

**Midtsemestertest**  
**TDT4120 Algoritmer og Datastrukturer**  
**Onsdag 1. oktober 2003**

**LØSNINGSFORSLAG**

**1. Algiator**

- a) Når  $n$  går mot uendelig er kompleksiteten både øvre og nedre avgrenset av en konstant ganget med  $n$ .
- b)  $\Theta(n^2)$
- c) At rekkefølgen i inndataene bevares for like elementer.
- d) Nei. Det gjelder bare dersom alle løvnoder er i samme dybde.

**2. Spennetre**

- a) (A,D), (A,B), (B,C), (C,H), (H,G), (G,E), (G,F)
- b) De samme som i a),  $\log(n)$  blir her  $\log(8)$  fordi det er 8 noder i denne grafen.  $\log(8)$  er 3 og ved å gange alle kanter med en positiv konstant vil vi få det samme minimale spennetreet som tidligere.

**3. Towers of Hanoi**

- a)  $A = \text{flytt}(\text{antall}-1, \text{fra}, \text{via}, \text{til});$   
 $B = \text{flytt}(\text{antall}-1, \text{via}, \text{til}, \text{fra});$
- b)  $T(n) = 2 * T(n-1) + \Theta(1)$   
 $T(1) = 1$
- c) Her vil vi gi tre ulike løsningsmetoder. Den første står ikke i den nye utgaven av boken, men vi mener at dette er en veldig god metode å bruke så vi tar den med. Denne metoden kalles iterasjonsmetoden. Metode 2 er å bruke rekurrenstrær. Denne er kanskje ikke spesielt formell og det er nok larest å bruke den for å finne en løsning å gjette på til alternativ 3 dersom man vil sikre seg full score. Alternativ 3 er slik boken ville løst denne oppgaven. Vi har her gjort denne løsningen veldig nøye slik at det skal være enklest mulig for dere å lese den og skjønne hva som gjøres i hvert ledd.

*Alternativ 1*

Bruker 1 istedenfor  $\Theta(1)$  her, for enkelhets skyld:

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2 * T(n-1) + 1 \\ &= 2 * (2 * T(n-2) + 1) + 1 \\ &= 2 * (2 * (2 * T(n-3) + 1) + 1) + 1 \\ &\dots \\ &= 2^i * T(n-i) + (1 + 2 + 4 + \dots + 2^{i-1}) \end{aligned}$$

Ligningen er gyldig for alle verdier av  $i$ . Vi velger  $i=n-1$ :

$$\begin{aligned} &= 2^n * T(n-n+1) + (1 + 2 + 4 + \dots + 2^{n-2}) \\ &= 1 + 2 + \dots + 2^{n-1} \end{aligned}$$

Her kan vi bruke formelen for sum av en geometrisk rekke (finnes i læreboka):

$$\begin{aligned} &= (1-2^{n+1}) / (1-2) \\ &= 2^{n+1} - 1 \\ &= \Theta(2^{n+1}) \end{aligned}$$

Eventuelt kan man bare si at  $1 + 2 + \dots + 2^{n-1}$  er  $\Theta(2^{n-1}) = \Theta(2^n)$ , fordi en geometrisk rekke domineres av sitt siste ledd (asymptotisk).

#### Alternativ 2

Tegn rekurrenstre for rekurrensen. Hver node vil ha en kostnad på  $\Theta(1)$ , og hver node vil ha to barn. Høyden på treet blir  $n$ . Den totale kjøretiden blir da  $O(\text{antall noder})$ , og antall noder i et perfekt balansert binærtre med høyde  $n$  er  $O(2^n)$ .

#### Alternativ 3

Denne rekurrensen er ikke på formen  $T(n) = a * T(n/b) + f(n)$ , dermed kan vi ikke bruke masterteoremet. Vi må dermed gjette en løsning for så å prøve å bevise denne med substitusjonsmetoden. Gjettingen kan vi komme frem til enten ved å tegne oppet rekurrenstre, ved å sette inn for  $T(n-1)$  et par ganger i rekurrensen eller vi kan tippe ut fra erfaring. I dette tilfelle vil det være naturlig å tippe på at løsningen er at  $T(n) = \Theta(2^n)$  ut ifra et rekurrenstre. For å vise at noe er  $\Theta(2^n)$  må vi vise at det er  $\Omega(2^n)$  og at det er  $O(2^n)$ . Vi prøver først å vise at det er  $O$ .

