MINIMALT SPÄNNANDE TRÄD I VIKTAD GRAF

ETT SPÄNNANDE TRÄD FÖR EN GRAF G ÄR EN DELGRAF TILL G SOM ÄR ETT TRÄD (SAMMANHÄNGANDE, HAR INGA CYKLER) OCH INNEHÅLLER ALLA HÖRN I G.

VIKTEN FÖR ETT SPÄNNANDE TRÄD ÄR SUMMAN AV DOM INGÅENDE KANTERNAS VIKTER.

PRIMS ALGORITM:

INDATA: GRAF $G = \langle V, E \rangle$, KANTVIKTER $f: E \to \mathbb{R}$, STARTHÖRN s

UTDATA: ETT MINIMALT SPÄNNANDE TRÄD FÖR G LAGRAT MED FADERSPEKARE $\pi(u)$

```
PRIM(V, E, f, s) =
  Q ← V
  FÖR VARJE u ∈ Q:
      KEY[u] ← ∞
  KEY[s] ← 0        (Q ÄR EN HEAP MED S ÖVERST)
  π[s] ← NIL
  WHILE Q ≠ ∅ DO
      u ← HeapExtractMin(Q)
      FÖR VARJE GRANNE v TILL u:
          IF v ∈ Q AND f(u,v) < KEY(v) THEN
              π[v] ← u
              KEY[v] ← f(u,v)    (HÄR MÅSTE V FLYTTAS I HEAPEN)
```

TIDSKOMPLEXITET: $\mathcal{O}(|V| \log |V| + |E| \log |V|) = \mathcal{O}(|E| \log |V|)$

Minimalt spännande träd, Kruskal

Kruskals algoritm:

Indata: Graf $G=\langle V,E\rangle$, kantvikter $f: E \to \mathbb{R}$

Utdata: Ett minimalt spännande träd för G lagrat som en kantmängd $A \subseteq E$.

Kruskal(V, E, f) =

```
A ← ∅
för varje u ∈ V
    MakeSet(u)
sortera kanterna i E efter stigande vikt
för varje kant (u,v) ∈ E i stigandeviktsordning:
    if FindSet(u) ≠ FindSet(v) then
        A ← A ∪ {(u,v)}
        Union(u, v)
returnera A
```

Komplexitetsanalys:

MakeSet(u) tar tid $O(1)$

FindSet och Union tar tid $O(\log |V|)$

sorteringen av E tar tid $O(|E| \log |E|)$

Totalt: $O(|V| \cdot 1 + |E| \log |E| + |E| \log |V|) = O(|E| \log |E|)$

om grafen är sammanhängande

Korrekthet för Prim och Kruskal

Idé: Visa att varje kant som läggs till i algoritmen är säker, dvs ingår i något MST.

Definitioner:

- Ett **snitt** (cut) är en delning av V i S och $V-S$.
- En kant **korsar** snittet om ena änden $\in S$ och andra $V-S$.

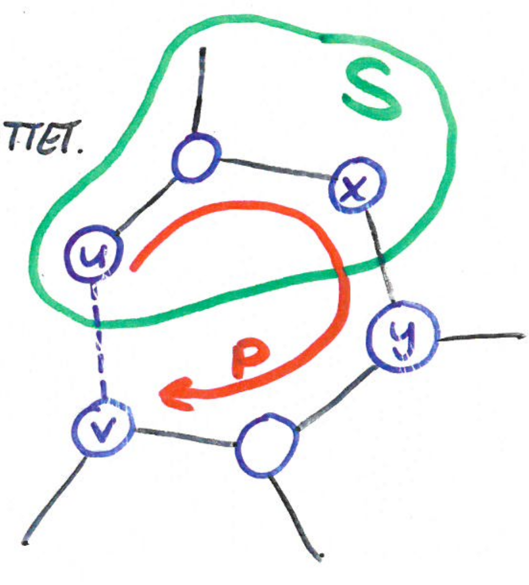
Sats: Givet $G=\langle V,E\rangle$, $f: E \to \mathbb{R}$, $A \subseteq E$, $S \subseteq V$. Om

- det finns ett MST som innehåller A,
- ingen kant i A korsar snittet $(S, V-S)$,
- (u,v) är den lättaste kant som korsar snittet

så är (u,v) säker att lägga till, dvs det finns ett MST som innehåller $A \cup \{(u,v)\}$.

Bevis: Låt T vara MST som innehåller A men inte (u,v).
Konstruera T' som är ett MST och innehåller $A \cup \{(u,v)\}$:

T innehåller stig p mellan u och v.
Det finns kant (x,y) i p som korsar snittet.
Låt $T' = T \cup \{(u,v)\} - \{(x,y)\}$.



T' är uppenbart ett spännande träd

(u,v) är den lättaste korsande kanten

$\Rightarrow f(u,v) \leq f(x,y) \Rightarrow |T'| \leq |T|$

$\Rightarrow T'$ är MST.

