Obligatorisk Oppgave 1

Algoritmer og Datastrukturer ITF20006

Lars Vidar Magnusson

Frist 6.2.15

Den første obligatoriske oppgaven tar for seg de fem første forelesningene, som i hovedsak har dreiet seg om hvordan man analyserer algoritmer.

Praktisk Informasjon

Det er med disse oppgavene dere skal få den nødvendige erfaringen dere trenger, så det kreves at alle innleveringer skal være individuelt arbeid. Det er selvsagt til lov å hjelpe hverandre, men den endelige besvarelsen skal være et resultat av eget arbeid. Besvarelser som viser tegn til plagiering vil bli underkjente.

Dere kan selv velge i hvilket språk dere implementerer algoritmene, men hjelpefunksjoner vil bli gitt i Java. Hvis dere skulle trenge direkte hjelp med koden, kan det være greit å holde seg til enten Java eller C#.

Dere skal lage en enkel rapport som beskriver besvarelsen og samler alle tekstlige svar. Rapporten skal struktureres i henhold til oppgaven. Koden som løser programmeringsoppgavene skal være kjørbar og skal ta en inputfil som argument fra kommandolinjen, for så å skrive ut løsningen. Koden skal leveres i ren kildekode (ikke et prosjekt fra en eller annen IDE med en masse unødvendige filer), og det skal legges ved et enkelt skript som kompilerer koden.

Både rapport, kode og skript skal pakkes i en tarball og skal sendes til meg på lars.v.magnusson@hiof.no med emne "ITF20006 $OPPGAVE $STUDENT".

Oppgave 1

Analyser kjøretiden til de følgende algoritmene gitt i pseudokode. Det skal gjøres både en detaljert analyse, en asymptotisk analyse. Dere kan anta at hver linje er tar én tidsenhet.

```
OPPGAVE1A(n)
1  sum = 0
2  for i = 1 to n
3        sum = sum + i
4  return sum
```

```
OPPGAVE1B(n)
1  sum = 0
2  for i = n/2 to n
3       for j = n/2 to n
4            sum = sum + i
5            sum = sum + j
6  return sum
```

```
OPPGAVE1C(n)
1  sum = 0
2  for i = 1 to n − 1
3       sum = sum + i
4  for j = 1 to n − 1
5       sum = sum + j
6  return sum
```

```
OPPGAVE1D(n)
1  sum = 0
2  for i = 1 to n
3       for j = 1 to n
4            for k = 1 to n
5                 sum = sum + k
6  return sum
```

```
OPPGAVE1E(n)
1  sum = 0
2  for i = 1 to n
3       for j = 1 to n
4            for k = 1 to n
5                 sum = sum + i
6                 sum = sum + j
7                 sum = sum + k
8  return sum
```

```
OPPGAVE1F(n)
1  sum = 0
2  i = 1
3  while i ≤ n
4       sum = sum + i
5       i = 2i
6  return sum
```

```
OPPGAVE1G(n)
1  sum = 0
2  i = 1
3  while i ≤ n
4       for j = 1 to n
5            sum = sum + j
6       i = 2i
7  return sum
```

Oppgave 2

A

Finn en gjetning for en asymptotisk øvre grense for recurrence ligningen under ved hjelp av rekursjonstrær. Bevis den asymptotiske grensen med substitusjonsmetoden.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 4T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

B

Hvilke metoder for å løse recurrence-ligninger er brukbare på ligningen under. Begrunn svaret. Bruk en av de applikable metodene for å komme frem til en øvre grense.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(n-1) + \Theta(n) & \text{if } n > 1 \end{cases}$$

C

Benytt masterteoremet til finne den asymptotiske grensen for recurrencelignigen under.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 16T(n/2) + \Theta(n^4) & \text{if } n > 1 \end{cases}$$

Oppgave 3

I denne oppgaven skal dere implementere og analysere algoritmen SQUARE-MATRIX-MULTIPLY-RECURSIVE som vi gikk gjennom i forelesningen.

SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)

```
1  n = A.rows
2  let C be a new n × n matrix
3  if n == 1
4       c11 = a11 · b11
5  else partition A, B, and C
6       C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
              +SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
7       C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
              +SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
8       C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
              +SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
9       C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
              +SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
10 return C
```

Dere skal implementere to versjoner, en som kopierer delmatrisene inn i nye delmatriser, og en som bruker indeks-variable for å angi hvilken del av

matrisen det jobbes med. Du kan ta utgangspunkt i kildekoden som er lagt ut sammen med oppgaven.*Matrix.java* kan brukes i den versjonen som kopierer, mens *SubMatrix.java* kan brukes i den versjonen som bruker indeks-variable.

Dere skal resonnere dere frem til en recurrence-ligning for begge utgavene, og forklare hvorfor samme ligning kan brukes til å beskrive begge to. Dere skal så gjøre en analyse ved hjelp av rekursjonstrær slik at dere kommer frem til en gjetning for en funksjon som beskriver kjøretiden til algoritmen. Denne skal dere så verifisere med substitusjonsmetoden.

Den faktiske kjøretiden til de to utgavene av algoritmen skal så testes med en rekke forskjellige input med forskjellige størrelser. Hva stor er forskjellen i kjøretid mellom de to utgavene? Er det noen forskjell i minneforbruket? Hvilken versjon, om noen, er den beste? Dere skal diskutere disse spørsmålene og eventuelt andre viktige observasjoner dere har gjort i rapporten.