Avsluttende eksamen i TDT4120 Algoritmer og datastrukturer

Eksamensdato	3. desember 2012
Eksamenstid	0900–1300
Sensurdato	3. januar 2013
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.

Obs: Se vedlegg bakerst for ekstra informasjon som kan brukes til å løse oppgavene.

a) Vis, ved å finne konstantene fra definisjonen, at $\sin(n)$ er $O(1)$.

Svar (6%): F.eks. $n_0 = 0$ og $c = 1$. (Her kan man også bruke ϵ i stedet for c, som i Kleinberg.) **Merk:** Cormens definisjon av O-notasjon krever at funksjonen som beskrives er ikke-negativ. Derfor aksepteres også svar av typen «$\sin(n)$ er ikke $O(1)$, siden den kan bli negativ».

b) En måte å unngå konsekvent worst-case-atferd på i hashing går ut på å velge hashfunksjonen tilfeldig. Hva kalles dette?

Svar (6%): Universell hashing. Her godkjennes også perfekt hashing.

c) En teknikk brukt i dynamisk programmering går ut på å lagre returverdien til en funksjon første gang den kalles med et gitt sett med parametre, og så bare returnere/bruke denne verdien direkte hvis funksjonen kalles med disse parametrene igjen. Hva kalles dette?

Svar (6%): Memoisering.

d) Tegn et eksempel på en urettet, sammenhengende graf med en markert startnode, der bredde-først-søk vil finne korteste veier til de andre nodene, mens dybde-først-søk garantert ikke vil det, samme hvilken rekkefølge naboene besøkes i. (Det vil si, dybde-først-søk må finne *minst én* gal sti.) Bruk så få noder som mulig.

Svar (6%): Komplett graf med tre noder. Flere noder (med en sykel i grafen) gir noe uttelling.

e) I læreboka (Cormen et al.) argumenteres det for at det er meningsløst å tillate negative sykler i korteste veier. Hvis vi begrenser oss til å kun lete etter korteste veier *uten sykler* (såkalte *simple paths*), hvorfor er det likevel problematisk om grafen vår inneholder negative sykler (som kan nås fra startnoden)?

Svar (6%): Problemet er da NP-hardt. (Forklaringer om at relax-baserte algoritmer vil gi feil svar, eller at vi ikke kjenner noen algoritmer som løser problemet vil kunne gi noe uttelling.)

f) I ryggsekkproblemet (0-1 *knapsack*), la $c[i, w]$ være optimal verdi for de i første objektene, med en kapasitet på w. La v_i og w_i være henholsvis verdien og vekten til objekt i. Fyll ut rekurrensen for $c[i, w]$, hvis vi antar at $i > 0$ og $w_i \leq w$.

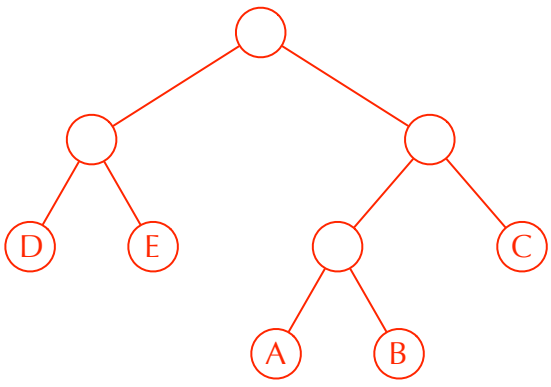
Svar (6%): $c[i, w] = \max\{v_i + c[i–1, w–w_i], c[i–1, w]\}$.
Merk: Her sto det $w_i \geq w$ i oppgaven. Det ble opplyst om at det var feil på eksamen. For $w_i \geq w$ bør egentlig også alternativet $c[i, w] = c[i–1, w]$ oppgis (for $w_i > w$), og det gis også full score om man har lagt til dette.

g) Edmonds-Karp er en implementasjon av Ford-Fulkerson som garanterer polynomisk kjøretid. Hva gjøres i Edmonds-Karp som gir denne garantien?

Svar (6%): Den bruker de korteste flytforøkende stiene. (Finnes med BFS.)
Her er det også OK å bare si at den bruker BFS.

h) Konstruer et Huffman-tre for tegnene A, B, C, D og E med frekvenser på henholdsvis 10, 11, 23, 12 og 13. La alltid det minste av to deltrær være til venstre. Tegn treet nedenfor.

Svar (6%): I oppgaven er ordet «minste» tvetydig, og kan tolkes som «med minst vekt» og «med færrest noder». Derfor godkjennes også alle rekkefølger av nodene, dvs. svar som har samme struktur som det følgende treet:



Merkelappene 0/1 på kantene og vekter i nodene kreves ikke, men trekker ikke ned.

En venn av deg har et problem som består i å avgjøre om en graf G inneholder en sykel av lengde k (målt i antall kanter), for et gitt heltall k. Vennen din vil vise at problemet er NP-komplett, og vil gjøre det ved å la grafen H være en sykel av lengde k, og så løse sykelproblemet ved hjelp av subgrafisomorfismeproblemet.

i) Hva er galt med tankegangen til vennen din?

Svar (6%): Han/hun reduserer feil vei.

j) Du vil sortere nodene i en rettet, asyklisk graf slik at alle kantene peker «fra venstre til høyre» (altså til noder senere i rekkefølgen). Hvis grafen har n noder og m kanter, hva blir kjøretiden for denne sorteringen?

