

Løsningsforslag, Eksamen i IT1105 Algoritmer og datastrukturer

1. juni 2004 0900-1300

Tillatte hjelpemidler: Godkjent kalkulator og matematisk formelsamling
Skriv svarene på oppgavearket. Skriv studentnummer på alle ark.

Oppgave 1 (25%)

- 5% a) I en urettet, uvektet graf, bør man bruke dybde-først-søk eller bredde-først-søk for å finne den korteste stien mellom to noder?

Svar: Bredde-først-søk.

- 5% b) Hvilken designmetode kjennetegner algoritmene MergeSort og QuickSort?

Svar: Splitt og hersk (divide and conquer).

- 5% c) Hva brukes Floyds algoritme til?

Svar: Å finne korteste vei fra alle noder til alle andre i en rettet graf.

- 5% d) Hvilke krav stilles til grafen om vi skal kunne bruke Dijkstras algoritme?

Svar: At ingen kant-vekter er negative.

- 5% e) Hvilken egenskap ved problemstrukturen gjør at dynamisk programmering egner seg som løsningsmetode?

Svar: Overlappende delproblemer.

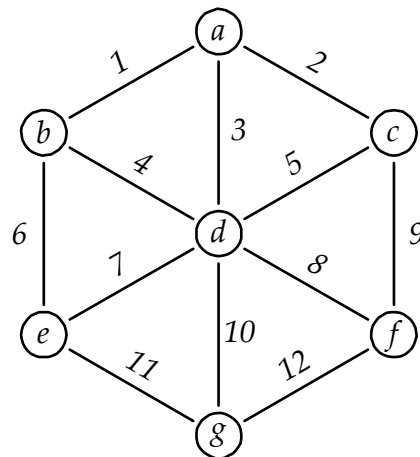
Oppgave 2 (10%)

- 5% a) Du skal utføre binærsøk etter nøkkelen $K = 7$ i følgende tabell, A :

[2, 5, 9, 14, 24, 25, 26, 27, 29, 34, 39, 40, 45, 46, 47]

Tabellen er 0-indeksert, slik at $A[0] = 2$, $A[1] = 5$ og så videre. For hvert trinn i algoritmen sammenlignes K med ett element i A . Oppgi indeksene til disse elementene i rekkefølge. (Hvis du, for eksempel, begynte med å sammenligne K med det første tallet, 2, så skulle det første tallet i svaret ditt være 0.) Oppgi kun de relevante indeksene. Ingen begrunnelse er nødvendig.

Svar: 7, 3, 1, 2



Figur 1: En vektet graf

- 5% b) Figur 1 viser en vektet graf med 7 noder. Utfør Kruskals algoritme og oppgi alle kantene som er med i det resulterende spenntreet. Skriv løsningen på formen $a-b$, $c-d$, ... der $a-b$ er kanten mellom a og b , og så videre.

Svar: $a-b$, $a-c$, $a-d$, $b-e$, $d-f$, $d-g$

Oppgave 3 (35%)

- 5% a) Hva er forskjellen på O , Θ og Ω ?

Svar: O gir en asymptotisk øvre grense, Ω gir en asymptotisk nedre grense og Θ gir både en øvre og en nedre grense.

- 5% b) Er $2^n \in \Omega(2^{n+1})$? Gi en kort begrunnelse.

Svar: Ja, fordi $2^{n+1} = 2 \cdot 2^n \in \Omega(2^n)$. Med andre ord er det bare en konstant faktor som skiller de to uttrykkene.

- 5% c) Er $3^n \in O(2^n)$? Gi en kort begrunnelse.

Svar: Nei, fordi her er det mer enn en konstant faktor som skiller uttrykkene.

- 5% d) En algoritmedesigner prøver å lage en rekursiv algoritme for å stokke kort. Hun deler kortstokken i to like store deler, lar fem tilfeldige kort fra hver halvdel bytte plass, og stokker så hver halvdel rekursivt. For enkelhets skyld, anta at kortstokken oppfører seg som en tabell med tall og at antallet kort er en toerpotens. Hva blir kjøretiden til algoritmen? Sett opp en rekurrens og skriv løsningen med asymptotisk notasjon.

Svar: Rekurrens: $T(n) = 2T(n/2) + \Theta(1)$. Løsning: $T(n) = \Theta(n)$.

- 5% e) Algoritmedesigneren innser at metoden ikke var god nok og beslutter seg for å gå fra å bytte om på fem kort fra hver halvdel (i hvert rekursive kall) til å bytte om på fem *prosent* av kortene. Hva blir kjøretiden til algoritmen nå? Sett opp en rekurrens og skriv løsningen med asymptotisk notasjon.

Svar: Rekurrens: $T(n) = 2T(n/2) + \Theta(n)$. Løsning: $T(n) = \Theta(n \cdot \log n)$.

- 5% f) Etter en stund blir algoritmedesigneren lei av å stokke kort. I stedet for å stokke begge halvdelene rekursivt stikker hun nå bare *en* tilfeldig valgt halvdel rekursivt. Hva blir kjøretiden til algoritmen nå? Sett opp en rekurrens og skriv løsningen med asymptotisk notasjon.

