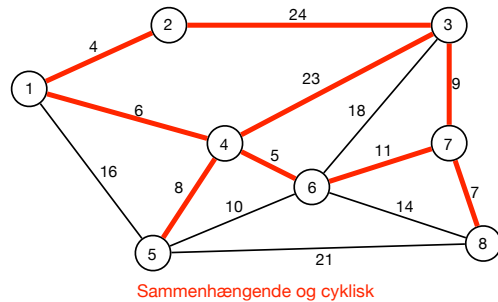
Introduktion

- **Vægtede grafer.** Vægt $w(e)$ på hver kant e i G .
- **Udspændende træ.** Delgraf T af G over alle knuder der er **sammenhængende** og **acyklisk**.
- **Mindste udspændende træ (MST).** Udspændende træ af minimal samlet vægt.



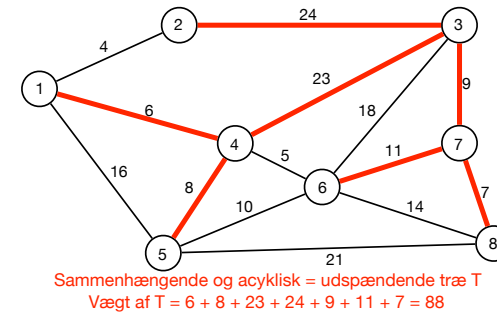
Introduktion

- **Vægtede grafer.** Vægt $w(e)$ på hver kant e i G .
- **Udspændende træ.** Delgraf T af G over alle knuder der er **sammenhængende** og **acyklisk**.
- **Mindste udspændende træ (MST).** Udspændende træ af minimal samlet vægt.



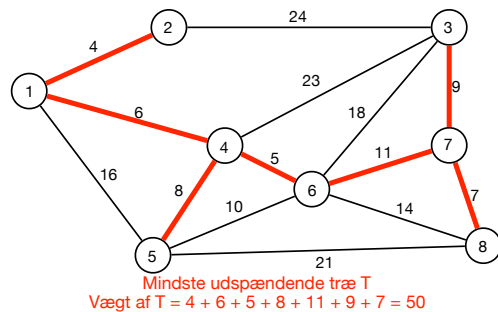
Introduktion

- **Vægtede grafer.** Vægt $w(e)$ på hver kant e i G .
- **Udspændende træ.** Delgraf T af G over alle knuder der er **sammenhængende** og **acyklisk**.
- **Mindste udspændende træ (MST).** Udspændende træ af minimal samlet vægt.



Introduktion

- **Vægtede grafer.** Vægt $w(e)$ på hver kant e i G .
- **Udspændende træ.** Delgraf T af G over alle knuder der er **sammenhængende** og **acyklisk**.
- **Mindste udspændende træ (MST).** Udspændende træ af minimal samlet vægt.



Anvendelser

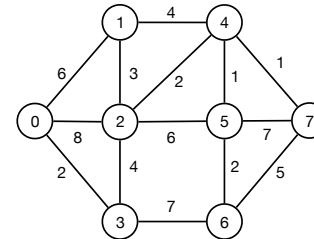
- **Netværk design.**
 - Computer, vej, telefon, elektrisk, kredsløb, kabel tv, hydraulisk, ...
- **Approximationsalgoritmer.**
 - Handelsrejsendes problem, steiner træer.
- **Andre anvendelser.**
 - Meteorologi, kosmologi, biomedicinsk analyse, kodning, billedanalyse, ...
 - Se f. eks. <http://www.ics.uci.edu/~eppstein/gina/mst>

Mindste udspændende træ

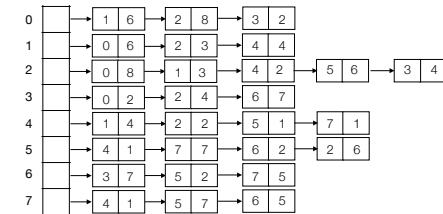
- Introduktion
- Repræsentation af vægtede grafer
- Egenskaber for mindste udspændende træer
- Prims algoritme
- Kruskals algoritme

Repræsentation af vægtede grafer

- Incidensmatrix og incidensliste.
- Tilsvarende for **orienterede** grafer.



