



NTNU

Algoritmer og Datastrukturer
~~Innovation and Creativity~~

Minimale spenntrær, SCC og topologisk sortering.

Simon Jonassen

Institutt for Datateknikk og Informasjonsvitenskap

19. September 2006

Minimale spenntrær

Gitt en graf $G = (V, E)$. Et minimalt spenntrær er en asyklisk delmengde T , av alle elementer $(u, v) \in E$ som sammenkobler alle noder og som har $w(T) = \sum_{(u,v) \in T} w(u, v)$ minimal.

Veldig generelt:

- Generisk-MST: Vi bygger ut løsningen ved å legge til trygge kanter.
- Prim: Vi starter med en node og bygger ut løsningen ved å legge til lette kanter vi oppdager etter hvert.
- Kruskal: Vi bygger ut løsningen ved å legge til kanter etter økende vekt uten å danne sykler.

Minimale spenntrær: Generisk

Generisk MST (Cormen)

Generic-MST(G, w)

$A \leftarrow \emptyset$

while A does not form a spanning tree

 find an edge (u, v) that is safe for A

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

return A

Forutsetning: for hvert steg er A en delmengde av et MST til G . En kant er trygg hvis den kan legges til uten å bryte forutsetningen over.

Minimale spenntær: Prim

Prim (Cormen)

```
MST-Prim( $G, w, r$ )
  foreach  $u \in V[G]$ 
     $key[u] \leftarrow inf$ 
     $\pi[u] \leftarrow nil$ 
   $key[r] \leftarrow 0$ 
   $Q \leftarrow V[G]$ 
  while  $Q \neq \emptyset$ 
     $u \leftarrow ExtractMin(Q)$ 
    foreach  $v$  adjacent to  $u$ 
      if  $v \in Q$  and  $w(u, v) < key[v]$ 
         $\pi[v] \leftarrow u$ 
         $key[v] \leftarrow w(u, v)$ 
```

Minimale spenntrær: Prim

Hva med følgende?

Prim

Prim(G):

- La $G=(E,V)$ være en graf, hvor E er alle kanter og V er alle noder i G . La w være en vektfunksjon.
- La C være en tom delmengde av V , legg en vilkårlig node inn i C .
- La A være en tom delmengde av E .
- Fortsett inntil $C=V$:
 - Velg en kant $e'=(u',v')$ av alle $e=(u,v)$ hvor $u \in C$ og $v \notin C$ med $w(e')$ minimal. Legg e' til A og legg u' til C .
- A er et MST til G .

Minimale spenntrær: Prim

Observasjoner

- under alle steg er A en delmengde av et MST.
- under alle steg velger vi den korteste kanten av de som starter i C og slutter utenfor C .

Teor. 23.1, s. 563

Anta at $G = (V, E)$ er en sammenkoblet ikke-rettet graf med en viss funksjon w definert for alle elementer i E . Anta også at A er en delmengde av E som er inkludert i et MST for G og $(S, V - S)$ er et utsnitt av G som respekterer A . Hvis en kant (u, v) er en lett kant som krysser $(S, V - S)$, så er den trygg for A .

Minimale spenntrær: Prim

Generisk-MST: Vi bygger ut løsningen ved å legge til trygge kanter.

Konklusjoner

- A er en delmengde av et MST
 - Kanten som legges til er en "trygg" kant
- Prim er en spesiell versjon av Generic-MST.

Minimale spenntrær: Prim

Kjøretid

- tabelloppsett: $O(V)$
- heap-oppsett: $O(V)$
- ekstrahering av hjørner fra en heap: $O(\log V)$ $O(V)$ ganger.
 $O(V \log V)$ totalt.
- nabonodeoppslag: $O(E)$ ganger med oppdatering av heapinnhold $O(\log V)$. Totalt $O(E \log V)$ i Worst-Case.
- Totalt: $O(V + V + V \log V + E \log V) = O(V \log V + E \log V)$.
- Forbedringer: med Fibonacci-heap tar en heap-oppdatering $O(\log(1))$. Dermed får vi:
 $O(V + V + V \log V + E) = O(E + V \log V)$

Minimale spenntær: Kruskal.