Tipper altså først at løsningen er  $T(n) = O(2^n)$

Dette fører til at  $T(n) \leq c * 2^n$

Dermed skal jeg anta at løsningen stemmer for  $T(n-1) \leq c * 2^{n-1}$  og deretter sjekke om dette medfører at det stemmer at  $T(n) \leq c * 2^n$

Setter så inn i rekurrensen at  $T(n-1) \leq c * 2^{n-1}$ :

$$T(n) = 2 * T(n-1) + \Theta(1)$$

$$T(n) \leq 2 * c * 2^{n-1} + \Theta(1)$$

$$T(n) \leq c * 2^n + \Theta(1)$$

Vi må så sjekke om det da stemmer at dette fører til at  $T(n) \leq c * 2^n$ .

Vi sammenligner derfor:

$$c * 2^n + \Theta(1) \leq c * 2^n$$

Her ser vi at dette ikke stemmer fordi det fører til at:

$$\Theta(1) \leq 0$$

Dette fører altså ikke frem, men vi må her ikke la oss lure til at løsningen vi har gjettet er feil. Det er gitt et eksempel på hvordan vi skal tenke her i boken på side 65. Vi har bare kommet til en feil i et konstantledd og gjetter derfor at løsningen er på formen:  $T(n) \leq c * 2^n - b * \Theta(1)$ . For enkelhets skyld setter vi her, i likhet med i alternativ 1, at  $\Theta(1) = 1$

Så prøver vi igjen:

Skal igjen prøve å vise at om  $T(n) \leq c * 2^{n-1} - b$  så fører det til at

$$T(n) \leq c * 2^n - b$$

Setter så inn:

$$T(n) = 2 * T(n-1) + \Theta(1)$$

$$T(n) \leq 2 * (c * 2^{n-1} - b) + 1$$

$$T(n) \leq c * 2^n - 2 * b + 1$$

Vi må så sjekke om dette faktisk fører til at  $T(n) \leq c * 2^n - b$

$$c * 2^n - 2 * b + 1 \leq c * 2^n - b$$

Flytter over og ordner opp i uttrykket og får at:

$$- 2 * b + 1 \leq b$$

$$b \geq 1$$

Vi har altså funnet ut at antagelsen vår stemmer for  $b \geq 1$ .

Vi kan nå vise dette på tilsvarende måte for  $\Omega$ -notasjon:

Nå tipper vi altså at løsningen er at  $T(n) = \Omega(2^n)$

Da håper vi å kunne bevise at:  $T(n) \geq c * 2^n - b$

Vi antar så at dette stemmer for  $n-1$ :  $T(n-1) \geq c * 2^{n-1} - b$

Så skal vi sette inn dette i rekurensen og sjekke om det medfører at det stemmer for

$T(n)$ .

$$T(n) = 2 * T(n-1) + \Theta(1)$$

$$T(n) \geq 2 * c * 2^{n-1} - 2 * b + 1$$

$$T(n) \geq c * 2^n - 2 * b + 1$$

Nå skal vi sammenligne dette med det vi håper å bevise, nemlig at

$$T(n) \geq c * 2^n - b$$

Da har vi at:

$$c * 2^n - 2 * b + 1 \geq c * 2^n - b$$

Vi rydder her opp og får:  $b \leq 1$

Utifra dette kan vi slutte at begge tilfeller stemmer når  $b = 1$ . Og vi får at løsningen av rekurensen er:  $T(n) = c * 2^n - 1$

Vi må så sjekke for boundary conditions.

Vi har at  $T(1) = 1$ . Vi setter inn i uttrykket og sjekker om dette stemmer for både  $O$ - og  $\Omega$ -notasjon:

Først  $O$ :

$$T(1) \leq c * 2^1 - 1$$

$$T(1) \leq c * 2 - 1$$

Sjekker så dette mot antagelse som at  $T(1) \leq 1$

$$2 * c - 1 \leq 1$$

$$c \leq 1$$

Vi ser altså at det stemmer at  $T(n) = O(2^n)$

Deretter sjekker vi for  $\Omega$ :

$$T(1) \geq c * 2^1 - 1 = 2 * c - 1$$

Sjekker dette mot antagelse som at  $T(1) \geq 1$

$$2 * c - 1 \geq 1$$

Dermed stemmer dette for  $c \geq 1$ .

