HØGSKOLEN I SØR-TRØNDELAG
Avdeling for informatikk og e-læring

Målform:	Bokmål					
Eksamensdato:	1. desember 2015					
Varighet/eksamenstid:	5 timer					
Emnekode:	TDAT2005					
Emnenavn:	Algoritmer og datastrukturer					
Klasse(r):	2ING					
Studiepoeng:	10					
Faglærer(e): (navn og telefonnr på eksamensdagen)	Helge Hafting, tlf 73559544 Mildrid Ljosland, tlf 73559556/93080942					
Kontaktperson(adm.) (fylles ut ved behov – kun ved kursemner)						
Hjelpemidler:	Ett stemplet A4-ark med valgfritt innhold					
Oppgavesettet består av: (antall oppgaver og antall sider inkl. forside)	7 oppgaver og 6 sider inkludert forside og 2 vedlegg.					
Vedlegg består av: (antall sider)	2 sider					

Merknad:

Oppgaveteksten kan beholdes av studenter som sitter eksamenstiden ut.

NB! Les gjennom hele oppgavesettet før du begynner arbeidet, og disponer tiden.

Dersom noe virker uklart i oppgavsettet, skal du gjøre dine egne antagelser og forklare dette i besvarelsen.

Lykke til!

Oppgave 1 (30%)

Søkemotorselskapet «BedreEnnGoogle» har K (for tiden 100, men det øker jevnt og trutt) maskiner som samler inn websider fra hver sine områder/domener, og produserer søketreff fra de websidene de har samlet inn. Hver av de K maskinene sender sine beste treff videre til en hovedmaskin som plukker ut de N høyest rangerte treffene og sender dem videre til kunden. For tiden er N=1000, men man vurderer å øke det.

Per i dag sender hver av de K maskinene N treff til hovedmaskinen, slik at hovedmaskinen plukker ut de N beste blant K*N forslag. Men det betyr jo at (N-1)*K treff egentlig er bortkastet, så det er kommet et forslag om at hver maskin bare skal produsere R treff, der $R \geq N/K$ og $R < N$. (R må selvfølgelig være et heltall, så N/K avrundes til minste heltall som er større eller lik N/K.) Man er imidlertid usikker på hvor stor R må være for at man skal være rimelig sikker på å få de N beste totalt.

Eksempel (K=2 og N=10): Gitt følgende rangeringer i sortert rekkefølge:
Maskin 1: 53, 39, 25, 13, 11, 9, 5, 3, 2, 1
Maskin 2: 36, 31, 28, 24, 22, 19, 17, 14, 10, 8
Da bør treffene 53, 39, 36, 31, 28, 25, 24, 22, 19, 17 leveres. Men hvis hver maskin bare leverer R=10/2=5 treff, vil hovedmaskinen levere treffene 53, 39, 36, 31, 28, 25, 24, 22, 13, 11. Altså blir de to siste feil. Hvis R=6 vil resultatet bli 53, 39, 36, 31, 28, 25, 24, 22, 19, 13 som også er feil, mens hvis R=7 (eller større) får vi riktig resultat.

Du jobber i «BedreEnnGoogle» og har fått i oppdrag å finne ut hvor stor R bør være. Du bestemmer deg for å lage et program som simulerer dette og på bakgrunn av simuleringsresultatene finner sannsynligheten for at de N beste treffene er blitt med for ulike verdier av R.

Du finner ut at programmet ditt må gjøre følgende:

A. Gjenta 1000 ganger:
    1. Produser K tallrekker, hver med N tall, ved hjelp av en random-generator.
    2. Fra de K*N tallene, finn de N største. Dette er fasiten.
    3. Gjenta for 10 forskjellige R-verdier mellom N/K og N:
        Finn ut om fasiten oppnås hvis vi tar med R treff fra hver maskin. Hvis ja: Øk antallOK for denne R-verdien med 1.
B. For hver av de 10 antallOK-ene, del på 1000, så har du sannsynligheten for å presentere alle de N beste for denne R-verdien.

I alle spørsmålene om kompleksitet i oppgavene under, skal du gi svaret i O-notasjon uttrykt ved N og K. Du behøver ikke å tenke på Ω og Θ. Husk begrunnelse! Du kan anta at $N \geq K$.

a) Hva er kompleksiteten av punkt 1, altså å produsere K tallrekker med N tall i hver? Hva blir kompleksiteten når du skal gjenta dette 1000 ganger?

