NTNU
Institutt for datateknikk
og informasjonsvitenskap

Eksamensoppgave i TDT4120 Algoritmer og datastrukturer

Faglig kontakt under eksamen	Magnus Lie Hetland
Tlf.	91851949
Eksamensdato	7. desember 2013
Eksamenstid (fra–til)	0900–1300
Hjelpemiddelkode	D. Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.
Målform/språk	Bokmål
Antall sider	5
Antall sider vedlegg	0
Kontrollert av	Pål Sætrom

_____

Dato Sign

Les alle oppgavene før du begynner, disponer tiden og forbered spørsmål til faglærer ankommer lokalet. Gjør antagelser der det er nødvendig. Skriv kort og konsist på **angitt sted**. Lange forklaringer og utledninger som ikke direkte besvarer oppgaven tillegges liten eller ingen vekt.

Algoritmer kan beskrives med tekst, pseudokode eller programkode, etter eget ønske (med mindre annet er oppgitt), så lenge det klart fremgår hvordan den beskrevne algoritmen fungerer. Korte, abstrakte forklaringer kan være vel så gode som utførlig pseudokode, så lenge de er presise nok. Algoritmer som konstrueres bør generelt være så effektive som mulig, med mindre annet er opplyst. Kjøretider oppgis med asymptotisk notasjon, så presist som mulig. Alle deloppgavene teller like mye.

1. What is the minimum and maximum number of edges in a connected, undirected graph with n nodes?

Minimum: $n - 1$ (For et tre)
Maksimum: $n(n - 1)/2$ (For en komplett graf)

2. What is the minimum and maximum number of nodes in a rooted binary tree of depth n? (The depth is the maximum number of edges in any path from the root.)

Minimum: $n + 1$
Maksimum: $2^{n+1} - 1$

3. Hva er forventet (*average-case*) asymptotisk kjøretid for et mislykket søk i en hash-tabell med m plasser (*slots*) som inneholder n elementer? Uttrykk svaret som funksjon av n og m. (Du kan anta enkel, uniform hashing og at kollisjoner løses vha. *chaining*.)

Svar: $\Theta(1 + n/m)$

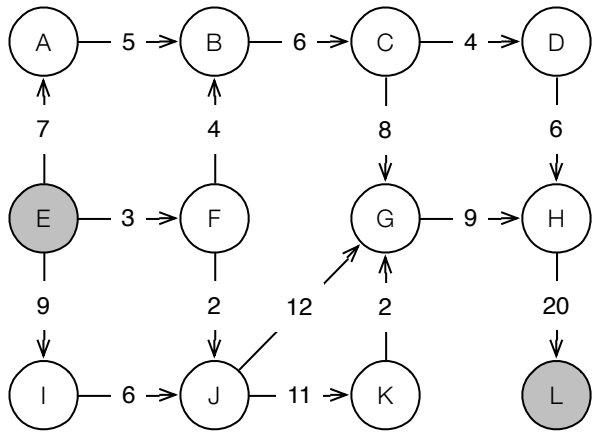
4. Et problem kan ofte brytes ned i delproblemer, der enkelte delproblemer er avhengige av andre delproblemer. For å løse problemet, må delproblemene løses i en rekkefølge som tilfredsstiller disse avhengighetene. Om man ser på delproblemene som noder og avhengighetene som kanter i en rettet graf, hva kalles en slik rekkefølge?

Svar: En topologisk sortering

5. Løs rekurrensen $T(n) = T(n+1) - 2^n$; $T(1) = 1$. Oppgi svaret eksakt.

Svar:
$T(n+1) = T(n) + 2^n$
$T(n) = T(n-1) + 2^{n-1}$
$T(n) = \text{sum}(2^{k-1} \text{ for } k=1\ldots n) = 2^n - 1$

6. Følgende rettede graf representerer et flytnettverk, med angitte kapasiteter. Hva er maksimal flyt fra node E til node L?



Svar: 13
Man f.eks. følgende stier:
EIJGHL gir 6 i flyt
EFJGHL gir 2 til, totalt 8
EABCDHL gir 4 til, totalt 12
EFBCGHL gir 1 til, totalt 13. Mer enn dette er umulig, siden vi har et snitt med kapasitet 13.

7. Hva er maks-haug-egenskapen (*the max-heap property*)?

Svar: At verdien i en node ikke er større enn verdien i foreldrenoden.

8. Sortering av n elementer kan generelt ikke utføres i $O(n)$ tid. Likevel klarer f.eks. tellesortering (*counting-sort*) dette. Hvordan kan det ha seg?

Svar: Den forutsetter at elementene er heltall i et begrenset verdiområde.

9. Your friend claims to have invented an algorithm that solves the traveling salesman problem in $O(n \log(\log(n)))$ time, where n is the number of vertices in the graph. If your friend's claim were correct, would there be any important consequences of his or her discovery? Explain briefly.

Svar: Han ville ha løst et NP-hardt problem i polynomisk tid, og dermed vist at P = NP.

10. You have n friends. You know that some of your friends hate each other. Hatred is always mutual. You know exactly which pairs of friends that hate each other. You wish to unfriend some of your friends, so that none of your remaining friends hate each other. Your task is to determine if you can solve this problem by unfriending k friends, where k is a an input parameter. You can assume that this decision problem is in NP. Show that it is NP-complete. Explain your reasoning clearly, and make sure you include all required parts of such an argument.