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



Mindste udspændende træ

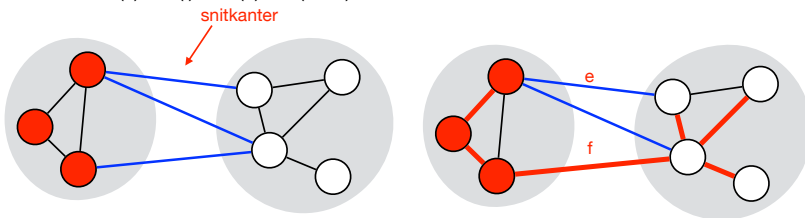
- Introduktion
- Repræsentation af vægtede grafer
- Egenskaber for mindste udspændende træer
- Prims algoritme
- Kruskals algoritme

Egenskaber for MST

- **Simplificerende antagelse.**
 - Alle kantvægte i G er forskellige.
 - G er sammenhængende
- \Rightarrow MST eksisterer og er unik.

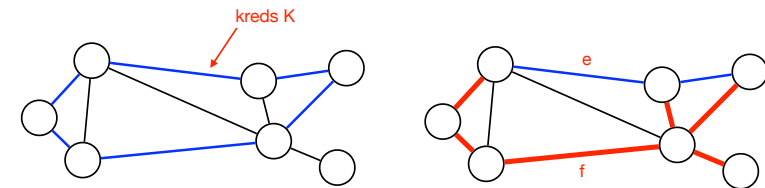
Snitegenskab

- **Def.** Et **snit** er en opdeling af knuderne i to (ikke-tomme) mængder.
- **Def.** En **snitkant** i et snit er en kant med et endepunkt i hver side af snittet.
- **Snitegenskab.** For et vilkårligt snit, er den letteste snitkant med i MST.
- **Bevis.** Modstridsbevis.
 - Antag den letteste snitkant e for et snit ikke er med i MST.
 - Hvis vi tilføjer e til MST laver vi en kreds.
 - \Rightarrow Der er en anden snitkant f i kredsen.
 - \Rightarrow Hvis vi fjerner f og tilføjer e har vi et nyt udspændende træ T .
 - Da $w(e) < w(f)$ er $w(T) < w(\text{MST})$.



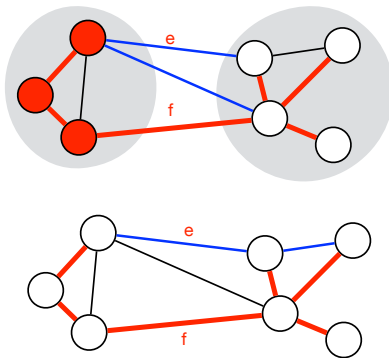
Kredsegenskab

- **Kredsegenskab.** For en vilkårlig kreds, er den tungeste kant i kreds **ikke** med i MST.
- **Bevis.** Modstridsbevis
 - Antag tungeste kant f på en kreds K er indeholdt i MST.
 - \Rightarrow Der er lettere kant e i K , der ikke er indeholdt i MST.
 - Hvis vi fjerner f og tilføjer e har vi et nyt udspændende træ T .
 - Da $w(e) < w(f)$ er $w(T) < w(\text{MST})$.



Egenskaber for MST

- **Snitegenskab.** For et vilkårligt snit, er den letteste snitkant med i MST.
- **Kredsegenskab.** For en vilkårlig kreds, er den tungeste kant i kreds **ikke** med i MST.

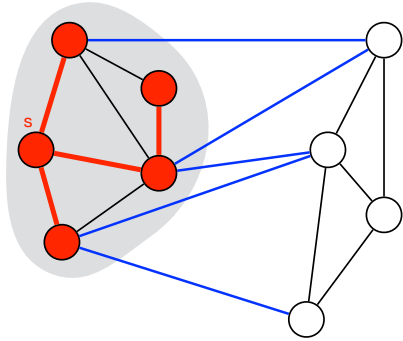


Mindste udspændende træ

- Introduktion
- Repræsentation af vægtede grafer
- Egenskaber for mindste udspændende træer
- Prims algoritme
- Kruskals algoritme

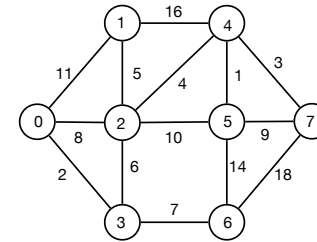
Prims algoritme

- Start ved en knude s og opbyg et træ T fra s .
- I hvert skridt, tilføj **letteste** kant med netop et endepunkt i T .
- Stop når T har $n-1$ kanter.



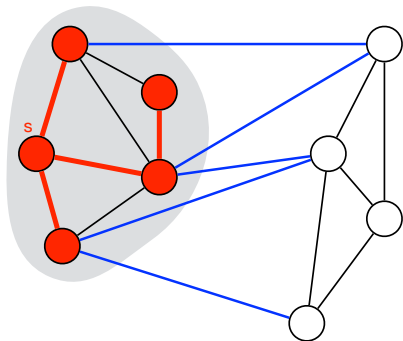
Prims algoritme

- Start ved en knude s og opbyg et træ T fra s .
- I hvert skridt, tilføj **letteste** kant med netop et endepunkt i T .
- Stop når T har $n-1$ kanter.
- **Opgave.** Håndkør Prims algoritme fra knude 0 på følgende graf.