PROBLEM: MINIMERA KÖRSTRÄCKAN

MAN HAR MÄTT LÄNGDEN AV VARJE VÄGSTRÄCKA I SVERIGE OCH STOPPAT IN DENNA INFORMATION I EN DATABAS.

NU VILL EN PERSON VETA EXAKT HUR HAN SKA KÖRA FRÅN HUDIKSVALL TILL GRYTHYTTAN FÖR ATT KÖRSTRÄCKAN SKA BLI SÅ LITEN SOM MÖJLIGT.

1. FORMULERA PROBLEMET MATEMATISKT.
2. HITTA EN EFFEKTIV ALGORITM SOM LÖSER PROBLEMET.

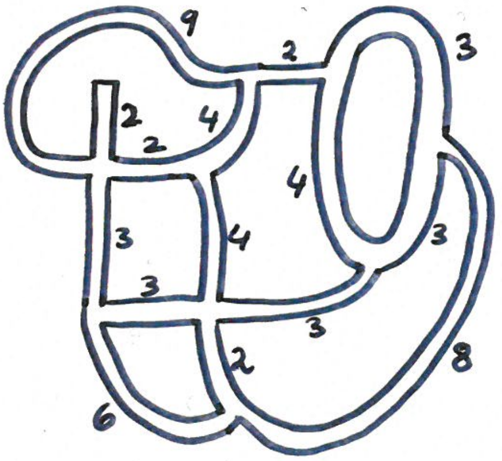
KÖRSTRÄCKEPROBLEMET SOM GRAFPROBLEM

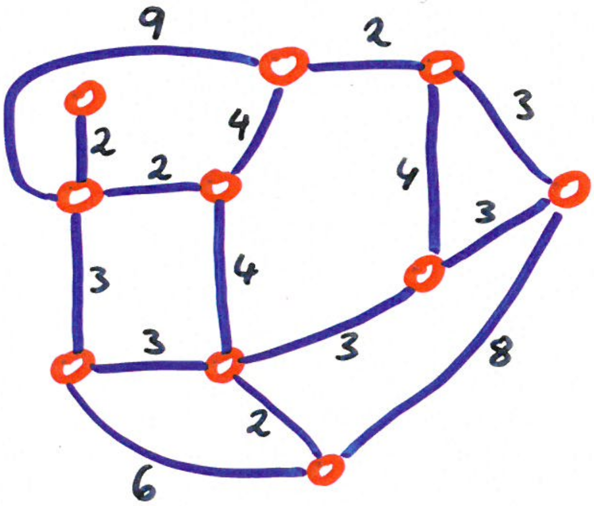
LÅT VARJE VÄGSKÄL MOTSVARAS AV ETT HÖRN OCH VARJE VÄG (MELLAN TVÅ VÄGSKÄL) MOTSVARAS AV EN KANT.

MÄRK VARJE KANT MED MOTSVARANDE VÄGSTRÄCKAS LÄNGD. (VIKTFUNKTION KANTER → $\mathbb{N}$)

NATURLIGA TALEN

EXEMPEL:





PROBLEMFORMULERING:

GIVET EN GRAF $G = \langle V, E \rangle$, EN KANTVIKTFUNKTION $f: E \to \mathbb{N}$ TVÅ HÖRN $s \in V$ OCH $t \in V$

HITTA EN STIG I G FRÅN s TILL t VARS SAMMANLAGDA KANTVIKTSUMMA ÄR MINIMAL.

Algoritm för grafproblemet "kortaste stig"

Dijkstras algoritm:

