

## EKSAMEN I FAG 45011 ALGORITMER OG DATASTRUKTURER

Fredag 20. desember 1996, kl 0900 - 1300

Faglig kontakt under eksamen: Arne Halaas, tlf 73 593442  
Alle kalkulator typer tillatt. Alle trykte og håndskrevne hjelpemidler tillatt.

**Merk:** Det anbefales å ikke skrive mer enn totalt 4 sider.

### Oppgave 1 (20 %).

Anta at  $x, y, u$  og  $v$  er variable av type "vektor" (PASCAL):  
Type indeks =  $[1..max]$  ; vektor = array indeks of real;

Gitt den rekursive funksjonen:

```
function P(x,y : vektor; i,j : indeks) : real;
begin
  if i > j then P := 0.0
  else if i = j then P := x[i] * y[i]
  else P := P(x,y,i,(i+j) div 2) + P(x,y,(i+j) div 2 + 1,j);
end;
```

"A div B" er heltallsdivisjon der desimaldel fjernes (trunkering).

- (a) (8 %) Hvor mange multiplikasjoner (M), henholdsvis addisjoner (A), utføres totalt ved kallet  $P(u,v,1,7)$  ?
- (b) (6 %) Angi tidskompleksiteten forbundet med kallet  $P(u,v,1,n)$  ved  $\Theta(f(n))$ . Finn  $f(n)$ .
- (c) (6 %) Skriv en ikke-rekursiv, effektiv erstatning for funksjonen P.

### Oppgave 2 (10 %).

Første fase i haugsortering (Heapsort) består i å "rette" treet, før den til enhver tid minste verdi (numerisk eller alfabetisk) kan trekkes ut fra rotnoden. Anta at treet før retting består av en tekstvektor  $Z[1..12]$  med innhold K,O,M,P,L,E,K,S,I,T,E,T (stigende indekser). Rotnoden  $Z[1]$  vil altså initielt inneholde tegnet "K", mens  $Z[12]="T"$ . Anta at laveste indeks har prioritet ved opprykk dersom det er 2 likeverdige alternativer.

- a) (10%) Hva inneholder  $Z[1..12]$  etter at treet er blitt rettet?

**Oppgave 3 (18 %)**

Ofte er det vanskelig å finne en standard algoritme som passer helt til et gitt praktisk problem. En har da 3 alternativer å velge mellom:

(1) å skrive en ny algoritme, (2) å modifisere en kjent algoritme, eller (3) å modifisere problemet (datasettet) slik at dette passer til en kjent algoritme. Alternativ (1) er klart mest arbeidskrevende.

Anta at vi skal løse korteste-vei-problemer, med 1 kildenode og positive kant-lengder (vektet), der det er mange korteste veier å velge mellom.

Anta at det er viktig for oss å finne en korteste vei som består av så få kanter som mulig.

(a) (8%) Vurder alternativ (2): Kan vi modifisere Dijkstras algoritme slik at denne finner en korteste vei som består av et minimalt antall kanter?

Hvis JA: Vis hvordan du vil modifisere Dijkstras algoritme (ta med kun de nødvendige endringer, henvis til Initialize-S-S, Extract-Min, Relax).

Hvis NEI: Gi begrunnelse.

(b) (10%) Vurder alternativ (3): Kan datasettet modifisere slik at Dijkstras algoritme finner en korteste vei som består av et minimalt antall kanter?

Hvis JA: Vis hvordan du vil modifisere datasettet (grafene  $G$ , vektene  $w$ ) slik at Dijkstras algoritme finner en korteste vei som består av et minimalt antall kanter.

Hvis NEI: Gi begrunnelse.

**Oppgave 4 (16 %)**

Gitt en urettet graf  $G = (V,E)$ , der  $V = \{V_1, V_2, \dots, V_n\}$  er  $n$  punkter i  $(x,y)$ -planet, og der  $E$  er  $n(n-1)/2$  kanter (rette linjestykker) som forbinder alle nodepar.  $G$  er i dette tilfellet en såkalt geometrisk, eller Euklidisk, graf.