Vi kan altså konkludere med at løsningen  $T(n) = \Omega(2^n)$

Siden vi vet at  $T(n)$  er både  $O$  og  $\Omega$  av  $2^n$  vet vi at  $T(n) = \Theta(2^n)$

#### 4. Ulykkesnummeret

$n^2$  - løsnings:

- a) For hvert tall, summer x neste tall til du har ulykkestallet som sum, hvis du overstiger ulykkestallet avbryter du og går til neste tall.
- b) java:

```
public static boolean harUlykkestall(int[] rekke, int ulykke){
    for( int i = 0; i< rekke.length; i++){
        int sum = 0;
        for(int j = i; j< rekke.length; j++){
            sum += rekke[j];
            if(sum == ulykke){
                return true;
            }else if(sum > ulykke){
                break;
            }
        }
    }
    return false;
}
```

- c) Kjøretid:  $\Theta(n^2)$ . Ser dette av dobbel for-løkke med konstant tid inni.
- d) NEI, finnes  $\Theta(n)$  -løsning.

$n$ -løsning:

- a) Som aksjespekulanten, men med visse modifikasjoner. Ha to pekere, en til start av subsekvensen og en til slutt av subsekvens. begynn med begge på 0, for hvert element, flytt sluttpeker en frem og legg til tallet den peker på., sjekk om tallet er funnet, hvis du er over ulykkestallet, flytt startpeker en frem og trekk fra tallet den pekte på før. Når du ikke kan flytte sluttpeker mer frem er du ferdig.
- b) java:

```
public static boolean harUlykkestall(int[] rekke, int ulykke){
    int start = 0;
    int sum = 0;
    for(int slutt = 0; slutt<rekke.length; slutt++){
        sum += rekke[slutt];
        while(sum>ulykke){
            sum -= rekke[start];
            start++;
        }
        if(sum==ulykke){return true;}
    }
    return false;
}
```

- c) Kjøretid:  $\Theta(n)$ , ser dette fordi det er en enkel for-løkke, og while-løkken inni er garantert til aldri å kjøre mer enn n ganger totalt. Går ut fra at ulykke er større enn null.
- d) JA, man må undersøke alle N elementene for å kunne uttale seg om subsekvensene.

## 5. Masterteoremet

- a) Vi sjekker hva  $n^{(\log_b(a))}$  er. Dette blir  $n^4$ . Dette blir dermed tilfelle 1 fordi vi ser at  $n^3 * \log(n) = O(n^{4 - \epsilon})$ . Dermed kan vi konkludere med at  $T(n) = \Theta(n^4)$
- b) Vi sjekker hva  $n^{(\log_b(a))}$  er. Dette blir  $n^2$ . Dette blir dermed tilfelle 2 fordi vi ser at  $n^2 = \Theta(n^2)$ . Dermed kan vi konkludere med at  $T(n) = \Theta(n^2 * \log(n))$

## 6. Bonusoppgave

- a) Sett  $D = \text{dist}(P1, Pn)$  (Euklidsk avstand).

Finn den minste "største-kanten"  $D$  som gir en løsning vha. binærseek:

1. Sett  $i = 0, j = D$ .
2. Finn korteste vei (med Dijkstra) når bare kanter med lengde mindre eller lik  $(i+j)/2$  tas med.
3. Fantes en slik vei?  
Ja: Sett  $j = (i+j)/2$   
Nei: Sett  $i = (i+j)/2$
4. Hvis  $|i-j| < \epsilon$  (hvis vektene er reelle) eller hvis  $i=j$  (hvis vektene er heltall), sett  $D = i$ .  
Hvis ikke, gå til 1.

*(Dijkstras algoritme i et minimalt spennetre vil også gi beste vei – men det bør begrunnes godt hvorfor dette er riktig. Man kunne da like godt ha brukt DFS eller BFS.)*

- b) Lager ny versjon av Relax:

Relax-Max( $u, v, w$ ):

1. **if**  $d[v] > \max\{d[u], w(u, v)\}$
2. **then**  $d[v] \leftarrow \max\{d[u], w(u, v)\}$
3.  $p[v] \leftarrow u$

(Altså – kun 2 linjer må endres)