

Eksamenshefte

TDT4120

Algoritmer og datastrukturer

Eirik Benum Reksten

1 SIF8010 august 2003 - Oppgave 1

I de følgende tre deloppgavene (1 a, b og c) skal du bruke den vektete, rettede grafen $G = (V, E)$, med $V = 1..6$. Kant-vektene defineres av matrisen C , slik at $C[i, j]$ er vekten til kanten fra i til j .

Vektmatrisen C er implementert ved hjelp av nabolister som er lagret i den endimensjonale tabellen W , slik at $W[i]$ er nabolisten til node i . Nabolistene inneholder par av typen (w, j) der w er vekten på kanten til den aktuelle nabonoden j . Kanter som ikke eksplisitt er oppgitt antas å ha uendelig stor vekt.

I det følgende, anta at første element i alle tabeller har indeks 1. Tabellen W er definert slik:

$$\begin{aligned}W[1] &= [(4, 2), (6, 3), (2, 4)] \\W[2] &= [] \\W[3] &= [(1, 5), (1, 6)] \\W[4] &= [(2, 5)] \\W[5] &= [(4, 6)] \\W[6] &= [(3, 2)]\end{aligned}$$

Ut fra denne definisjonen ser vi for eksempel at kanten fra node 1 til node 3 har vekt 6 ($C[1, 3] = 6$) og at kanten fra node 2 til node 4 har vekt ∞ ($C[2, 4] = \infty$). Alle algoritmene i denne oppgaven skal følge prioritetsregelen gitt nedenfor:

Prioritetsregel: Der en algoritme kan velge mellom flere noder, anta at den alltid velger den av de mulige nodene som har lavest nummer. Der en algoritme kan velge mellom flere kanter, anta at den alltid velger den av de mulige kantene som kommer tidligst i en leksikalsk sortering.

Dette betyr at node 3 velges før node 6, at kanten fra 2 til 3 velges før kanten fra 2 til 4, men etter kanten fra 1 til 5.

- a) Finn en topologisk sortering av grafen. Bruk dybde-først-søk til dette, som vist i læreboka, og følg prioritetsregelen. Bruk denne rekkefølgen til å finne korteste vei fra node 1 til node 2 med dynamisk programmering (DAG-SHORTEST-PATH). Bruk tabellen nedenfor til å vise hvordan hvert trinn i algoritmen oppdaterer $d[i]$ (avstandsestimatet til node i):

	$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$
Start	0	∞	∞	∞	∞	∞
Trinn 1						
Trinn 2						
Trinn 3						
Trinn 4						
Trinn 5						

- b) Utfør Prims algoritme på G 's underliggende urettede graf, det vil si grafen som er lik G bortsett fra at kantene ikke har retning (kantenes opprinnelige retning kan fremdeles ha betydning for prioritetsregelen). Oppgi kantene i den rekkefølgen de legges til i spenntreet på. Kantene beskrives med fra- og til-node som oppgitt i tabellen W . Start i node 1. Svar i tabellen nedenfor.

	Fra-node	Til-node
Trinn 1		
Trinn 2		
Trinn 3		
Trinn 4		
Trinn 5		

- c) Utfør Dijkstras algoritme på G for å finne korteste vei fra node 1 til alle andre noder. (Her har selvfølgelig kantenes retning stor betydning.) Vær nøye med å følge prioritetsregelen. Svar i tabellen nedenfor, på tilsvarende måte som i oppgave a. *Start* er tilstanden etter at initialiseringen har blitt utført, og *Trinn X* er tilstanden etter at X noder har blitt fargelagt grå og fått RELAX kjørt på sine naboer.