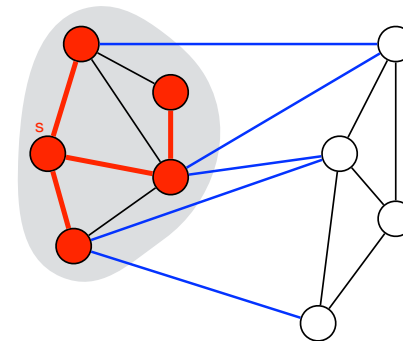
Prims algoritme

- **Lemma.** Prims algoritme beregner MST.
- **Bevis.**
 - Kig på snit mellem besøgte og ikke besøgte knuder i et skridt af algoritme.
 - Vi tilføjer **letteste** snitkant e til T .
 - Snitegenskab \Rightarrow kanten e er i MST $\Rightarrow T$ er MST efter $n-1$ skridt.



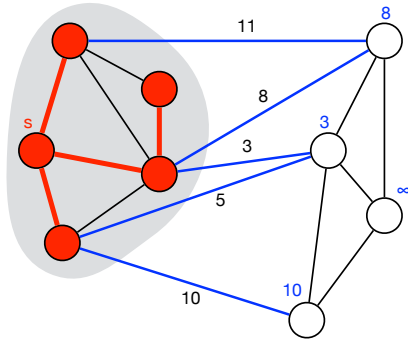
Prims algoritme

- **Implementation.** Hvordan implementerer vi Prims algoritme?
- **Udfordring.** Find den letteste kant med netop et endepunkt i T .



Prims algoritme

- **Implementation.** Vedligehold knuder uden for snit i **prioritetskø**.
 - **Nøgle** af knude v = letteste kant der forbinder v til træet. (∞ hvis ikke forbundet).
- I hvert skridt:
 - Find letteste kant = EXTRACT-MIN
 - Opdater vægt af naboer af ny knude med DECREASE-KEY.

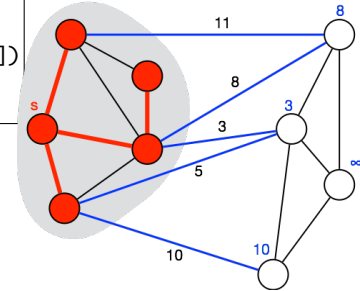


Prims algoritme

```

PRIM(G, s)
for alle knuder v ∈ V
    v.key = ∞
    v.π = null
    INSERT(P, v)
DECREASE-KEY(P, s, 0)
while (P ≠ ∅)
    u = EXTRACT-MIN(P)
    for alle naboer v af u
        if (v ∈ P and w(u, v) < key[v])
            DECREASE-KEY(P, v, w(u, v))
            v.π = u
    
```

- **Tid.**
 - n EXTRACT-MIN
 - n INSERT
 - $O(m)$ DECREASE-KEY
- **Samlet tid med min-hob.** $O(n \log n + n \log n + m \log n) = O(m \log n)$



Prims algoritme

- **Theorem.** Prims algoritme implementeret med en min-hob beregner en MST i $O(m \log n)$ tid.
- **Grådighed.** Prims algoritme er eksempel på en **grådig** algoritme.
 - I hvert skridt, træf det optimale **lokale** valg.
 - \Rightarrow **global** optimal løsning.

Prims algoritme

- **Prioritetskøer og Prim.** Kompleksitet af Prims algoritme afhænger af prioritetskø:
 - n INSERT
 - n EXTRACT-MIN
 - $m + 1$ DECREASE-KEY

Prioritetskø	INSERT	EXTRACT-MIN	DECREASE-KEY	Total
tabel	$O(1)$	$O(n)$	$O(1)$	$O(n^2)$
binær hob	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(m \log n)$
Fibonacci hob	$O(1)^\dagger$	$O(\log n)^\dagger$	$O(1)^\dagger$	$O(m + n \log n)$

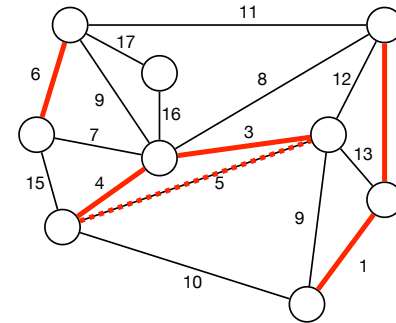
† = **amortiseret** køretid

Mindste udspændende træ

- Introduktion
- Repræsentation af vægtede grafer
- Egenskaber for mindste udspændende træer
- Prims algoritme
- **Kruskals algoritme**

