

NTNU
Norges teknisk-naturvitenskapelige
universitet

**Fakultet for informasjonsteknologi,
matematikk og elektroteknikk**

**Institutt for datateknikk
og informasjonsvitenskap**

BOKMÅL



AVSLUTTENDE EKSAMEN I

TDT4120/IT1105

ALGORITMER OG DATASTRUKTURER

Mandag 5. desember 2005

Kl. 09.00 – 13.00

Faglig kontakt under eksamen:

Arne Halaas, tlf. 416 61 982

Magnus Lie Hetland, tlf. 918 51 949

Hjelpemidler:

Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

Sensurdato:

5. januar 2006. Resultater gjøres kjent på <http://studweb.ntnu.no> og sensurtelefon 81548014.

Viktig:

Les hele oppgavesettet før du begynner. Les oppgaveformuleringene grundig. Det er angitt i poeng hvor mye hver deloppgave teller ved sensur. Gjør nødvendige antagelser der dette er nødvendig.

Skriv kort og konsist. Skriv fortrinnsvis i rutene på oppgavearket.

Oppgave 1 (18%)

Anta gitt en heltallsvektor $V[1..n]$

- a) Finn en mest mulig effektiv algoritme for å avgjøre om det eksisterer en undermengde (et utvalg) av k elementer i V med sum høyst lik (\leq) verdien T .

(3 poeng)

Det er gitt at $n/4 \leq k \leq n/2$ og at $n > 1000$.

- b) Hva er tidskompleksiteten for ditt algoritmeforslag i a) i verste tilfelle (*worst case*)?

(2 poeng)

- c) Skisser kort en så effektiv som mulig algoritme for å finne de $k < 6$ minste verdiene i V .

(3 poeng)

$\text{SORT100}(V[i..j])$ er en gitt $O(j-i)$ -metode som sørger for at de inntil 100 minste verdiene i $V[i..j]$ blir plassert i sortert rekkefølge i $V[i, i+1, i+2, \dots, i+k]$, der $k = \min(i+99, j)$, mens de øvrige elementene, hvis disse eksisterer, forblir usorterte og plasseres i $V[i+k+1, i+k+2, i+k+3, \dots, j]$.

- d) Hva er tidskompleksiteten til en algoritme som bruker SORT100 gjentatte ganger for å sortere $V[1..n]$, der n er mye større enn 100? Begrunn svaret.

(5 poeng)

Anta nå at heltallsvektoren $V[1 \dots n]$ er sortert. Dersom vi søker etter verdien x i V kan vi undersøke midt-verdien i V i forhold til x og derved eliminere halvparten av V i det fortsatte søket. Binærsøk er en algoritme som gjentar denne operasjonen ved å halvere størrelsen på den resterende sekvensen inntil x enten er funnet eller inntil en kan slå fast at x ikke er blant V -verdiene.

- e) Skriv en kort, men detaljert, (pseudo)kode for Binærsøk og angi kjøretiden for algoritmen i verste tilfelle.

Kjøretid: $O(\quad)$
(5 poeng)

Oppgave 2 (17%)

- a) La H være den binære haugen (*binary heap*) $[2, 4, 3, 4, 7, 6, 4, 5, 6, 7, 9, _]$. Hvordan ser H ut etter at du setter inn verdien 1? (Den siste posisjonen er i utgangspunktet ledig.)

(3 poeng)

- b) Vis hvordan en tabell (*array*) $A = [5, 9, 7, 4, 0, 2, 8, 8]$ ser ut etter hver SWAP-operasjon (ombytting av to tall) fram til fullført sortering ved Quicksort. Den siste verdien (den lengst til høyre) i tabellen (eller deltabellen under betraktning) skal alltid velges som pivot-element. Anta at PARTITION er implementert som oppgitt under. Fyll ut tabellen til høyre.

5	9	7	4	0	2	8	8

(4 poeng)

PARTITION(A, p, r)

$x \leftarrow A[r]$

$i \leftarrow p - 1$

for $j \leftarrow p$ **to** $r - 1$

if $A[j] \leq x$

$i \leftarrow i + 1$

 SWAP($A[i], A[j]$)

SWAP($A[i + 1], A[r]$)

return $i + 1$

Stipendiat Smartens hevder å ha klekket ut en sorterings-algoritme som kan ta inn en gyldig binærhaug (*binary heap*) med n verdier og sortere disse på $O(n)$ tid. Smartens hevder at han unngår den fundamentale $\Omega(n \log n)$ -tid laveste grense for sammenligningsbasert sortering fordi elementene i den binære haugen allerede er delvis sortert.

c) Gi en så overbevisende begrunnelse som mulig for at Smartens dessverre tar feil.