Svar (6%): $\Theta(m + n)$

Betrakt følgende pseudokode. Anta at A er en tabell med lengde $n \geq 1$, der n er en heltallspotens av 2. Anta at PARSUM til å begynne med kalles med $i = 0$ og $j = n - 1$. Funksjonen floor(x) returnerer det største heltallet y slik at $y \leq x$.

```
PARSUM(A, i, j):
if i < j
    k = floor((i + j) / 2)
    return PARSUM(A, i, k) + PARSUM(A, k + 1, j)
else
    return A[i]
```

k) Hva er parallellitetsgraden (*parallelism*) til algoritmen PARSUM, som funksjon av n? Oppgi svaret med asymptotisk notasjon.

Svar (7%): Her var intensjonen at de to kallene skulle være parallelle, men siden **spawn** ikke er oppgitt, godkjennes både $\Theta(n/\lg n)$ og $\Theta(1)$.

l) Løs rekurrensen $T(n) = T(2n)/2 - n$, $T(1) = 1$. Oppgi svaret med asymptotisk notasjon.

Svar (7%): $\Theta(n \lg n)$
Merk: I oppgaven sto det + i stedet for –. Det ble oppgitt på eksamen at dette var feil. Likevel tas oppgaven ut av sensur der det er til fordel for studenten.

Du skal finne korteste/billigste veier i en rettet graf der vektene ligger i *nodene* heller enn i kantene. Det vil si, hver node v har en vekt $w(v)$. Du representerer dette som et vanlig korteste-vei-problem ved å gi alle kantene inn til en gitt node v en vekt på $w(v)$ og ignorerer vekten i noden. Du bruker nå Dijkstras algoritme for å finne korteste vei fra en gitt startnode til alle andre noder. Grafen har n noder og m kanter.

m) I verste tilfelle, hvor mange ganger vil du måtte gjøre endringer i tabellen med avstandsestimater, d? (Det er altså snakk om oppdateringer i estimatet *etter* initialiseringen.) Oppgi svaret med asymptotisk notasjon. Begrunn svaret kort.

Svar (8%): $\Theta(n)$. Kun den første kanten som oppdateres (med RELAX) inn mot en node v kan medføre en forandring, siden de andre forgjengernodene må ha høyere avstand, men samme kantvekt.

Betrakt følgende pseudokode.

```
GLYMPHID(anx, gerb):
for bith in 701, 301, 132, 57, 23, 10, 4, 1
    for chiss = bith to gerb–1
        ewok = anx[chiss]
        while chiss ≥ bith and anx[chiss – bith] < ewok
            anx[chiss] = anx[chiss – bith]
            chiss = chiss – bith
        anx[chiss] = ewok
```

n) Hvilket problem løser algoritmen GLYMPHID?

Svar (8%): Den sorterer de *gerb* første elementene i tabellen *anx* i synkende rekkefølge. (GLYMPHID er en variant av en sorteringsalgoritme ved navn SHELLSORT. Merk at den i siste iterasjon, der bith = 1, oppfører seg som en invertert INSERTION-SORT, uavhengig av de foregående iterasjonene.) Det gir også uttelling om man ikke påpeker rollen til *gerb*. Man vil også få noe uttelling om man kun har sagt at algoritmen sorterer *anx*.

Anta at du får oppgitt to heltallstabeller $A = [a_0, a_1, \ldots, a_{n-1}]$ og $B = [b_0, b_1, \ldots, b_{m-1}]$. Du vil avgjøre om tabellen A kan deles opp (partisjoneres) i sammenhengende, ikke-tomme, ikke-overlappende segmenter slik at hvert segment summerer til ett av tallene i B. Alle tallene i A skal være del av nøyaktig ett segment. Flere segmenter kan summere til samme tall i B og ikke alle tallene i B trenger å brukes.

Eksempel: A = [2, 4, 5, 1, 2, 2, 9, –5], B = [6, 8].
Svaret er her ja. (Mulig partisjon: sum(2, 4) = 6, sum(5, 1, 2) = 8, sum(2, 9, –5) = 6.)

Merk: Du trenger ikke finne selve partisjonen. Du trenger bare avgjøre hvorvidt en slik partisjon er mulig eller ikke.

o) Beskriv en algoritme som løser problemet så effektivt som mulig. Hva blir kjøretiden i verste tilfelle? (Oppgi svaret med asymptotisk notasjon.)

Svar (10%): Løses vha. dynamisk programmering, der vi har en boolsk tabell T, slik at $T[k]$ er sann hvis og bare hvis det finnes en partisjon for de k første elementene.

```
bygg en perfekt hashtabell H for B
T[0] = true
for k = 1 to n
    T[k] = false              # Løsbart for de a_0 ... a_{k-1}?
    sum = 0                   # Summen av a_{i-1} ... a_{k-1}
    for i = k downto 1
        sum = sum + a_{i-1}
        T[k] = T[k] or ((sum in H) and T[i - 1])
        if T[k]: break
return T[n]
```

Kjøretid: $\Theta(m + n^2)$

Vedlegg

Hvis alle nøklene er kjent i det man bygger en hashtabell er det mulig å forbedre *worst-case*-kjøretiden for oppslag til å bli det samme som vanlig *average-case*-kjøretid. Dette kalles *perfekt hashing*. Anta at en slik perfekt (kollisjonsfri) hashtabell over n elementer kan bygges i $O(n)$ tid.

Perfekt hashing er her beskrevet fordi det ikke er pensum, men er nyttig i oppgave o.

To grafer er isomorfe hvis de har samme struktur – dvs. de er like om man ser bort fra merkelappene på nodene. Subgrafisomorfismeproblemet består i å avgjøre om en gitt graf H er isomorf med en eller annen subgraf i en annen gitt graf G. Dette problemet er NP-komplett.