	$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$
Start						
Trinn 1						
Trinn 2						
Trinn 3						
Trinn 4						
Trinn 5						
Trinn 6						

- d) Anta at du skal implementere Kruskals algoritme for grafer der kant-vektene er heltall i et fast tallområde $[0, k]$ for en liten, konstant verdi k . Du bestemmer deg for å bruke tellesortering på kantene først, slik at de blir tilgjengelige i riktig rekkefølge. Bortsett fra dette implementerer du algoritmen som normalt (som i læreboka). Hva blir kjøretiden? Uttrykk kjøretiden med Θ -notasjon, der m er antall kanter og n er antall noder.
- e) Anta rent hypotetisk at å flette (MERGE) to sorterte tabeller kunne gjøres i konstant tid. Hva ville da kjøretiden til flettesortering (MERGE-SORT) bli? Bruk Θ -notasjon.

2 SIF8010 august 2003 - Oppgave 2

Anta at du har et grafikk-bibliotek tilgjengelig som lar deg tegne linjer i rutenett av typen vist i figur 1 og 2 på side 6. Kall til dette biblioteket gjøres med følgende pseudokode:

```
draw a line from (x1, y1) to (x2, y2);
```

Dette utsagnet tegner en rett linje fra punktet med koordinater (x_1, y_1) til punktet med koordinater (x_2, y_2) . De følgende deloppgavene dreier seg om funksjonene `func1` og `func2`, beskrevet med pseudokode nedenfor (abs er absoluttverdi-funksjonen):

```
func1(double a, b, c, d, e, f) {
    if (abs(b-a) <= 1) return;
    double g = (a + b)/2;
    draw a line from (g, c) to (g, d);
    if (e > g) {
        func2(g, b, c, d, e, f);
    }
    if (e < g) {
        func2(a, g, c, d, e, f);
    }
}
```

```
func2(double a, b, c, d, e, f) {
    if (abs(d-c) <= 1) return;
    double g = (c + d)/2;
    draw a line from (a, g) to (b, g);
    if (f > g) {
        func1(a, b, g, d, e, f);
    }
    if (f < g) {
        func1(a, b, c, g, e, f);
    }
}
```

- Selv om funksjonene `func1` og `func2` i utgangspunktet ikke gjør noe nyttig, minner de (tilsammen) om en todimensjonal variant av en algoritme i pensum. Hvilken?
- Tegn resultatet av å kjøre `func1(0, 160, 0, 160, 160, 150)`
- Tegn resultatet av å kjøre `func1(0, 160, 0, 160, 70, 50)`
- Anta at vi setter $m = b - a$ og $n = d - c$. Uttrykk kjøretiden til `func1` som funksjon av m og n . Velg selv hva du mener er mest fornuftig av *best-case* eller *worst-case* kjøretid. Bruk Θ -notasjon.

3 SIF8010 august 2003 - Oppgave 3

N forskjellige heltall settes inn i et binært søketre i tilfeldig rekkefølge. La $Q(N)$ betegne den gjennomsnittlige (forventede) dybden til det minste elementet i treet. Spesielt er $Q(0) = 0$, $Q(1) = 1$ og $Q(2) = 1.5$.

- Utled en rekurrensformel for $Q(N)$, der $N > 0$.
- Bruk rekurrensformelen fra **a** til å finne et eksplisitt uttrykk for $Q(N)$ (dvs. uten bruk av Q til høyre for likhetstegnet).

4 TDT4120 desember 2003 - Oppgave 2

Anta at du har tre tabeller, A , B og C , med positive reelle tall. Hver av tabellene har lengde n .

- Du vil finne et segment $A[i\dots j]$ slik at $A[i] \times A[i+1] \times \dots \times A[j]$ blir størst mulig. Hvordan vil du gå fram? Referer gjerne til algoritmer i pensum. Hva blir kjøretiden?
- Du ønsker å finne ut om det finnes tre tall a , b og c , slik at A inneholder a , B inneholder b og C inneholder c og slik at $a + b + c = x$ for en gitt x . Beskriv kort (enten med pseudokode eller dine egne ord) en algoritme som løser problemet i $\Theta(n^2 \lg n)$ tid, worst-case.
- Du ønsker å løse samme problem som i oppgave b, men kan nå anta at A , B og C er heltallstabeller, og at heltallene faller i et tallområde fra 1 til M . Beskriv kort (enten med pseudokode eller dine egne ord) en algoritme som løser problemet i $\Theta(n^2)$ tid, worst-case. Gi også (i stikkordsform) eventuelle antagelser om M og maskinvaren du bruker for at kjøretiden skal gjelde.