Indata: $G = \langle V, E \rangle$, $f : E \to \mathbb{N}$, $s \in V$, $t \in V$

Utdata: längden av den kortaste stigen i G från s till t

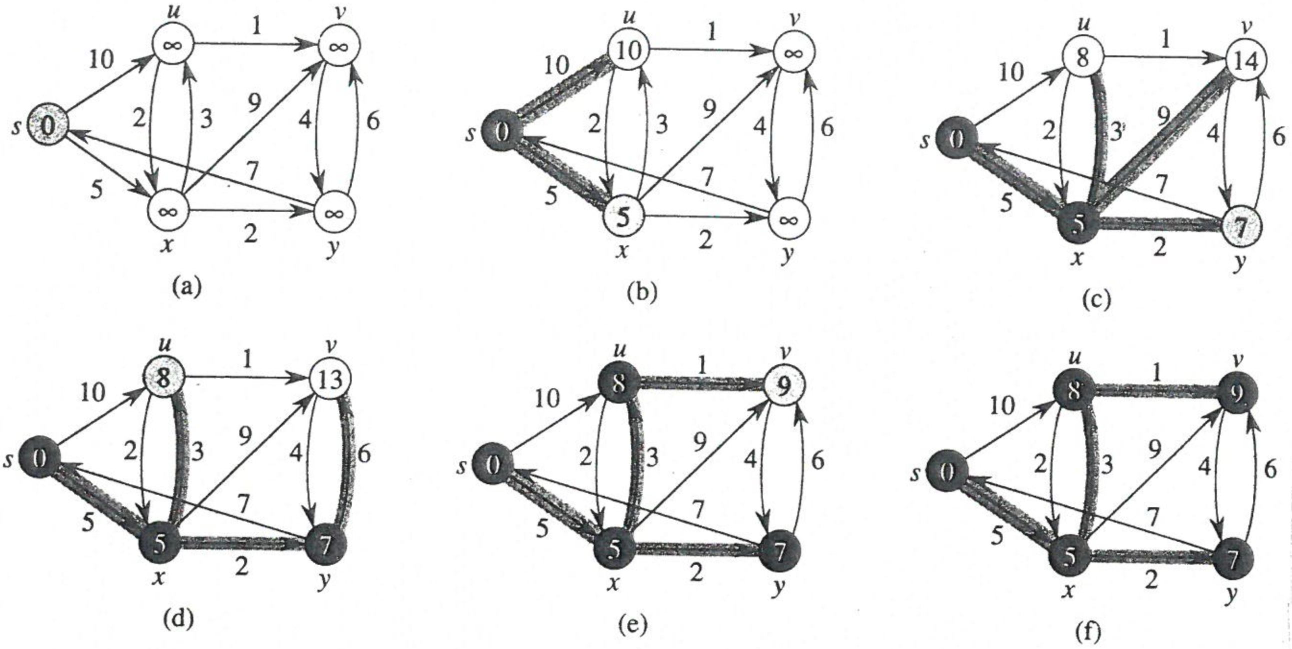
Märk varje hörn med det hittills kortaste kända avståndet från s.

Upprätta en mängd S med dom hörn till vilka den optimala kortaste stigen är känd.

Algoritm:

1. För varje hörn $u \in V$:
   om $(s,u) \in E$ märk u med $f(s,u)$
   annars märk u med ∞
2. Märk s med 0 och låt $S = \{s\}$
3. Så länge $t \notin S$:
   utvidga S med det hörn som är märkt med det kortaste avståndet och uppdatera hörnmärkningen
4. Returnera avståndet som t är märkt med

Exempel på Dijkstras algoritm:



(a) (b) (c) (d) (e) (f)

Analys:

S utvidgas $|V|$ gånger (högst).

Vid varje utvidgning letar man upp det hörn som är märkt med kortaste avståndet: $O(|V|)$

Uppdatering av hörnmärkningen görs högst en gång för varje kant i grafen: $O(|E|)$

Initiering av S och märkningen tar tid $O(|V|+|E|)$

Totalt: $O(|V|^2+|E|+|V|+|E|) = O(|V|^2)$

(eftersom $|E| \in O(|V|^2)$)

Korrekthet för Dijkstras algoritm

Låt $\delta(s,v)$ vara det kortaste avståndet från s till v.
Låt $d[v]$ vara hörnet v-s märkning i ett läge i algoritmen.

Bevisskiss:

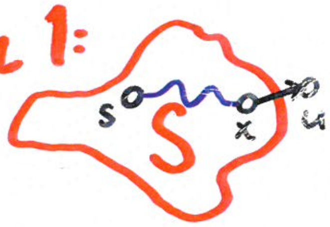
Notera att $d[v] \geq \delta(s,v)$ alltid gäller för alla hörn.

Induktion över S:

Basfall: $S = \{s\}$, $d[s]=0$, $\delta(s,s)=0$ **OK!**

Induktionssteg: Visa att om $d[v]=\delta(s,v)$ för alla $v \in S$ när u just ska läggas till S så är $d[u]=\delta(s,u)$.

Fall 1:



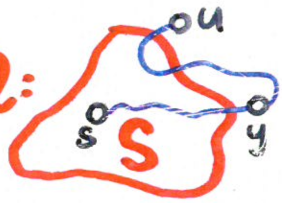
Kortaste stigen från s till u går helt inuti S utom sista kanten (x,u).

Antagandet $\Rightarrow d[x]=\delta(s,x)$

Algoritmen satte $d[u] = d[x]+f(x,u) = \delta(s,x)+f(x,u) = \delta(s,u)$

OK!

Fall 2:



Låt y vara första hörnet utanför S i kortaste stigen från s till u.

Fall 1 $\Rightarrow d[y]=\delta(s,y) \leq \delta(s,u)$

Algoritmen lägger till u före y $\Rightarrow d[u] \leq d[y]$

Vi har nu: $d[y] \leq \delta(s,u) \leq d[u] \leq d[y]$

$\Rightarrow d[y] = \delta(s,u) = d[u]$ **OK!**

Alla hörn som kan nås från s kommer med i S. Övriga har $d[v]=\infty$.