Svar: Rekurrens: $T(n) = T(n/2) + \Theta(n)$. Løsning: $T(n) = \Theta(n)$.

- 5% g) Utpå kvelden begynner algoritmedesigneren å bli skikkelig lei. Hun bestemmer seg for å gå tilbake til å kun bytte om på fem tilfeldige kort, men fortsetter å kun stokke én av halvdelene rekursivt. Hva blir kjøretiden nå? Sett opp en rekurrens og skriv løsningen med asymptotisk notasjon.

Svar: Rekurrens: $T(n) = T(n/2) + \Theta(1)$. Løsning: $T(n) = \Theta(\log n)$.

Oppgave 4 (30%)

- 5% a) Du har en sortert sekvens med unike ID-nummer (positive heltall) $a_1 \dots a_n$. Du ønsker å finne det laveste positive heltall som ennå ikke er brukt som ID-nummer. Hvordan kan du avgjøre om det finnes en "ledig plass" (et tall som ikke er brukt) i delsekvensen $a_i \dots a_j$ i konstant tid? (For eksempel vil det i sekvensen (3, 4, 6, 7, 9) være to ledige tall, nemlig 5 og 8.)
Hint: Det er viktig at det er snakk om unike tall, og at de er heltall, ikke reelle tall.

Svar: Sjekk om $a_j - a_i > j - i$. I så fall vil det være "ledig plass."

- 5% b) Hvordan kan du bruke sjekken fra forrige deloppgave til å finne det laveste tallet som ikke er i bruk? Hva blir den totale kjøretiden?
Hint: Hvilken algoritme fra pensum kan tillempes til dette problemet?

Svar: Tilpasser binærsøk. Sjekk begge halvdelene av sekvensen som over. Hvis det er ledig plass i den venstre, undersøk den rekursivt. Hvis ikke, undersøk den høyre rekursivt. Kjøretiden blir logaritmisk.

- 5% c) Du har et ubegrenset lager med n forskjellige typer pappesker og ønsker å lage et høyt tårn av dem. Hver pappeske har en vekt og en kapasitet, begge målt i gram (heltall). Vekten av et tårn er altså summen av vekten til alle eskene i tårnet, og hver eske kan kun bære en vekt (summen av eskene

over) tilsvarende sin kapasitet. Anta at hver eske veier minst ett gram og har en kapasitet på minst ett gram. Skisser en *rå makt*-løsning på problemet. Hva blir kjøretiden, uttrykt ved n og den største kapasiteten, C ? (Her legges det ikke vekt på at algoritmen skal være effektiv.)

Svar: Hvis den høyeste kapasiteten er C , kan det ikke være mer enn C esker i tårnet. For hver mulig høyde $1..C$ prøver vi alle mulige tårn (alle mulige esker for hver posisjon i tårnet). Kjøretiden blir $O(\sum_{i=1}^C n^i) = O(Cn^C)$.

Her er det rom for mange lignende løsninger med andre kjøretider.

- 10% d) Anta at du representerer hver esketype med et heltall i , vekten til typen med $w[i]$ og kapasiteten til typen med $c[i]$. Sett opp en rekursiv funksjon, enten som en matematisk formel eller med pseudokode, for høyden $h(x)$ (i antall esker) til det høyeste tårnet som kan bygges hvis den totale vekten maksimalt kan være x .

Svar:
$$h(x) = \max_i (h(\min\{c[i], x - w[i]\}) + 1), \text{ der } i \text{ er slik at } w[i] \leq x.$$

Forklaring (ikke påkrevd av studentene): Vi prøver ut alle mulige kasser som kan plasseres i bunnen og finner ut hvor mye kapasitet vi har igjen. Den resterende kapasiteten blir $\min\{c[i], x - w[i]\}$. Vi bruker denne kapasiteten til å løse problemet rekursivt, og legger til 1. Dette gjøres for alle n kassetyper, og vi velger den som gir best svar.

- 5% e) Skisser en algoritme som finner høyden til det høyeste tårnet du kan bygge. Skriv algoritmen med pseudokode. Hva blir kjøretiden, uttrykt som funksjon av antall esker, n , og den høyeste kapasiteten, C ?

Hint: Det er en bestemt algoritmisk designmetode som egner seg til å løse rekursive problemer som beskrevet i forrige deloppgave.

Svar: Bruk dynamisk programmering til å beregne $h(x)$. Fyll en tabell $H[1..C]$ der C er den høyeste kapasiteten tilgjengelig. Hvert element krever en maksimalisering over de n mulige eske-typene, så kjøretiden blir $O(Cn)$.

```
Tårn( $w, c, n$ ):  
H[0] = 0  
C = max{  $c[1] \dots c[n]$  }  
for  $x = 1 \dots C$   
    best = 0  
    for  $i = 1 \dots n$   
        if  $w[i] \leq x$   
            rest = min{  $c[i], x - w[i]$  }  
            best = max{ best, H[rest] + 1 }  
    H[x] = best  
return H[C] + 1
```