UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i :	IN 115
Eksamensdag :	Lørdag 20 mai, 2000
Tid for eksamen :	09.00 - 15.00
Oppgavesettet er på :	5 sider
Vedlegg :	Intet.
Tillatte hjelpemidler :	Alle trykte og skrevne

Kontroller at oppgavesettet er komplett før du begynner å besvare det.
Les gjennom hele settet før du begynner med besvarelsen.

Oppgavesettet består av 3 oppgaver som kan løses uavhengig av hverandre. Oppgavene innen hver del er tenkt løst i den rekkefølge de står, men dette er selvsagt ikke noe krav til din besvarelse. Prosenten angitt på hver del antyder hvor mye vekt det vil bli lagt på denne delen under sensureringen. I oppgave 1-g står 'kan puffes' og det betyr at du i verste fall får 4.0 på dette punktet hvis du ikke besvarer det punktet. Programmer skal skrives i Java (men hvis du har fått godkjent obligatoriske oppgaver ett tidligere år, kan du også besvare oppgaven i Simula, selv om vi anbefaler at du også da nytter Java). Du behøver ikke gi noen fullstendig dokumentasjon av programmene, men du kan skrive noen få linjer som gir leseren nøkkelen til forståelse av programmet. Du kan anta at leseren kjenner problemstillingen i oppgaven meget godt. Alle steder der det er spørsmål etter et program (eller en programbit) skal du skrive dette helt ut, og ikke bare henvise til liknende programmer, f.eks. i læreboka eller i et bibliotek.

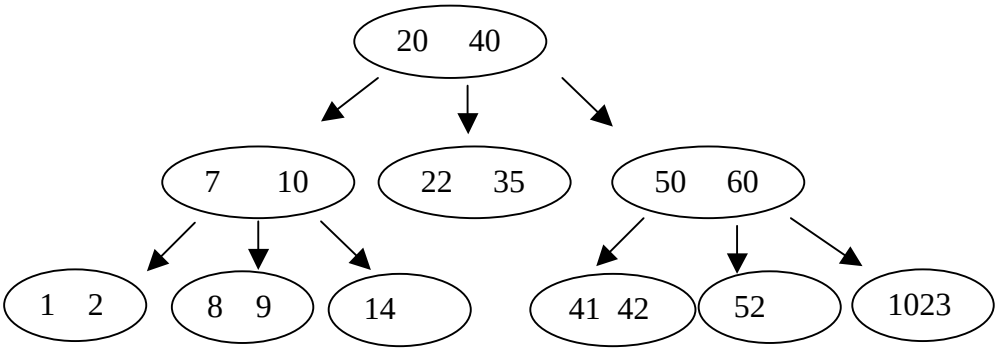
Les oppgavene nøye, og lykke til! *Arne Maus og Almira Karabeg*

Oppgave 1 (40 %)

I denne oppgaven skal vi se på trinære trær. I et trinært tre har hver node en eller to verdier, v1 og eventuelt v2. Hvis noden har to verdier, er v1 < v2. Noden har tre pekere, vsub, msub og hsub som peker på hvert sitt subtre til noden. Subtreet som pekes på av vsub har verdier < v1. Enhver verdi 'v' i subtreet som pekes ut av msub , er alle slik at v1 < v < v2. Subtreet som pekes ut av hsub har verdier > v2. Vi antar at bare ulike verdier er representert i treet.

Alle interne noder – dvs. alle noder med minst ett ikketomt subtre, har to verdier. Det er bladnoder – dvs noder med bare tomme subtrær, som kan ha bare én verdi. Eksempel på et trinært tre:

(fortsettes neste side)



For å implementere et trinærtre, definerer vi klassen TriNode og interfacen TritreI:

```
class TriNode{
      public int v1, v2;
      public Boolean v2Filled;   // true hvis v2 har en verdi
      public TriNode vsub, msub, hsub;
      TriNode (int v)
      { v1 = v; v2Filled = false; }
      public void settV2 (int val)
      { … se oppgave a)….;}

}

interface TriTreI {
      TriNode rota ();
      void settInn (int val);
      TriNode finn (int val);
      boolean fjern (int val);
      void skrivSortert ();
}
```

Oppgave 1a) Skriv metoden **`settV2(int v)`** i TriNode som gjør alt som er nødvendig for å sette den andre verdien v2 i en bladnode; Forklar hvilke forutsetninger du tenker er oppfylt når denne metoden kalles.

