Øving 9, teori: Minimale spenntrær

Dette dokumentet beskriver et løsningsforslag til teoriøving 9, 2018.

Question 1: Tidligere eksamensoppgave

Hva inneholder prioritetskøen som brukes i Prims algoritme?

- Noder ✓
- Kanter

Question 2: Disjoint Set Forest

Hvilke påstander stemmer om Disjoint Set Forest-datastrukturen (side 568 i boka)?

- Rangen *u.rank* for en node *u* er en øvre grense for høyden til *u* ✓
- Rangen *u.rank* for en node *u* er nøyaktig lik høyden til *u*
- Etter Find-Set(x) vil alle noder i treet som x tilhører ha samme forelder x.p
- Uten stikomprimeringsheuristikken vil ikke Find-Set(x) finne riktig representant

Rank ville vært nøyaktig lik høyden hvis det ikke var for stikompresjon. Med stikompresjon blir det en øvre grense i stedet. Find-Set(x) vil kun komprimere stien fra x til representantnoden. Treet som x tilhører kan inneholde andre grener enn den grenen x befinner seg i som ikke komprimeres. Stikomprimeringsheuristikken brukes kun for å forbedre kjøretiden, og påvirker ikke korrektheten til Find-Set(x).

Kommentar: Denne oppgaven hadde tidligere som riktig svar "Rank er en øvre grense for treets høyde" som er feilformulert. Rank defineres kun for noder, som reflekteres i den nye formuleringen.

Question 3: Prim og Kruskal

Under forløpet til Prim og Kruskal kaller vi mengden med kanter som foreløpig er valgt A. For hvilke algoritmer kan A inneholde mer enn ett tre om gangen?

- Bare Prim
- Prim og Kruskal
- Ingen av dem
- Bare Kruskal ✓

For Prim utgjør kantene i A alltid et sammenhengende tre.

Question 4: Prim og Kruskal

Hvilke påstander stemmer?

- Kruskal velger alltid en lett kant som den neste kanten. ✓
- I Prims algoritme henter Extract-Min(Q) ut noden u fra prioritetskøen Q med lavest verdi for feltet u.π
- Kruskal er en grådig algoritme. ✓
- Prim er en grådig algoritme. ✓

Extract-Min(Q) henter ut noden u med lavest verdi for feltet u.key, ikke u.π.