5 TDT4120 desember 2003 - Oppgave 3

Du har oppdaget følgende pseudokode i en gammel lærebok i algoritmer. Du er usikker på hvilket språk læreboken er skrevet på, og har litt problemer med å skjønne enkelte av ordene i pseudokoden:

```
BRILLIG(A[1...N]):
  if N = 1:
    return A[1], A[1]
  slithy ← ⌊N/2⌋
  gyre, gimble ← BRILLIG(A[1 . . . slithy])
  wabe, mimsy ← BRILLIG(A[slithy + 1 . . . N])
  if gyre < wabe:
    borogroves ← gyre
  else
    borogroves ← wabe
  if gimble < mimsy:
    mome ← mimsy
  else
    mome ← gimble
```

return *borogroves, mome*

- a) Hva gjør algoritmen BRILLIG?
- b) Anta at $N = 256$. Hvor mange sammenligninger av typen *gyre < wabe* og *gimble < mimsy* utføres totalt? (Vi er kun ute etter ett tall.)
- c) Sett opp en eksakt rekurrens som uttrykker antall sammenligninger som en funksjon $C(N)$. Anta at $N = 2^M$ for et heltall M .
- d) Løs rekurrensen i oppgave c. Uttrykk svaret eksakt, uten bruk av asymptotisk notasjon.

Du bestemmer deg for å optimalisere algoritmen. Du endrer utsagnet

```
if  $N = 1$ :  
    return  $A[1], A[1]$ 
```

til det følgende:

```
if  $N = 2$ :  
    if  $A[1] < A[2]$ :  
        return  $A[1], A[2]$   
    return  $A[2], A[1]$ 
```

- e) Sett opp en eksakt rekurrens som uttrykker antall sammenligninger som en funksjon $C(N)$. Anta at $N = 2^M$ for et heltall M . Rekurrensen skal også telle sammenligninger av typen $A[1] < A[2]$.
- f) Løs rekurrensen i oppgave e. Uttrykk svaret eksakt, uten bruk av asymptotisk notasjon.

6 TDT4120 desember 2003 - Oppgave 4

- a) Anta at du har en urettet graf. Du vet at hver node har maksimalt 3 naboer. Argumenter svært kort for at det er mulig å finne en to-farging av grafen som er slik at hver node maksimalt har 1 konflikt (nabo med samme farge).

Hint: Bruk det totale antall konflikter i argumentasjonen.

Anta at du har oppgitt et flytnettverk definert ved følgende kapasitetsmatrise:

$$C = \begin{vmatrix} 0 & 7 & 6 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Her kan vi for eksempel se at kapasiteten mellom node 4 og node 5 er $C[4, 5] = 5$.
Anta at node 1 er kilden og at node 7 er sluket.

- b) Hvor mange mulige snitt finnes det mellom kilde og sluk?
- c) Hvordan kan man bruke FORD-FULKERSON til å finne et minimalt snitt? Skriv kort.
- d) Finn et minimalt snitt i flytnettverket. Beskriv snittet ved å oppgi alle nodene som befinner seg på samme side som kilden.
- e) Et sett med stier i en graf $G = (V, E)$ er kant-disjunkte hvis ingen kant i E inngår i mer enn en av stiene i settet. Beskriv en algoritme som finner (det maksimale) antall kant-disjunkte stier mellom to gitte noder s og t i en urettet graf.
- f) Du ønsker å øke den maksimale flyten i et flyt-nettverk så mye som mulig, men du får bare lov til å endre kapasiteten på en kant. Hvordan finner du en slik kant? (Bruk pseudo-kode eller egne ord. Anta at du har algoritmer tilgjengelig for å finne maks-flyt og et minimalt snitt.) Hva blir kjøretiden (*worst-case*, i Θ -notasjon)? Vil det alltid være mulig å finne en slik kant? (Begrunn svaret.)