Vi skal nå implementerer interfacen TriTreI i en class TriTre:

```
class  TriTre implements TriTreI{
      protected TriNode rot;
      public TriNode rota () { return rot;}
      TriTre (int val)
      { rot = new TriNode(val);}
      ……
}
```

Oppgave 1b) Skriv metoden **`settInn (int val)`**. Husk at denne og noen av de andre metodene lettest implementeres rekursivt, slik at det kan hende at du her også vil lage en settInn metode med flere parametre.

(fortsettes neste side)

Oppgave 1c) Skriv metoden **`finn(int val);`**

Oppgave 1d) Skriv metoden **`skrivSortert();`** som traverserer treet og ved hjelp av System.out.println(..); skriver ut verdiene i stigende sortert rekkefølge,

Oppgave 1e) Hva blir høyden på et velballansert trinærtre med n noder?

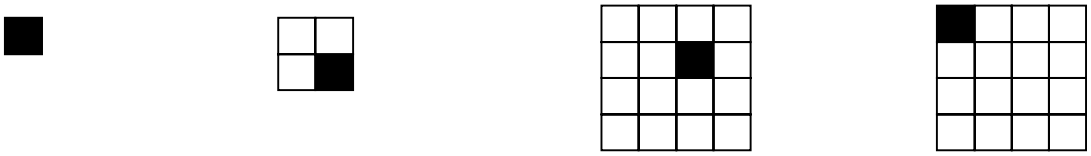
Oppgave 1f) Hvordan ville du implementere et trinærtre med verdiene representert i en heltalls-array 'a'? Skriv ikke kode, men bare angi hvor du ville plassere verdiene og subtrærne.

Oppgave 1g) *Kan puffes.* Skriv metoden **`fjern (int val`**). Her blir det mange spesialtilfeller, og du kan se bort fra spesialtilfellet at alle verdiene i treet fjernes. Hvis du ikke har tid til å skrive kode, får du 2.5 i karakter hvis du korrekt greier å skissere alternativene for hvilken verdi som evt. skal 'skyves' opp i treet og skal erstatte den verdien som fjernes, samt logikken når du skal fjerne en node.

Oppgave 2 (45 %)

I denne delen av eksamen skal vi bruke to av de grunnleggende teknikkene vi har til å lage algoritmer. Disse skal vi bruke til å løse problemet med å 'flislegge'(dvs dekke) et defekt sjekkbrett med fargete 'triminoer' (figur 2), og vi skal analysere algoritmene.

Et defekt sjakkbrett er et 2^k x 2^k brett av kvadratiske ruter med eksakt én defekt rute. Defekt betyr at vi ikke kan bruke denne ruten og den er farget sort. Fig. 1 viser noen defekte sjakkbrett. Det er opplagt at for k = 0 er det bare en rute, men for enhver annen verdi av k, er det eksakt 2^{2k} ulike defekte sjekkbrett:



k = 0 k = 1 k = 2 k = 2

Figur 1.

I det defekte sjekkbrettproblemet, blir vi bedt om å 'flislegge' (dvs. dekke til) sjakkbrettet med triminoer. Triminoer og de måter de kan orienteres på, er vist i Figur 2. Hvert trimino (alle tre rutene det består av) er farget og fargene nummereres 1,2,3,...osv . I en flislegging kan triminoene ikke ligge oppå hverandre, og de må dekke hele sjekkbrettet med unntak av den defekte ruten. To triminoer som har felles grense, må ikke ha samme farge (men hvis de bare ligger hjørne mot hjørne, kan de godt ha samme farge).

(fortsettes neste side)

Figur 2.