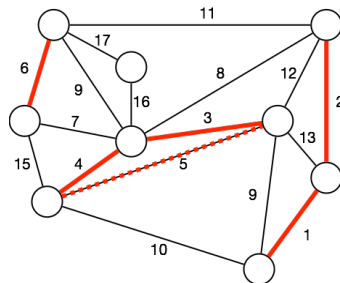
Kruskals algoritme

- Kig på kanter fra letteste til tungeste.
- I hvert skridt, tilføj kant til T hvis den **ikke** medfører en kreds i T.
- Stop når T har $n-1$ kanter.



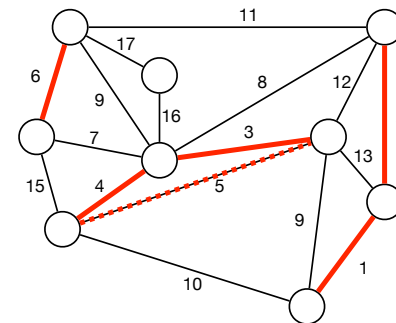
Kruskals algoritme

- **Lemma.** Kruskals algoritme beregner MST.
- **Bevis.**
 - Algoritme kigger på kanter fra letteste til tungest. To tilfælde for en kant $e = (u,v)$.
 - **Tilfælde 1.** e danner kreds og bliver ikke tilføjet til T.
 - e er tungeste kant på kreds.
 - Kredsegenskab $\Rightarrow e$ er ikke med i MST.
 - **Tilfælde 2.** e danner ikke kreds og bliver tilføjet til T.
 - e er letteste kant på snittet givet ved sammenhængskomponenter for u og v .
 - Snitegenskab $\Rightarrow e$ er med i MST.
- $\Rightarrow T$ er MST når $n-1$ kanter er tilføjet.



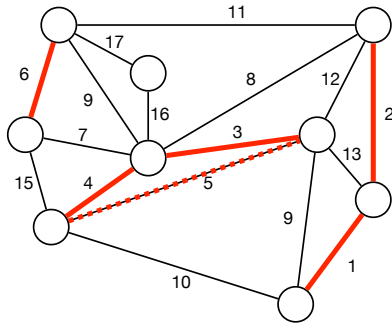
Kruskals algoritme

- **Implementation.** Hvordan implementerer vi Kruskals algoritme?
- **Udfordring.** Afgør om en kant danner en kreds.



Kruskals algoritme

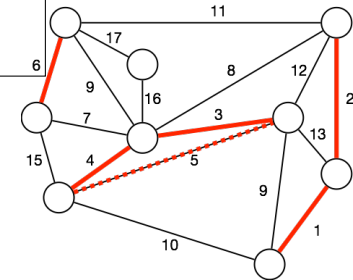
- **Implementation.** Vedligehold kanter i T i datastruktur til **dynamiske sammenhængskomponenter**.
- I hvert skridt:
 - Undersøg om kant danner en kreds = CONNECTED.
 - Tilføj ny kant = INSERT.



Kruskals algoritme

```

KRUSKAL(G)
  Sorter kanter
  INIT(n)
  for alle kanter (u,v) i sort. orden
    if (!CONNECTED(u,v))
      INSERT(u,v)
  return indsatte kanter
    
```



- **Tid.**
 - Sortering af m kanter.
 - 1 INIT
 - m CONNECTED
 - n INSERT
- **Samlet tid med vægtet forening.** $O(m \log m + n + m \log n + n \log n) = O(m \log n)$.

Kruskals algoritme

- **Theorem.** Kruskals algoritme implementeret med vægtet forening beregner en MST i $O(m \log n)$ tid.
- **Grådighed.** Kruskals algoritme er også eksempel på en **grådig** algoritme.

Mindste udspændende træ

- Hvad er den bedste algoritme til MST? Kan man løse problemet i lineær tid?

År	Tid	Opdaget af
???	$O(n \log m)$	Jarnik, Prim, Dijkstra, Kruskal, Boruvka, ?
1975	$O(m \log \log n)$	Yao
1986	$O(m \log^* n)$	Fredman, Tarjan
1995	$O(m)^\ddagger$	Karger, Klein, Tarjan
2000	$O(n\alpha(m,n))$	Chazelle
2002	optimal	Pettie, Ramachandran
20xx	$O(m)$???

\ddagger = randomiseret køretid

Mindste udspændende træ

- Introduktion
- Repræsentation af vægtede grafer
- Egenskaber for mindste udspændende træer
- Prims algoritme
- Kruskals algoritme