SIF8010 AlgDat, 16.12.2002 || **SVARARK#1/3||** **Stud. nr.:**

**En besvarelse kan gi inntil 100 poeng, disse er påført.**

**<u>Oppgave 1 (15%)</u> Fyll her inn kun de verdier som endrer seg fra foregående matrise**

Til (Startsituasjon)

| Fra \ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 2 | 9 | ∞ | -3 |
| 2 | ∞ | 0 | ∞ | 1 | 7 |
| 3 | ∞ | 4 | 0 | ∞ | ∞ |
| 4 | -2 | ∞ | -5 | 0 | ∞ |
| 5 | ∞ | ∞ | ∞ | 5 | 0 |

Node 1 tas med

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | 0 | | | -5 |
| 5 | | | | | |

Node 1-2 tas med

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | 3 | |
| 2 | | | | | |
| 3 | | | | 5 | 11 |
| 4 | | | | | |
| 5 | | | | | |

Node 1-3 tas med

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | -1 | | | |
| 5 | | | | | |

Node 1-4 tas med

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | | -2 | | |
| 2 | -1 | | -4 | | -4 |
| 3 | 3 | | | | 0 |
| 4 | | | | | |
| 5 | 3 | 4 | 0 | | |

Node 1-5 tas med

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | 1 | -3 | 2 | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |

**<u>Oppgave 2 (40%)</u>**

a) (10%) Algoritmeforslag:

3 alternativer, det dårligste først:
1) Sorter. Test: ”Er summen av de k minste tallene < T ” ? Kjøretid: O(n log n)
2) Hold heller de k minste verdiene i en Heap, løp gjennom alle verdiene. Test. Kjøretid: O(n log k)
3) Finn heller den k-te største verdien med rekursiv (median-av-median) Select. Test. Kjøretid: O(n)
Det gjelder her å ha foreslått best mulig alternativ. Å nevne ”knapsack” her er ikke bra.

b) (10%) Modifikasjon av Quicksort:

Bruk rekursiv (median-av-median) Select. Dette garanterer at vi får ”mange nok” verdier på hver side av pivot-elementet slik at O(n x n) unngås som et verste tilfelle. Oppgaven står i Læreboken.

SIF8010 AlgDat, 16.12.2002 || **SVARARK#2/3||** **Stud. nr.:**

## c) (10%) Algoritmeforslag:

Her er det bare å starte med en node v, farge v med Grønn, farge v's naboer Røde (motsatt farge) dersom det ikke oppstår en konflikt, og fortsette slik inntil en 1) får en konflikt, eller 2) til alle noder blir farget.
I tilfelle 1) har en oppdaget en løkke som inneholder et odde antall noder.
Kjøretidsanalyse: En ser på alle noder og kanter, i verste fall, O(V + E)

## d) (10%) Algoritmeforslag:

Algoritme A1: Risikerer (worst-case) å måtte løpe sekvensielt igjennom alle tallene.
I beste fall oppdages det ved første test av et tall-par at sekvensen ikke er av opp/ned-type.

Algoritme A2: Binær søking er det eneste fornuftige alternativ.
I beste tilfelle støter en på toppen på midten, og kan avslutte "i konstant tid".

Kjøretidsanalyse for A1 og A2: (Fyll ut parentesene)

| | Best case | Worst case |
|---|---|---|
| A1 | O(1)<br>θ(1 )<br>Ω( 1 ) | O(n)<br>θ(n)<br>Ω(n) |
| A2 | O(1)<br>θ(1)<br>Ω(1 ) | O(log n)<br>θ(log n)<br>Ω(log n ) |

Kommentarer: Her testes grunnleggende forståelse av tids-
kompleksitet på grunnleggende algoritmer.

| SIF8010 AlgDat, 16.12.2002 || **SVARARK#3/3||** **Stud. nr.:** |
|---|

## Oppgave 3 (15%)

a) Asymptotisk kjøretid: O(n/x + x/y + y)

b) Diskusjon av Hopp's ide:

Hopp bør uansett velge x og y slik at antall element (H) en i verste fall må undersøke blir minimalt. Om vi setter H(x,y) = n/x+x/y+y gir standard analyse at H er minimal dersom (x,y) = (n^(2/3), n^(1/3)). Gjenomsnittsverdien av H er enkelt H/2.
Hopp's metode er ikke god, han burde ha fortsatt å hoppe i sprang større enn 1 inntil det er få elementer igjen. Bare slik utnyttes det godt at A er sortert. Han burde altså utvide formelen til n/x+x/y+y/z+z/w+ ... + 1.
Standard analyse vil også her greitt gi oss de optimale x,y,z... . Et regelmessig mønster dukker fram, prøv !
Hopp er bare iferd med å gjenoppdage at binær søking er det beste en kan foreta seg når en hopper i en sortert sekvens. Gunnleggende informasjonsvitenskap sier at vi ikke kan gjøre det bedre enn å se på midt-elementet når vi søker i en sortert datamangde. Det dårligst utfallet er da nemlig at vi får halvvert restmengden det må søkes i, og bedre kan det – i gjennomsnittstilfellet – ikke gjøres.

## Oppgave 4 (15%)

A $\boxed{\text{slutt}_i > \text{maks}}$ B $\boxed{\text{maks=slutt}_i}$ C $\boxed{\text{teller++}}$ ;

"teller" vil ved utgangen av n-løkken inneholde den søkte verdien.
(En kan her selvsagt snu logikken og derved bytte boksene B og C.)

Algoritmens kjøretid: $\boxed{\text{Sorteringen dominerer lineær løkke, altså O(n log n)}}$

## Oppgave 5 (15%)

Forklarende figur. Før kun på det som er nødvendig for at ideen kan forklares.

Nettverket som skal "gis" til algoritme S bygges slik:
Bipartitt graf G,J med en kant fra enhver G-node til enhver J-node.
Disse kantene gis skranker [0,1] og kostnader $\{d_{gj}\}$. En sluknode S og en kildenode T koples hhv til G- og J-nodene. Alle disse kantene gis skranker [1,1] og kostnader 0. (spiller ingen rolle hvilken verdi disse får, - hvorfor ikke ?). T koples så med en tilbakekopling til S og gis kost =0 samt skranker [200,200]. Disse skrankene "stemmer overens" med [1,1]-skrankene og "tvinger" igjennom at 200 personer "flyter" gjennom arrangementet. Ut kommer F-verdiene, og det eneste problem som da gjenstår er å finne partneren i en stor mengde. Alle bør derfor gis et g/j-nummer på ryggen, så ikke maten blir kald.
En kan her "rasjonalisere bort" 201 kanter og 2 noder. Ser du hvordan?