Kruskal (Cormen)

MST-Kruskal(G, w)

$A \leftarrow \emptyset$

foreach vertex $v \in V[G]$

$MakeSet(v)$

sort elements of E into nondecreasing order by w .

foreach $(u, v) \in E$

 if $FindSet(v) \neq FindSet(u)$

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

$Union(u, v)$

return A

Minimale spenntrær:Kruskal

Hva med følgende?

Kruskal

Kruskal(G,w):

- La $G=(E,V)$ være en graf, hvor E er alle kanter og V er alle noder i G .
- La A være en tom delmengde av E .
- La K være en kopi av E .
- Fortsett inntil K er tom:
 - Fjern en kant e med lavest vekt $w(e)$, fra K .
 - Dersom e danner ingen sykler med alle elementer i A legg e til A .
- A er et MST til G .

Minimale spenntrær:Kruskal

Observasjoner

- under alle steg er A en delmengde av et MST.
- under alle steg velger vi den korteste kanten som kobler sammen to ulike komponenter (deltrær) i A .

Cor. 23.2, s. 565 i Cormen

Anta at $G = (V, E)$ er en sammenkoblet ikke-rettet graf med en viss funksjon w definert for alle elementer i E . Anta også at A er en delmengde av E som er inkludert i et MST for G og at $C = (V_C, E_C)$ er en sammenkoblet komponent av et skog $G_A = (V, E)$. Hvis (u, v) er en lett kant som kobler C til en annen komponent i G_A , så (u, v) er trygg for A .

Minimale spenntrær:Kruskal

Generisk-MST: Vi bygger ut løsningen ved å legge til trygge kanter.

Konklusjoner

- A er en delmengde av et MST
- Kanten som legges til er en "trygg" kant

Kruskal er en spesiell versjon av Generic-MST.

Minimale spenntrær: Kruskal

Kjøretid

- sortering: $O(E \log E)$
- sykelsjekkning: MakeSet $O(V)$ ganger og FindSet $O(E)$ ganger. Totalt blir det $O((V + E)\alpha(V))$ (se Kap 21.4 i Cormen)
- Totalt: $O(E \log E) + O((V + E)\alpha(V)) = O(E \log V^2) + O(E \log V) = O(E \log V)$.

Strongly Connected Components

Definisjon

SCC av en graf $G = (V, E)$ er en maksimal mengde av alle hjørner $C \subseteq V$ slik at for hvert par av hjørner $u, v \in C$ har vi både $u \rightsquigarrow v$ og $v \rightsquigarrow u$ (dvs. begge hjørner er oppnåelige fra hverandre).

Algoritme

- Kjør DFS på G og finn alle endetider på G .
- Kjør DFS på G^T men start etter synkende endetid
- Resulterende dybde først skog separerer SCC'er.

Strongly Connected Components

litt teori...

- Anta at det finnes to SCC, C og C' , i en graf $G = (V, E)$ hvor $u, v \in C$ og $u', v' \in C'$. Hvis det finnes en sti fra u til u' finnes det ingen sti fra v' til v i G . (Lemma 22.13, s.554 i Cormen).
- Hvis C og C' er to SCC i en rettet graf $G = (V, E)$. Anta at det finnes en kant $(u, v) \in E$ hvor $u \in C$ og $v \in C'$. I så fall $f(C) > f(C')$. (Lemma 22.14, s. 555 i Cormen).
- Hvis C og C' er to SCC i en rettet graf $G = (V, E)$. Anta at det finnes en kant $(u, v) \in E^T$ hvor $u \in C$ og $v \in C'$. I så fall $f(C) < f(C')$. (Cor. 22.14, s. 555 i Cormen).

Topologisk Sortering

Problemstilling

Finner en ordning av alle noder i en DAG slik at alle noder som starter en kant listes opp før alle noder som avslutter den.

Algoritme

- Kjør DFS på alle noder og legger dem i front av en liste når vi blir ferdige med dem.