7 TDT4120 desember 2003 - Oppgave 5

Det finnes en algoritme som løser problemet

Du har oppgitt et sett S bestående av N reelle tall, samt et reelt tall T og et heltall $K \leq N$.
Finnes det en delmenge Q av S med K elementer, der summen av elementene i Q er maksimalt lik T ?

i $\Theta(N)$ tid. Er det rimelig å tro at vi kan finne en like effektiv løsning på følgende problem? Begrunn svaret.

Du har oppgitt et sett S bestående av N reelle tall, samt et reelt tall T og et heltall $K \leq N$.
Finnes det en delmenge Q av S med maksimalt K elementer, der summen av elementene i Q er lik T ?

8 TDT4120 desember 2003 - Oppgave 6

SUM(N)

```
top ← 1; S[top] ← N; S[0] ← 2; stacksum ← N
WRITE('N = ')
while top > 0
  for i in 1..top - 1
    WRITE(S[i], ' + ')
  WRITELINE(S[top])
  while S[top] = 1
    top ← top - 1
```

```

    stacksum ← stacksum - 1
if top > 0:
    S[top] ← S[top] - 1
    stacksum ← stacksum - 1
    while stacksum < N
        top ← top + 1
        if N - stacksum ≤ S[top - 1]
            S[top] ← N - stacksum
            stacksum ← N
        else
            S[top] ← S[top - 1]
            stacksum ← stacksum + S[top]
WRITE(' = ')

```

Hva skriver funksjonen SUM ut hvis $N = 6$? Anta at S er en tabell med så mye plass som er nødvendig. Anta at funksjonen WRITELINE skriver ut argumentene sine (uten mellomrom imellom) og starter en ny linje, mens WRITE skriver ut argumentene sine (uten mellomrom imellom) uten å starte en ny linje.

9 TDT4120 juni 2004 - Oppgave 3

- a) Hva er forskjellen på O , Θ og Ω ?
- b) Er $2^n \in \Omega(2^{n+1})$? Gi en kort begrunnelse.
- c) Er $3^n \in O(2^n)$? Gi en kort begrunnelse.
- 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.
- 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.
- f) Etter en stund blir algoritmedesigneren lei av å stokke kort. I stedet for å stokke begge halvdelene rekursivt stokker hun nå bare *en* tilfeldig valgt halvdel rekursivt. Hva blir kjøretiden til algoritmen nå? Sett opp en rekurrens og skriv løsningen med asymptotisk notasjon.
- 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 *en* av halvdelene rekursivt. Hva blir kjøretiden nå? Sett opp en rekurrens og skriv løsningen med asymptotisk notasjon.

10 TDT4120 juni 2004 - Oppgave 4

- 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 $a_i \dots a_j$ 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.

- 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?

- 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 *brute force*-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.)

- 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 $h(x)$ (x antall esker) til det høyeste tårnet som kan bygges hvis den totale vekten maksimalt kan være x .

- 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.

11 TDT4120 desember 2004 - Oppgave 3

- a) Her følger en liste med betegnelser på kjøretidsklasser - ordne dem i stigende rekkefølge: *konstant*, *eksponensiell*, *lineær*, *kubisk*, *kvadratisk*, *faktoriell*, *logaritmisk*, *$n \log n$* . Ingen begrunnelse er nødvendig.