(10 poeng)

Oppgave 3 (15%)

Vi har gitt 3 vektorer $A[1..n]$, $B[1..n]$, $C[1..n]$, alle med verdier som er fødselsdatoer på formen "ddmmåå", f.eks.: $A[1342] = 180368$.

Vi har som mål å finne ut hvilke datoer som er med i både A , B og C .

Du skal lage en enkel og særdeles effektiv algoritme for å framskaffe en vektor D som inneholder disse felles fødselsdatoene. Tenk på konstantleddene i metoden også.

a) Beskriv en slik algoritme SNITTABC. Uttrykk algoritmens tidskompleksitet ved Θ -notasjonen.

Kjøretid: $\Theta(\quad)$

(15 poeng)

Oppgave 4 (15%)

Vi har gitt et olje-transportnettverk N med 6 noder: A (kilde), B , C , D (sluk), E , F , der oljerørens kapasitet er gitt ved

$$AB:3, AC:3, BC:2, BE:3, CF:2, EF:4, ED:2, FD:3.$$

- a) Du skal beregne maksimal flyt fra A til D ved den generelle Ford-Fulkerson-metoden. For at svaret ditt skal bli entydig i de tilfeller du har flere stivalg, skal du besøke noder i alfabetisk stigende rekkefølge, dvs. B før C , E før F , etc. Fyll ut, i tabellen nedenfor, de strømforøkende stiene (*augmenting paths*) i den rekkefølge de blir oppdaget, samt den netto flytøkning stien representerer. Nedenfor er første sti fylt inn. Vær nøye med å følge de beskrevne reglene. (Merk at du ikke nødvendigvis trenger å bruke alle radene i tabellen.)

(8 poeng)

Strømforøkende sti (<i>augmenting path</i>)	Netto flytøkningbidrag
A-B-C-F-D	2

- b) Hvilke minimale snitt finnes i nettverket over? (Oppgi hvert snitt som en kantliste.)

(7 poeng)

Oppgave 5 (25%)

På en nyttårsfest er det invitert N gutter og N jenter, og arrangøren vil oppnå at så få som mulig får en "ikke spesielt ønsket" bordkavaler/dame til bords. Det er av den grunn blitt gjennomført en spørreundersøkelse blant deltakerne, og data fra denne er samlet i to matriser G og P .

Her er $G(i, j) = 0$ dersom gutt nr. i ikke spesielt ønsker jente nr. j , $j = 1, 2, \dots, N$; ellers er $G(i, j) = 1$.

Tilsvarende er $P(j, i) = 0$ dersom jente nr. j ikke spesielt ønsker gutt nr. i , $i = 1, 2, \dots, N$; ellers er $P(j, i) = 1$.

Ønskene er ikke nødvendigvis gjensidige; eksempelvis kan $G(5, 19) = 1$ mens $P(19, 5) = 0$.

Vi definerer nå Problem PAR: Finn en par-sammensetting som er slik at så få personer som mulig ender opp med å få en ikke spesielt ønsket person til bords.

- a) Formuler problemet PAR som et Lineær-Programmerings-problem (LP-problem). Du må klart definere de variable, målfunksjonen og kravene.

(15 poeng)

- b) Ofte er det slik at spesielle LP-problemer kan løses på en spesiell måte. Hvordan vil du så effektivt som mulig løse PAR-problemet i praksis? (Husk at N kan være vilkårlig stor.)

(10 poeng)

Oppgave 6 (10%)

Anta at GLAGNAR er en type datastruktur (en klasse) med 4 attributter/felter, *buncha*, *muncha*, *cruncha* og *human*. Anta at disse feltene er udefinerte idet objektet opprettes. Anta at *zoid* er en minimums-heap av GLAGNAR-instanser, der GLAGNAR-instansene sammenlignes basert på sine *muncha*-attributter. Du er usikker på hva *buncha*-feltene er satt til for instansene i *zoid*, men du vet at *muncha*-feltene er satt til ulike tall. *cruncha* og *human*-attributtene for disse instansene er udefinerte. Betrakt følgende pseudokode:

```
while zoid.length > 1
  pop a fra zoid
  pop b fra zoid
  rinds ← en ny GLAGNAR-instans
  rinds.muncha ← a.muncha + b.muncha
  rinds.cruncha ← a
  rinds.human ← b
  sett (dvs. push) rinds inn i zoid
```

a) Hva gjør koden?

(10 poeng)