Svar: Reduksjon fra VERTEX-COVER. For et gitt VERTEX-COVER-problem, lag én venn per node og ett uvennskap per kant. Det vil da være mulig å unfriende k venner og bli kvitt alle vennskapene hvis og bare hvis det finnes et vertex cover av størrelse k i den opprinnelige grafen – de unfriendede vennene dekker sine uvennskap (dvs. kanter), og det er ingen uvennskap (kanter) igjen dersom problemet er løst. Siden VERTEX-COVER er kjent NP-komplett og vi har redusert fra det til vårt problem så er vårt problem NP-komplett.

11. Student Lurvik hevder at han har funnet en måte å få Dijkstras algoritme til å tåle negative kanter: Finn den korteste kanten, altså den «mest negative», og kall lengden dens for w. Adder $|w| + 1$ til alle kantlengdene, slik at alle kantene får positive lengder. Kjør deretter DIJKSTRA på den modifiserte grafen; ta svaret man får og bruk de tilsvarende kantene i den originale grafen. Bevis at dette alltid gir rett svar, eller finn et moteksempel som viser at Lurvik sin algoritme kan gi feil svar.

Svar: Antall kanter påvirker hvor mye lengden økes under transformen. Ethvert eksempel som viser dette gir full uttelling.

12. Dijkstras algoritme kan implementeres ved å bruke en binærhaug (*binary heap*) som prioritetskø, eller ved å bare bruke den usorterte nodetabellen som prioritetskø. Dette fører til forskjellige kjøretider. Gitt følgende pseudokode for DIJKSTRA, oppgi de linjene som blir påvirket av dette valget og forklar hvorfor kjøretidene blir hhv. $O(E \lg V)$ og $O(V^2)$. Når bør man bruke hva?

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

Linjer som påvirkes (linjenummer): 3, 4, 5, 8 (her er 5 og 8 viktigst)
Forklaring av kjøretider:
For hver node (linje 5) må man hente ut minimum – $O(V \lg V)$ vs $O(V^2)$.

For hver kant (linje 8) må man oppdatere estimat – $O(E \lg V)$ vs $O(E)$.
Når man bør usortert liste/tabell:
Når man har veldig mange kanter (flere enn $\Theta(V^2/\lg V)$).

13. Multiplikasjon av en matrise med n rader og m kolonner med en matrise som har m rader og k kolonner har kjøretiden $O(nmk)$. Algoritmen for å løse 0/1-knapsack-problemet hvor man skal stjele n gjenstander ved hjelp av en sekk som har kapasitet k kjører på $O(nk)$. Førstnevnte algoritme har polynomisk kjøretid, mens den andre algoritmen har eksponentiell kjøretid. Hvordan forklarer man dette?

Svar: For ryggsekkproblemet er k en eksponentiell funksjon av problemstørrelsen.

14. Din venn Smartnes mener han har lest i pensum at worst-case-kjøretiden til sammenligningsbasert sortering er Ω(n lg n), men han skjønner ikke helt hva det betyr. Forklar kort hva det betyr, eller forklar kort hvorfor dette ikke gir mening.

Svar: Det betyr at man ikke kan gi noen garantier som er bedre enn $\Omega(n \lg n)$ – altså en nedre grense – selv om man i beste tilfelle (*best-case*) kan få bedre kjøretider og i verste tilfelle selvfølgelig kan få verre.

15. To personer skal besøke en sekvens med byer. Du har oppgitt en ordnet sekvens av n byer, og avstandene mellom alle par med byer. Du skal fordele byene i to subsekvenser (der byene ligger i sin opprinnelige rekkefølge, men ikke nødvendigvis rett etter hverandre i den opprinnelige sekvensen) sånn at person A besøker byene i den første sekvensen (i rekkefølge) men ingen av de andre, person B besøker byene i den andre sekvensen (i rekkefølge) men ingen av de andre, og sånn at summen av avstandene A og B reiser totalt er minimert. (Hver by skal altså besøkes av nøyaktig én av A eller B.) Anta at A og B starter i de første byene i sine respektive subsekvenser. Beskriv en effektiv algoritme som løser problemet. Hva blir kjøretiden?

Svar: Konseptuelt: Lag en DAG der nodene representerer posisjonene til A og B (dvs., hver node er et par med to ulike byer), og kantene representerer at enten A eller B går videre til den neste ubesøkte byen (med tilhørende kostnad lik reiseavstanden). Finn korteste vei med DP (DAG-Shortest-Path). Antall noder er kvadratisk og antall kanter per node er konstant. Kjøretiden blir altså $\Theta(n^2)$.
Nodene trenger ikke representeres eksplisitt – man kan også ha to tabeller som angir beste kostnad gitt at hhv A og B besøker en gitt by. For å finne neste verdi for A må man søke seg bakover for alle mulige forgjenger-byer. Man må samtidig legge til kostnaden for at B har besøkt de mellomliggende byene. Minnebruken blir da lineær, men kjøretiden er fortsatt kvadratisk.