- b) Hva gjør følgende programsnitt, og hva blir kjøretiden? Bruk Θ -notasjon og begrunn svaret. Skriv kort.

```
AZATHOTH(A[1...n])
  i ← 1
  j ← n
  while i < j
    if A[i] > A[j]
```

```

        i ← i + 1
    else
        j ← j - 1
    return A[i]

```

- c) Hva gjør følgende programsnitt og hva blir kjøretiden? Funksjonen *floor()* runder ned til nærmeste heltall. Bruk Θ -notasjon og begrunn svaret. Skriv kort.

```

AZAROTH(i, j, A[1...n], B[1...n])
  if i = j
    if A[j] > B[j]
      return A[j] - B[j]
    else
      return B[j] - A[j]
  k ← floor((i + j)/2)
  return AZAROTH(i, k, A, B) + AZAROTH(k + 1, j, A, B)

```

- d) En algoritme har et fast sett med instruksjoner som alltid utføres, et sett med instruksjoner der antallet steg er en fast prosent av problemstørrelsen og k rekursive kall på en k -del av problemet. Hva blir kjøretiden til algoritmen? Bruk Θ -notasjon og begrunn svaret med en kort skisse av utregningen din.

12 TDT4120 desember 2004 - Oppgave 4

Hvis vi fargelegger nodene i en graf og to nabonoder har samme farge, kalles dette en konflikt. En graf er k -fargbar hvis den kan fargelegges med k farger uten at det oppstår konflikter.

- a) Finnes det en kjent polynomisk algoritme for å avgjøre om en graf er k -fargbar, for en vilkårlig k ? Hvis ja, hvordan virker denne og hva er kjøretiden (i Θ -notasjon)? Hvis nei, hvorfor ikke?
- b) Hvordan vil du løse problemet hvis $k = 2$? Angi øvre og nedre asymptotiske grenser for kjøretiden.
- c) Anta at du har en graf som *ikke* er tofargbar. Du ønsker å fargelegge den med to farger slik at antall konflikter blir minst mulig. Gi en kort beskrivelse av en enkel *branch and bound*-løsning på dette problemet.

13 TDT4120 desember 2004 - Oppgave 5

For hver av algoritmene under, hvilke av de følgende asymptotiske kjøretidsklassene kan brukes til å beskrive algoritmens kjøretid? Du kan ikke anta noe om probleminstansene.

$\Omega(n), \Omega(n \lg n), \Omega(n^2), O(n), O(n \lg n), O(n^2), \Theta(n), \Theta(n \lg n), \Theta(n^2)$

Oppgi de relevante kjøretidsklassene (for eksempel ' $\Omega(n^2)$, $O(n)$ ') eller skriv 'Ingen' hvis ingen av klassene kan brukes. Hvis flere klasser kan brukes skal alle oppgis i svaret. Gi en kort begrunnelse til hver deloppgave.

- a) Innsetting av n verdier i et tomt binært søketre.
- b) Sortering ved innsetting.
- c) MergeSort.

14 TDT4120 desember 2004 - Oppgave 6

Student Lurvik skal summere opp n positive flyttall $x[1] \dots x[n]$, slik at avrundingsfeilen blir minst mulig. Avhengig av hvordan dette gjøres, kan de ulike tallene delta i et ulikt antall summerasjoner. For eksempel vil $x[1]$, $x[2]$, $x[3]$ og $x[5]$ delta i flere summerasjoner enn $x[6]$ i følgende summeringsrekkefølge:

$$(((x[1] + x[3]) + (x[2] + x[5])) + x[6])$$

Merk her at både tallrekkefølgen og parantessettingen velges fritt.

Hvor stor avrundingsfeilen for en sum av to flyttall blir er avhengig av hvor stor summen er (større sum gir større feil). Lurvik innser at det kan være lurt å summere slik at de små flyttallene deltar i flest mulig summerasjoner (det vil si, de kommer "dypt" i parantessettingen), mens de store flyttallene deltar i færrest mulig. Hvis $p(i)$ er parentes-dybden til flyttall $x[i]$ (antall parenteser utenfor elementet), så definerer Lurvik *feilen* til en mulig løsning som

$$p(1) \cdot x[1] + p(2) \cdot x[2] + \dots + p(n) \cdot x[n]$$

Lurvik ønsker nå å finne en parantessetting som minimaliserer dette feiluttrykket. Skisser en løsning på problemet. Hvilken designmetode bruker du? Vis, med støtte i pensum, at løsningen er korrekt.