# Eksamen i fag
# SIF8010 Algoritmer og Datastrukturer
# Fredag 9. August 2002, kl 0900-1500

**Faglig kontakt under eksamen**: Arne Halaas, tlf. 73593442.
**Hjelpemidler**: Alle kalkulatortyper tillatt. Alle trykte og håndskrevne hjelpemidler tillatt.
**Viktig**: Svar skal fortrinnsvis skrives på vedlagt svarark, og kun der. Om nødvendig kan utfyllende svar legges ved på egne ark. Oppgavene er ment å kunne besvares konsist.

## Oppgave 1

Betrakt den rettede grafen $G=(V, E)$ gitt ved følgende kantvekt-matrise:

$$W = \begin{bmatrix} \infty & 23 & 157 & 60 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 166 & \infty & \infty \\ \infty & 187 & \infty & 195 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 80 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 176 \\ \infty & \infty & \infty & \infty & \infty & \infty & 173 \\ \infty & \infty & 183 & \infty & \infty & \infty & \infty \end{bmatrix}$$

Fra denne matrisen kan vi for eksempel se at det er en kant fra node 1 til node 2 med vekt 23, fordi elementet $w_{12} = 23$ (der $w_{ij}$ er elementet i rad $i$ og kolonne $j$). Derimot er det ingen kant fra node 3 til node 6, fordi $w_{36} = \infty$.

Anta at følgende regel gjelder: Hvis en algoritme kan velge mellom flere likeverdige alternativer velger den alltid noden med lavest tallverdi. For eksempel, hvis dybde-først-søk kan besøke node 4 eller node 6 vil den besøke node 4.

a) I hvilken rekkefølge vil nodene besøkes (for første gang) dersom et dybde-først-søk utføres med utgangspunkt i node 1?

b) I hvilken rekkefølge vil nodene besøkes (for første gang) dersom et bredde-først-søk utføres med utgangspunkt i node 1?

c) I hvilken rekkefølge vil nodene legges til i spenntreet dersom Prims algoritme utføres med utgangspunkt i node 1? Anta her at algoritmen virker på den underliggende urettede grafen med de samme kantvektene. For eksempel har vi både $w_{32} = 187$ og $w_{23} = 187$.

d) I hvilken rekkefølge vil nodene besøkes (for første gang) dersom Dijkstras algoritme utføres med utgangspunkt i node 1 på den opprinnelige, rettede grafen?

## Oppgave 2

a) Hvilke av følgende tre utsagn er sanne? (Det kan være ingen, ett, eller flere sanne utsagn. Hvis du mener ingen er sanne, lar du feltet på svararket stå tomt.)

1) $n^{100} = O(2^n)$
2) $2^n = O(n^{100})$
3) $2^n = \Theta(3^n)$

b) Gi en kort begrunnelse for hvert av de tre utsagnene i deloppgave a). Hvorfor er de sanne/usanne?

c) Anta at du har et balansert binært søketre med $n$ elementer. Du ønsker å skrive ut alle elementene i verdiområdet fra $x$ til $y$. Anta at det finnes $k$ elementer i dette verdiområdet. Algoritmen din må finne alle disse elementene og skrive dem ut. Hva blir kjøretiden? Uttrykk svaret i theta-notasjon.

d) Hva er hensikten med heap-strukturen? Hvorfor kan man ikke like gjerne bruke binærtrær (binære søketrær)? Man kan bruke binærtrær til å finne største (og minste) element, og den gjennomsnittlige asymptotiske kostnaden for å sette inn og ta ut elementer er jo den samme. Hvilken fordel er det da heaper (hauger) har (for sitt anvendelsesområde)?

## Oppgave 3

Betrakt følgende program i Java-lignende pseudokode:

```
010  int mystery(int[] b, int[] c) {
020      int m = b.length
030      int n = c.length
040      int[][] a = new int[m][n]
050      initialize(a)
060      return f(b, c, m–1, n–1, a)
070  }
080  int f(int[] b, int[] c, int d, int e, int[][] a) {
090      if (d== –1 || e== –1)
100          return 0
110      else if (b[d]==c[e])
120          a[d][e] = f(b, c, d–1, e–1, a) + 1
130      else
140          a[d][e] = max(f(b, c, d–1, e, a), f(b, c, d, e–1, a))
150      return a[d][e]
160  }
```

Linjenummereringen er kun gjort av hensyn til deloppgave d). Anta at funksjonen *initialize* setter alle elementene i tabellen $a$ til –1, og at funksjonen $max(x, y)$ returnerer det største av tallene $x$ og $y$.

**Husk**: Selv om du er usikker på enkelte detaljer i programmet kan det likevel hende du kan klare å besvare spørsmålene.

a) Programmet implementerer en algoritme fra pensum. Hvilken?

b) Kjøretiden til denne enkle rekursive implementasjonen er eksponensiell. Problemet som løses har imidlertid en egenskap som gjør at kjøretiden kan forbedres betraktelig. Hvilken egenskap er det?

c) En kjent metode fra pensum kan brukes for å forbedre rekursive programmer når problemet har egenskapen beskrevet i deloppgave b). Metoden lar programmet forbli rekursivt. Hva heter denne metoden?

d) Du skal nå forbedre kjøretiden ved å legge til en enkelt kodelinje. Skriv denne kodelinjen (i Java eller pseudokode) med tilhørende linjenummer på svararket. Oppgi et linjenummer som angir hvor den nye linjen skal settes inn, for eksempel 075 for å sette inn linjen mellom linje 070 og 080.