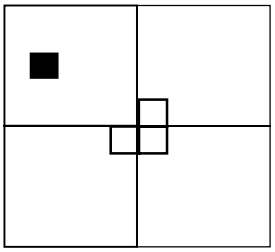
Et defekt sjakkbrett med k = 0, er trivielt flislagt, fordi det ikke har noen ikke-defekte ruter. Hvis k = 1 er det eksakt 3 ikke-defekte ruter på sjakkbrettet, og dette kan da opplagt flislegges med en av triminoene fra figur 2, og det er da også opplagt at det bare brukes en farge.

Oppgave 2a)

1. Hvor mange triomoner trengs for å fullstendig flislegge et defekt sjakkbrett når k > 1 ? Kall dette tallet N.
2. Det viser seg at N alltid et heltall. Hvis du skulle vise det formelt, hvilken bevisteknikk ville du da bruke ?

Oppgave 2b) Vi skal i denne oppgaven drøfte hvilke kjente teknikker som er best egnet til å både flislegge et defekt sjakkbrett med triminoer og få farger på disse slik at triminoener som ligger inntil hverandre, har ulike farger.

1. Se først på flislegginga og se på figur 3 for å få en idé om hvordan man legger den første triminoen. Ut fra dette, forslå en kjent teknikk til å foreta flisleggingen. Gi navnet på teknikken og skissert hvordan den ville bli anvendt på dette spesielle problemet.
2. Vi skal nå se på fargeleggingen av de triminoene du brukte til å flislegge i pkt. 1. Det finnes en berømt setning som sier at man maksimalt trenger 4 farger til en slik fargelegging, men vi skal her ikke nødvendigvis prøve å oppnå denne optimale løsningen, men igjen bruke en fundamental teknikk fra kurset som vil prøve å bruke så få farger som mulig. Gi navnet på denne teknikken og skisser hvordan du ville bruke den på dette spesielle problemet.



2^k

2^k

Figur 3.

(fortsettes neste side)

Oppgave 2c) Du skal nå skrive en Javametode **`tileAndColourBoard(..)`** for å flislegge og fargelegge et slikt defekt sjakkbrett.
Metoden er inne i en klasse 'ChessBoard' som har to int arrayer 'board []' og 'tileColour []'. Arrayen 'board' representerer de 2^k x 2^k rutene på sjakkbrettet og board[0][0] er øvre, venstre rute på brettet. Vi skal nummerere de N stk. triminoene vi bruker til flisleggingen (jfr. oppgave 2a) fra 1 og opp til og med N, og når du er ferdig med flislegginga, skal board[i][j] inneholde nummeret til den triminoen som dekker rute(i,j) på sjakkbrettet.
Likeledes skal 'tileColour[i]' etter fargeleggingen inneholde nummeret på fargen til den 'i'te triminoen du bruker i flisleggingen. Vi nummererer de fargene vi bruker fra 1 og oppover. I tillegg kan det være det nyttig at klassen inneholder et heltall 'nextTile' som inneholder nummeret til neste trimino du vil bruke i flislegginga.

Parametre er :

```
public void tileAndColourBoard
 (int topRow, int topCol, int defectRow, int defectCol, int size)
```

og det første kallet er:

```
tileAndColourBoard (0,0, dr, dc, size)
```

hvor size = 2^k og dr og dc er indeksene til den defekte ruten.

Skriv ut det antall farger du trengte for å fargelegge alle timinoene på brettet.

Oppgave 2d) Analyser kompleksiteten til din algoritme som funksjon av 'k'. Bruk stor O() eller Theta-notasjon θ () i svaret.

Oppgave 3 (15 %)

I denne oppgaven skal vi se på Huffman-koding av symboler.

Oppgave 3a) Betrakt teksten ***isismispi.*** Konstruer Huffmantreet for de fire symbolene : i, m, p, s. Når du lager treet, start med et tre med ett symbol i hver bladnode og symbolene i alfabetisk rekkefølge fra venstre mot høyre. Bestem Huffman-koder for symbolene.

Oppgave 3b) For at det skal være mulig å dekode og kode med bruk av variabel lengde på koden for symbolene, må man lagre en tabell over symbolene og deres korresponderende koder. Til å begynne med skal du bruke koden du fant i oppgave 3a til å dekode stringen: 1000110111101010. Gi så en pseudokode for dekodings-algoritmen. Forklar hvorfor den alltid vil virke.

(fortsettes neste side)