Gitt også følgende funksjon som rekursivt bygger et tre basert på  $G$  :

```
function Tre(G,n)
  if n= 1 then Tre := {V1} else
  begin T := Tre(G- $\{V_n\}$ ,n-1)  /fjerner  $\{V_n\}$  med dens kanter/
    La u være en node i T med korteste avstand til node  $V_n$ 
    Tre := T  $\cup$   $\{V_n\} \cup \{(u, V_n)\}$  /tilkople  $\{V_n\}$  og kant til u/
  end
```

a) (8 %) Produserer funksjonen Tre minimale spenntreer for geometriske grafer? (Begrunn svaret.)

b) (8%) Vil følgende utkast til rekursiv algoritme kunne brukes til å finne minimale spenntreer i geometriske grafer? (Begrunn svaret.)

I. Del  $G = (V,E)$  inn i 2 halvdelene  $G = G_1 \cup G_2$ , der  $x$ -verdiene for  $G_1$ 's noder er mindre eller lik  $x$ -verdiene til  $G_2$ 's noder.

II. Finn minimale spenntreer  $T_1$  og  $T_2$  for henholdsvis  $G_1$  og  $G_2$ .

III. Forbind  $T_1$  og  $T_2$  med en kortest mulig kant.

### **Oppgave 5 (22 %).**

Du blir her bedt om å vurdere følgende sorteringsalgoritme:

Algoritme S: Sortering av  $n$  verdier  $A[1..n]$ :

Initialisering:

Steg1: Splitt  $A$  i  $r$  grupper  $G[1..r]$  hver med  $s$  elementer. Anta at  $r \times s = n$ .

Steg2: Finn minste  $A$ -verdi i hver gruppe. Anta  $\theta(s)$  tid pr. gruppe.

Kall disse  $r$  minsteverdiene  $m[1..r]$ .

Gjenta følgende to steg  $n$  ganger:

Steg3: Finn minste verdi,  $m[k]$ , blant verdiene i  $m[1..r]$ . Anta  $\theta(r)$  tid.

Steg4:  $m[k]$  kan nå flyttes (til ett eller annet sted) og erstattes med den minste gjenværende verdien i gruppe  $k$ . Anta  $\theta(s)$  tid.

Vi trekker her ut en etter en verdi fra  $A$ , i stigende orden. Vi bruker enkleste sort sekvensielt gjennomløp for å finne minste-verdiene underveis. Vi antar derfor samme tidsforbruk  $\theta(r)$  og  $\theta(s)$ , for henholdsvis Steg 3 og Steg 4 ved alle gjennomløp.

- (a) (6%) Hva er tidskompleksiteten for Algoritme S uttrykt ved  $n, r$  og  $s$ ?
- (b) (8%) Hvordan bør  $r$  og  $s$  velges for at Algoritme S skal bli mest mulig effektiv? Uttrykk tidskompleksiteten ved kun  $n$ .
- (c) (8%) Kan S konkurrere med Quicksort for noen verdier av  $n$ ? (Drøft)

### **Oppgave 6 (14 %).**

Det anbefales å løse denne oppgaven sist.

Anta at vi har samlet opp  $n$  problemer på en PC som står i et nettverk.

Disse problemene er tidkrevende og de skal derfor sendes til 2 superdatamaskiner for å bli løst der. Det er kjent at problem  $P(i)$  krever  $t(i)$  sekunder av superdatamaskinens CPU-tid,  $i = 1, 2, \dots, n$ .

Målet vårt er å dele de  $n$  problemene i 2 deler, der  $T_1$  er summen av  $t(i)$ -verdier i del 1,  $T_2$  er summen av  $t(i)$ -verdier i del 2, og der  $\max(T_1, T_2)$  er minimal.

- (a) (14%) Skisser (kort) en algoritme-ide for delingen, og gi et best mulig estimat for algoritmens tidskompleksitet.