Heretter kan du se bort fra at det skal gjentas 1000 ganger, og bare se på ett gjennomløp av løkka A.

b) Hvis du starter punkt 2 med å sortere de K tallrekkene, hva blir kompleksiteten av dette?
c) Når du har de K sorterte tallrekkene, hvordan vil du finne de N største tallene? Hvis du kommer på mer enn en måte å gjøre det på, velg den mest effektive.
d) Hva blir kompleksiteten av den metoden du har beskrevet i c? Av punkt 2 totalt?

Punkt 3 kan gjøres på følgende måte hvis man i punkt 2 holder rede på hvilken tallrekke de ulike tallene stammer fra:
Tell opp antall tall i fasiten som kommer fra de ulike tallrekkene. Finn maksimum av disse antallene, kall det M. For hver av de 10 R-verdiene: Hvis $M \leq R$, øk antallOK med 1.

e) Hva blir kompleksiteten av punkt 3? Og hva blir kompleksiteten av 1, 2 og 3 totalt? Av hele algoritmen, inkludert 1000 gjennomløp av løkka i A samt punkt B?

Nå skal vi gå over til å se på hvordan du kan gjøre punkt 2 hvis du ikke starter med å sortere de K tallrekkene.

f) Punkt 2 kan utføres ved å legge alle de K*N tallene i en heap og hente ut de N største. Hva blir kompleksiteten av dette?

Algoritmen quickselect er en algoritme som har samme tankegang som quicksort, men i stedet for å sortere tallene, løser den problemet: Blant n tall, finn de k minste (og dermed også de n-k største). (NB: n og k her betyr ikke det samme som N og K ellers i oppgaven.)
Den deler inn i "små" og "store" tall akkurat som quicksort, men kaller seg selv rekursivt bare på den ene delen som inneholder delingspunktet mellom de k minste og de n-k største. Dermed finner den etterhvert skillepunktet, og ender opp med de k minste på den ene siden og de n-k største på den andre siden (men vanligvis ikke i sortert orden).

Algoritmen er slik:

```
public static void quickselect(int[] t, int v, int h, int k) {
   if (h-v>2) {
      int delepos = splitt(t, v, h);
      if (delepos  > k) quickselect(t, v, delepos-1, k);
      else if (delepos < k) quickselect(t, delepos+1, h, k);
      else return;
   }
   else median3sort(t,v,h);
}
```

Metodene splitt og median3sort er som i læreboka:

```
public static int median3sort(int[] t, int v, int h) {
   int m = (v+h)/2;
   if (t[v]>t[m]) bytt(t,v,m);
   if (t[m] > t[h]) {
      bytt(t, m, h);
      if (t[v] > t[m]) bytt(t, v, m);
   }
   return m;
}
public static int splitt(int[] t, int v, int h) {
   int iv, ih;
   int m = median3sort(t, v, h);
   int dv = t[m];
   bytt(t,m, h-1);
   for (iv=v,ih=h-1;;) {
      while(t[++iv]<dv);
      while(t[--ih]>dv);
      if (iv>=ih) break;
      bytt(t,iv,ih);
   }
   bytt(t,iv,h-1);
   return iv;
}
```

g) Demonstrer hvordan dette vil fungere hvis du skal finne de 5 minste (og de 5 største) i tabellen 25, 13, 5, 31, 19, 8, 12, 10, 21, 4.

h) Vis at quickselect har kompleksitet O(n)

i) Vi kan utføre punkt 2 ved å lage en lang tabell av alle de K tabellene og utføre quickselect på den. Hva blir nå kompleksiteten av punkt 2?

Oppgave 2 (15%)

a) Forklar hva leksikografisk ordning er.

`Fra https://no.wikipedia.org/wiki/Eliteserien_i_fotball_for_menn`
henter vi følgende beskrivelse:
Lagene rangeres etter antall poeng (tre for seier, ett for uavgjort, null for tap) ved sesongslutt. Hvis to eller flere lag ender med samme poengsum, blir målforskjellen (antall scorede mål minus antall sluppet inn) tellende, deretter antall scorede mål, og til slutt lagenes innbyrdes resultater.

b) I vedlegg 1 er gitt klassen «Fotball». Lag en klasse «Lag» slik at setningen `Collections.sort(lagene);` i «Fotball» sortere lagene i henhold til reglene gitt over, bortsett fra at du ikke skal ta hensyn til innbyrdes resultater (siste regel). Du skal bruke leksikografisk ordning for å få til dette.