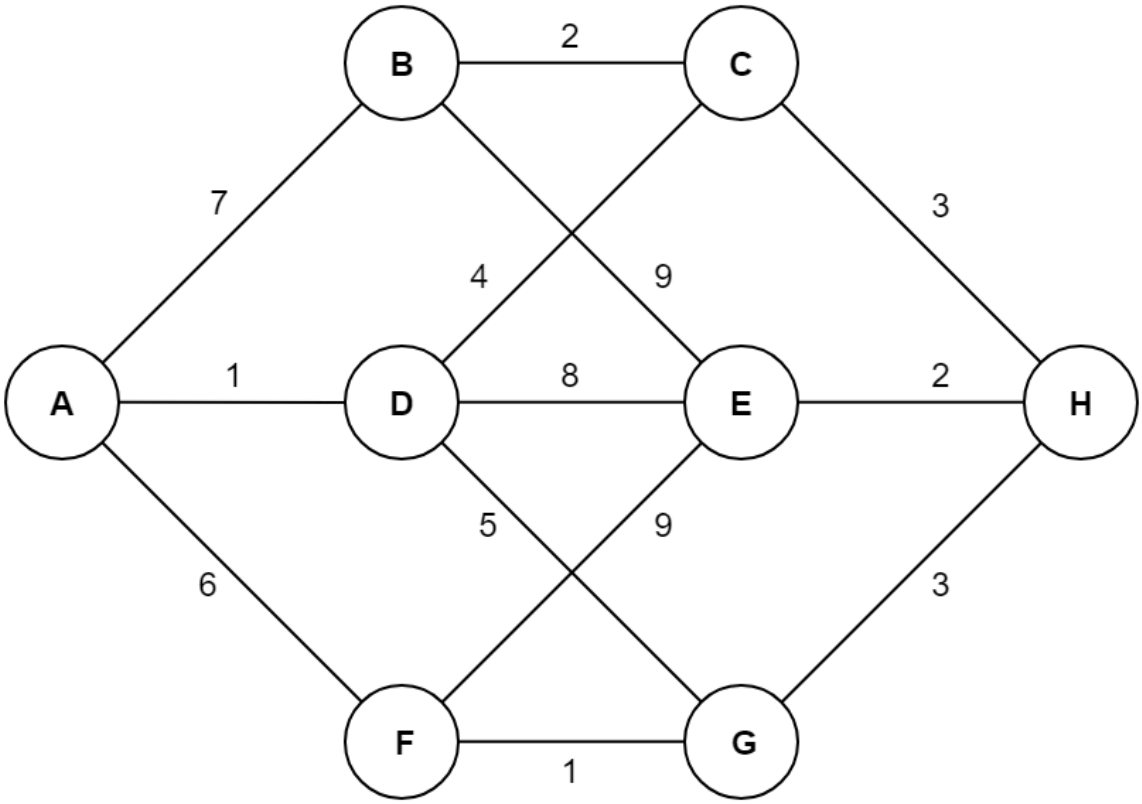
Question 5: Kruskals algoritme

Hva skjer dersom man unnlater å sortere kantene i Kruskals algoritme?

- Man får et minimalt spenntre.
- Man får ikke et spenntre.
- Man får et spenntre. ✓

Question 6: Finne minimale spenntrær

Hva blir det minimale spenntreet for figuren under?



- A-B, B-C, C-H, F-G, G-H, D-E, D-C
- A-D, D-E, E-H, C-H, B-C, F-G, G-H
- A-D, B-C, D-C, D-G, F-G, D-E, E-H
- A-D, D-C, B-C, C-H, G-H, F-G, E-H ✓

Question 7: Kruskals algoritme

Dersom du bruker Kruskals algoritme for å finne det minimale spenntreet i oppgave 6, hvilken kant velges som den syvende kanten?

- A-D
- D-G
- C-H
- C-D ✓

Question 8: Prims algoritme

Dersom du bruker Prims algoritme for å finne det minimale spenntreet i oppgave 6, hvilken kant velges som den femte kanten? Start i node A.

- E-H ✓
- E-F
- C-H
- F-G

Question 9: Minimale spenntrær

Et lite land langt, langt borte består av en mengde øyer. Innbyggerne har lenge samlet inn penger for å få veiforbindelse mellom alle øyene. Prislappen på broene er kun avhengig av broenes lengder og de ønsker derfor å binde sammen øyene med kortest mulig samlet brolengde. Hvor mange broer vil du trenge dersom det er n øyer?

- n/2
- n
- (n−1)n/2
- n−1 ✓

Svaret her er å lage et minimalt spenntre. Da er alle nodene (øyene) koblet sammen (et spenntre er en sammenhengende graf) og total kostnad er minimert. Et minimalt spenntre har alltid n-1 kanter.

Question 10: Minimale spenntrær

Hvis kanten (u,v) er den kanten med lavest vekt i en sammenhengende graf, så er (u,v) med i ett og bare ett minimalt spenntre av grafen.

- Usant ✓
- Sant

Enkelt å finne et moteksempel.

Question 11: Minimale spenntrær

Hvis kanten (u,v) har lavere vekt enn alle de andre kantene i en sammenhengende graf, så er (u,v) med i alle minimale spenntre av grafen.

- Sant ✓
- Usant

Kruskals algoritme, som alltid vil velge denne kanten, tilbyr oss intuisjon om at dette sannsynligvis er riktig.

