

Föreläsning 12 i ADK20

Grafer: minimala spännande träd

Stefan Nilsson

KTH

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.
- Ett spännande träd med minimal vikt kan beräknas med exempelvis **Prims algoritm** eller **Kruskals algoritm**.

Minimalt spännande träd i viktad graf, Prim

Prims algoritm:

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

Utdata: Ett minimalt spännande träd för G lagrat med förälderpekare $\pi(u)$

function PRIM(V, E, f, s)

$Q \leftarrow V$

for varje $u \in Q$ **do**

$\text{key}[u] \leftarrow \infty$

$\text{key}[s] \leftarrow 0$

$\triangleright$ (Q är en heap med s överst)

$\pi[s] \leftarrow \text{NIL}$

while $Q \neq \emptyset$ **do**

$u \leftarrow \text{HEAPEXTRACTMIN}(Q)$

for varje granne v till u **do**

if $v \in Q$ **and** $f(u, v) < \text{key}[v]$ **then**

$\pi[v] \leftarrow u$

$\text{key}[v] \leftarrow f(u, v)$

$\triangleright$ (Här måste v flyttas i heapen)

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

Minimalt spännande träd i viktad graf, Kruskal

Kruskals algoritm:

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

Utdata: Minimalt spännande träd för G lagrat som en kantmängd $A \subseteq E$

function KRUSKAL(V, E, f)

$A \leftarrow \emptyset$

for varje $u \in V$ **do**

 | MAKESET(u)

 Sortera kanterna i E efter stigande vikt

for varje kant $(u, v) \in E$ i stigande viktsordning **do**

 | **if** FINDSET(u) $\neq$ FINDSET(v) **then**

 | $A \leftarrow A \cup \{(u, v)\}$

 | UNION(u, v)

return A

Komplexitetsanalys:

- $\text{MAKESET}(u)$ tar tid $\mathcal{O}(1)$
- FINDSET och UNION tar tid $\mathcal{O}(\log |V|)$
- sorteringen av E tar tid $\mathcal{O}(|E| \log |E|)$

Totalt: $\mathcal{O}(|V| \cdot 1 + |E| \log |E| + |E| \log |V|) = \mathcal{O}(|E| \log |E|)$ om grafen är sammanhängande

Korrekthet för Prim och Kruskal

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

Definitioner:

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

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

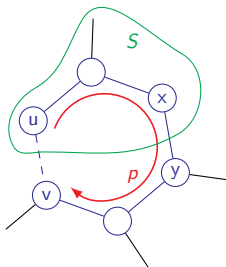
- Det finns ett MST som innehåller A
- Ingen kant i A korsar snittet $(S, V - S)$
- (u, v) är den lättaste kant som korsar snittet

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

Korrekthet för Prim och Kruskal

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 körsträcken

Man har mätt längden av varje vägsträcka i Sverige och stoppat in denna information i en databas.

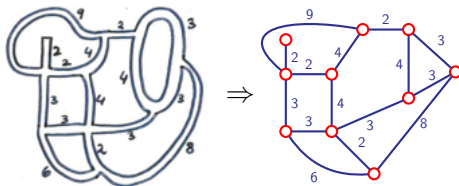
Nu vill en person veta exakt hur han ska köra från Hudiksvall till Grythyttan för att körsträcken ska bli så liten som möjligt

- 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 $\rightarrow \mathbb{N}$)

Exempel:



Problemformulering:

Givet en graf $G = \langle V, E \rangle$, en kantviktfunktion $f : E \rightarrow \mathbb{N}$, 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: $G = \langle V, E \rangle$, $f : E \rightarrow \mathbb{N}$, $s \in V$, $t \in V$

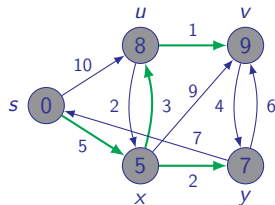
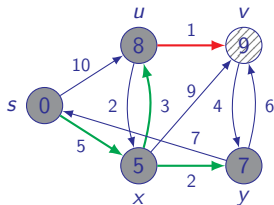
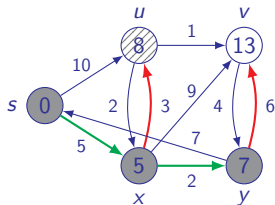
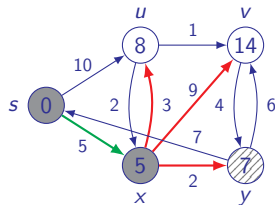
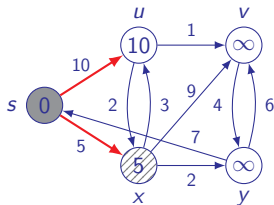
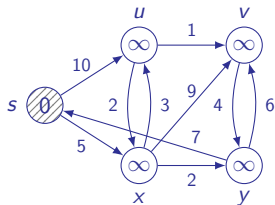
Utdata: Längden av den kortaste stigen i G från s till t

- 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



Analys av Dijkstras algoritm

- S utvidgas $|V|$ gånger (högst)
- Vid varje utvidgning letar man upp det hörn som är märkt med kortaste avståndet: $\mathcal{O}(|V|)$
- Uppdatering av hörnmärkningen görs högst en gång för varje kant i grafen: $\mathcal{O}(|E|)$
- Initiering av S och märkningen tar tid $\mathcal{O}(|V| + |E|)$
- Totalt: $\mathcal{O}(|V|^2 + |E| + |V| + |E|) = \mathcal{O}(|V|^2)$ (eftersom $|E| \in \mathcal{O}(|V|^2)$)

Korrekthet för Dijkstras algoritm

- Låt $\delta(s,v)$ vara det kortaste avståndet från s till v
- Låt $d[v]$ vara hörnets 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:** $S = \{s\}$, $d[s] = 0$, $d(s,s) = 0$
- **Induktionssteg:** Visa att om $d[u] = \delta(s,u)$ för alla $v \in S$ när u just ska läggas till S så är $d[u] = \delta(s,u)$

- 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 $d[u] = d[x] + f(x,u) =$

$$\delta(s,x) + f(x,u) = \delta(s,u)$$

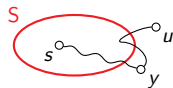
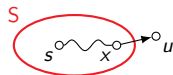
- 2 Låt y vara första hörnet utanför S i kortaste stigen från s till u

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: $d[y] \leq \delta(s,u) \leq d[u] \leq d[y] \Rightarrow$

$$d[y] = \delta(s,u) = d[u]$$



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