Avsluttende eksamen i
TDT4120 Algoritmer og datastrukturer

Eksamensdato	18. august 2010
Eksamenstid	0900–1300
Sensurdato	8. september
Språk/målform	Bokmål
Kontakt under eksamen	Magnus Lie Hetland (tlf. 91851949)
Tillatte hjelpemidler	Ingen trykte/håndskrevne; bestemt, enkel kalkulator

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, 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. Kjøretider oppgis med asymptotisk notasjon, så presist som mulig.

a) Hvis du kjører DFS i en urettet graf vil du ikke få noen *forward* eller *cross edges*. Hvorfor?

Svar (11%): Anta at du fant en. Den ville da allerede ha blitt undersøkt fra den andre siden.

b) Det kan være ønskelig å implementere QUICKSORT slik at den velger pivotobjektet tilfeldig (såkalt RANDOMIZED-QUICKSORT). Hva oppnår man med dette? Begrunn svaret kort.

Svar (11%): Unngå en konsekvent worst-case på gitt type input. Vil ikke forbedre worst-case eller average-case (med mindre du vet noe om input-fordelingen).

c) Anta at du implementerer Dijkstras algoritme, og at du bruker en usortert tabell eller liste som prioritetskø i stedet for en binær heap. Med andre ord can *decrease-key* nå gjøres i konstant tid, mens *extract-min* tar lineær tid. For en graf med m kanter og n noder, hva blir den totale kjøretiden for algoritmen? Oppgi svaret i Θ-notasjon. Begrunn svaret kort.

Svar (11%): $\Theta(n^2)$

d) Du kommer over et problem A som minner deg svært om Subset-Sum-problemet. Hva slags relasjon vil du prøve å etablere mellom A og Subset-Sum for å vise at A er NP-komplett?

Svar (11%): Reduser Subset-Sum til A i polynomisk tid.

e) Du har oppgitt en bipartitt vektet graf med n noder og m kanter. Du skal finne en bipartitt matching med maksimal vekt. Du vet at alle kantvektene er unike heltallspotenser av 2. Hvordan vil du løse problemet, og hva blir kjøretiden? Begrunn svaret kort.

(Nodene til en bipartitt graf kan deles i to mengder slik at ingen av mengdene har noen kanter internt. En bipartitt matching er et subsett av kantene mellom mengdene der ingen av kantene i subsettet har noen felles noder.)

Svar (11%): Summen av toerpotenser er alltid mindre enn neste toerpotens. Derfor vil det alltid lønne seg å ta med den største kanten som kan tas med, og vi kan løse problemet grådig. Sorter kantene synkende, og legg til lovlige kanter etter tur. Kjøretiden blir loglineær.

Du skal flette sammen $m = 2^k$ sorterte lister med lengder $L_1, \ldots L_m$ (der k er et positivt heltall) og vurderer to ulike algoritmer:

Algoritme 1 (Splitt og hersk): Hvis du har kun to lister, flett dem sammen (i lineær tid). Ellers: Gruppér de $m/2$ første filene i én mengde A og resten i en mengde B. Løs problemet for A og B rekursivt.

Algoritme 2 (Huffman): Gjenta det følgende til du står igjen med én liste: Velg ut de to minste listene og flett dem sammen (i lineær tid). Hvis de to listene hadde lengde L_p og L_q så har den resulterende listen lengde $L_p + L_q$, og den legges tilbake i mengden.

f) Gi et eksempel på et sett med listelengder som vil gi asymptotisk forskjellig kjøretid med Algoritme 1 og Algoritme 2. Begrunn svaret kort.

Svar (11%): La lengdene være en rekke med toerpotenser. Den siste ville være lineær, så balansert fletting vil gi loglineær kjøretid. Huffman vil gi lineær tid (jfr sum av toerpotenser).

g) Anta at du har n punkter i planet og du ønsker å finnet minimalt euklidsk spenntre over punktene. Det betyr at du skal finne et spenntre for den komplette grafen over punktene, der alle kanter har vekt lik avstanden mellom punktene. Gi en tett, nedre asymptotisk grense for kjøretiden. Forklar svaret kort.

Svar (11%): Problemet kan ikke løses raskere enn i loglineær tid, siden det kan brukes til sortering av reelle tall.

Det såkalte *ubegrensede* ryggsekkproblemet er nært beslektet med 0-1-ryggsekkproblemet. I stedet for at hvert objekt enten kan være med eller ikke, så kan det være med null eller flere ganger (men du kan fortsatt bare ha med hele objekter, i motsetning til *fractional knapsack*). Det er altså snakk om *objekt-typer* mer enn individuelle objekter.

h) Beskriv kort hvordan du ville løse det ubegrensede ryggsekkproblemet. Hva blir kjøretiden, som funksjon av n (antall objekt-typer) og W (ryggsekkapasitet)? Begrunn svaret kort.

Svar (11%): Som vanlig 0-1-ryggsekk, med DP, men trenger bare éndimensjonal tabell. For delløsning $m[n]$, ta maksimum av $m[n–1]$ og $m[n\text{-}w[i]] + v[i]$ for alle objekt-typer i.

Du skal invitere venner til fest. Du vurderer n venner, men du vet at hver av dem bare vil ha det hyggelig dersom han eller hun kjenner minst k andre på festen. (Du kan anta at dersom A kjenner B så kjenner B automatisk A.)

i) Beskriv en algoritme som en størst mulig delmengde av vennene dine der alle kjenner minst k av de andre, dersom en slik delmengde eksisterer. Forklar kort hvorfor algoritmen er korrekt.

Svar (12%): Induksjon/rekursjon. En person som kjenner færre enn k av de andre kan uansett ikke være med, og kan trygt fjernes. Hvis ingen slik person finnes er du ferdig.