Du kan bevise egenskapen formelt gjennom bevis ved kontradiksjon. Vi antar det motsatte, og lar T betegne et minimalt spenntre som ikke inneholder kanten (u, v). Dersom (u, v) så legges til i T, vil det nødvendigvis oppstå en sykel. Vi kan fjerne hvilken som helst kant fra denne sykelen og sitte igjen med et spenntre. For å minimere vektene i spenntreet, fjerner vi kanten fra sykelen med høyest kantvekt. Denne kanten har nødvendigvis høyere vekt enn (u, v), siden (u, v) har den laveste vekten i grafen. Dermed har vi forbedret T ved å substituere inn kanten (u, v), og T kan derfor ikke ha vært minimalt til å begynne med. Beviset fullføres ved kontradiksjon.

Question 12: Minimale spenntrær

Hvis alle kantene i en sammenhengende graf har forskjellige vekter, vil grafen kun ha ett minimalt spenntre.

- Sant ✓
- Usant

Med Kruskals algoritme vil spenntreet lages deterministisk dersom alle kantvektene er unike. Dette gir noe intuisjon.

Dette kan også bevises formelt med et liknende kontradiksjonsbevis. Vi antar det motsatte tilfellet, at det finnes flere minimale spenntrær. La T1 og T2 være to distinkte, minimale spenntrær. Vi kan da finne en kant e1 som er den kanten med lavest vekt som kun er med i enten T1 eller T2 (men ikke begge). Vi kan trygt anta at $e1 \in T1$. Dersom vi legger til e1 i T2, vil det oppstå en sykel. I denne sykelen vil det eksistere en kant e2 der $e2 \notin T1$ (i det motsatte tilfellet ville T1 inneholdt en sykel). Videre må vekten til e1 være lavere enn vekten til e2, siden e1 er definert til å være kanten med lavest vekt som kun er med i enten T1 eller T2. Dermed kan e2 substitueres med e1 i T2, og vi få et spenntre med lavere vekt. Dette er motstridende med at T2 var minimalt, og beviset fullføres ved kontradiksjon.

Question 13: Minimale spenntrær

Dersom ikke alle kantvektene i en sammenhengende graf er unike, vil denne grafen nødvendigvis ha flere minimale spenntrær.

- Sant
- Usant ✓

Vises enkelt ved moteksempel. Oppgave 6 er for øvrig ett slikt moteksempel.

Question 14: Grubleoppgave

Din venn Lurvik tror han har laget en god algoritme for å finne minimale spenntrær. Anta en sammenhengende og urettet graf. Algoritmen er som følger:

La S betegne mengden med foreløpig valgte noder, og A mengden med foreløpig valgte kanter.

```
1) S = ∅
2) A = ∅
3) for u ∈ V
4)     if u ∉ S
5)         Velg kanten (u, v) ut fra u med lavest vekt
6)         A = A ∪ {(u, v)}
7)         S = S ∪ {u, v}
```

Hvilket alternativ er riktig?

- Algoritmen finner alltid minimale spenntrær.
- Algoritmen finner aldri minimale spenntrær.
- Algoritmen finner spenntrær som ikke er minimale.
- Når algoritmen finner et spenntre, er det minimalt. ✓

I denne oppgaven kan man komme langt med å lage moteksempler. Da kan man enkelt se at algoritmen vil, for mange grafer, ikke gi en sammenhengende graf (den vil ikke finne et spenntre), og for andre grafer vil den finne et minimalt spenntre. For å bevise at algoritmen alltid finner et minimalt spenntre dersom den først finner et spenntre, kan man argumentere med teoremene som boka innfører. I øyeblikket en node tas med er noden enda ikke koblet til treet vårt. Da kan vi definere et snitt rundt denne noden. Blant kantene som krysser snittet (kantene ut fra noden) må en lett kant være en trygg kant. Vi kan derfor velge kanten med lavest vekt. I de tilfellene der algoritmen terminerer med en sammenhengende graf, er den grafen altså et minimalt spenntre.