

# Mindst udspændende træer

Mads Øhm Larsen

6. april 2008

## 1 Definition af problemet

Vi vil gerne finde det træ i en vægtet graf således at summen af vægtene er mindst mulige.

## 2 Definition af et udspændende træ

Et udspændende træ i en graf er et træ der gør at alle knuder i grafen hænger sammen.

## 3 Definition af mindst udspændende træ

Hvis vi har en graf defineret ved  $G = \{V, E, c\}$ , hvor  $V$  er knuderne,  $E$  er kanterne og  $c$  er den funktion der fører en fra knude til knude, så er det MST til grafen defineret ved at koble alle knuderne sammen således at kosten er minimal. Der vil altid være  $n - 1$  kanter i et MST, hvor  $n$  er antallet af knuder.

## 4 Generiske algoritme og dennes korrekthed

Vi gør brug af snit til at finde et MST. Et snit er når vi kan ligge en linie gennem grafen, og stadig respektere settet  $A$ , som indeholder kanterne. Algoritmen følger følgende løkke invariant: "Før hver iteration er  $A$  et subset af noget minimum udspændende træ". Ved hvert skridt finder vi en kant  $(u, v)$  som kan tilføjes  $A$  uden at overtræde invarianten, altså skal  $A \cup (u, v)$  også være et MST. En sådan kant hedder en "safe-edge", da den sikkert kan lægges til  $A$ .

Algoritmen kører indtil  $A$  er et udspændt træ. For hver iteration tager vi en kant der sikkert kan lægges til  $A$  og lægger den til. Har vi en sammenhængende graf, som vi skal have for at kunne finde et MST, ved vi jo at alle knuder hænger sammen, og derfor må alle knuder, på et eller andet tidspunkt være en "safe-egde" til  $A$ . Derfor kan dette lade sig gøre.

### 4.1 Korrekthed

Vi skal bruge 3 ting for at vise hvordan dette virker:

1. En kant krydser et snit,  $(S, V - S)$ , hvis det ene af dets endepunkter er i  $S$  og det andet i  $V - S$

2. Et snit repektore  $A$  hvis ingen af de kanter der er i  $A$  krydser snittet
3. En kant er en *let-kant* hvis det er den kant med mindst vægt der krydser snittet

Lad  $G$  være en sammenhængende, ikke-orienteret graf med en vægt funktion. Lad  $A$  være et subset af  $E$  der er inkluderet i et MST for  $G$  og lad  $(S, V - S)$  være et snit der repektore  $A$ . Lad derefter  $(u, v)$  være et let-kant, så er  $(u, v)$  sikker for  $A$ .

**Bevis:** Lader vi  $T$  være et MST som indeholder  $A$ . Lader vi  $T'$  være et andet MST, der også indeholder  $A$ , men ikke  $(u, v)$  kan vi, ved hjælp af klip-og-klisteteknikken vise at  $(u, v)$  er et safe-edge. Laver vi et snit, således at  $(u, v)$  har endepunkter på hver side, må der findes en anden knude der også har det i  $T$ . Fjerner vi denne kant, og lader  $(u, v)$  være et let kant, kan vi vise at også  $T'$  må være et MST. Vægten for de to knuder må være  $w(u, v) \leq w(x, y)$ . Men dette gør at  $w(T') \leq w(T)$ . Da  $T$  var et MST, må det dog forholde sig at  $w(T) \leq w(T')$ . Altså må de være ens, og vi har vist at  $T'$  må være et MST.

Vi mangler dog at vise at  $(u, v)$  er en sikker kant. Vi har  $A \subseteq T'$ , da  $A \subseteq T$  og  $(x, y) \notin A$ . Altså  $A \cup (u, v) \subseteq T'$ . Derfor, siden  $T'$  er et MST, er  $(u, v)$  en sikker kant.

## 5 Kruskal's algoritme

Denne basere sig direkte på den generiske algoritme, da også denne tager en sikker kant og tilføjer til træet. Vi tager en liste af kanter, og sortere dem i ikke-faldende orden efter vægt. Herefter gennemgås denne liste, og laver kanten ikke en cyklus i træet bliver den tilføjet.

Dette kan implementeres med disjunkte-mængder, hvor hver mængde er et træ. Vi kan derfor finde ud af om to kanter er i samme træ ved at lave en FIND-SET operation på dem. Og vi kan lægge kanten til vha. UNION. Kruskal's algoritme er grådig, da vi altid laver det bedste valg lige nu og her!

### 5.1 Køretidskompleksitet

Vi antager at vi bruge disjunkte-mængder med path-compression og union-by-rank. Det vil tage os  $O(E \lg E)$  at sortere kanterne. Vi bruger  $O(E)$  FIND-SET og UNION operationer, sammen med  $|V|$  MAKE-SET, tilsammen  $O((V + E)\alpha(V))$  tid, hvor  $\alpha$  er en meget langsomt voksende funktion. Siden grafen er forbundet har vi  $|E| \geq |V| - 1$ , så vores disjunkte-mængder operationer tager  $O(E\alpha(V))$  tid og siden  $\alpha(V) = O(\lg V) = O(\lg E)$  har vi at Kruskal's algoritme tager  $O(E \lg E)$  tid.

## 6 Prim's algoritme

Prim's algoritme er igen et special tilfælde af den generiske algoritme. Denne virker ved at vi lægger snittet således, at den ene mængde indeholder alle de kanter der er i  $A$ , og den anden indeholder alle de kanter der endnu ikke er i  $A$ . Derefter tilføjer vi den kant med mindst vægt og gentager.

### 6.1 Køretidskompleksitet

Prim's algoritme kan implementeres ved hjælp af min-hobe, hvor vi alle kanter der ikke er i  $A$  være en knude i min-hoben med nøgle der svarer til den mindste vægt til en knude i  $A$ . Hver gang vi tilføjer et nyt element i hoben, skal denne genopbygges ( $O(|V|)$ ). Det tager  $O(\lg |V|)$  at udtrække en knude og lige så at indsætte en knude. Alle kanter bliver set på én gang, altså  $O(E \lg |V|)$ .