Oppgave 3 (15%)

Se grafen i vedlegg 2.

a) Finn maksimum flyt gjennom grafen, ved hjelp av flytøkende veier. Oppgi hvilke flytøkende veier du brukte, og hvor mye flyt du fikk lagt til med hver av dem.
b) Sorter grafen topologisk, eller forklar hvorfor dette ikke er mulig.
c) Se bort fra retningen på kantene i denne deloppgaven. Finn et minimalt spenntre for grafen.
d) Tegn inn en ny kant fra S til C. Finn de sterkt sammenhengende komponentene i denne nye grafen.

Oppgave 4 (10%)

a) Fortell kort om hvordan Lempel-Ziv komprimering fungerer.
b) Demonstrer Lempel-Ziv kompresjon på «fort, fortere, fortest»

Oppgave 5 (10%)

a) Er det mulig å utvikle en algoritme som sorterer heltall asymptotisk raskere enn O(n)? Begrunn svaret.
b) I øvingsopplegget så vi på en blandet sorteringsalgoritme, quicksort med innsettingssortering som «hjelpealgoritme». Hva oppnår vi med en slik hjelpealgoritme?

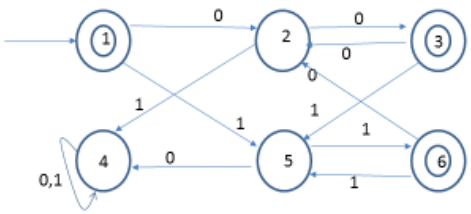
Oppgave 6 (10%)

Gitt D = {2,3,4,5,6,7,8,9,10,11,12}.

a) La relasjonen R være definert på D ved at (a,b)ϵR hvis og bare hvis (a mod 5)=(b mod 5). Vis at R er en ekvivalensrelasjon.
b) La relasjonen S være definert på D ved at (a,b) ϵS hvis og bare hvis a mod b=0. Er S en ekvivalensrelasjon? En partiell ordning? Husk begrunnelse.

Oppgave 7 (10%)

Gitt automaten A vist nedenfor.



a) Skriv opp alfabet, starttilstand, aksepterende tilstander og neste-tilstand-funksjonen for A.
b) La L være språket definert av A og la L_i være mengden av de strengene i L som har lengde i. Skriv opp L_0, L_1, L_2, L_3 og L_4. Lag et regulært uttrykk for L.

Vedlegg 1

```
public class Fotball {
   ArrayList<Lag> lagene = new ArrayList<Lag>();

   public static void main(String[] args) {
      Fotball f = new Fotball();
      f.test();
   }

   public void test() {
      lagene.add(new Lag("LagA"));
      lagene.add(new Lag("LagB"));
      lagene.add(new Lag("LagC"));
      lagene.add(new Lag("LagD"));
      lagene.add(new Lag("LagE"));
      lagene.add(new Lag("LagF"));
      for (int i = 0; i < lagene.size(); i++) {
       for (int j = 0; j < lagene.size(); j++) {
          if (i != j) {
             int a = (int)(Math.random()*5);
             int b = (int)(Math.random()*5);
             registerKamp(lagene.get(i), lagene.get(j), a, b);
          }
       }
      }
      Collections.sort(lagene);
      skrivResultat();
   }

   public void registerKamp(Lag a, Lag b, int scoreA, int scoreB) {
      a.register(scoreA, scoreB);
      b.register(scoreB, scoreA);
   }

   public void skrivResultat() {
     System.out.println("Navn Poeng Scoret-Imot");
     for (int i = 0; i < lagene.size(); i++) {
          System.out.println(lagene.get(i));
      }
   }

}
```

Eksempler på utskrift:

```
Navn Poeng Scoret-Imot
LagB:     20          26-19
LagC:     14          21-21
LagD:     14          20-23
LagE:     12          19-17
LagF:     11          23-26
LagA:     11          15-18

Navn Poeng Scoret-Imot
LagE:     16          19-15
LagA:     16          15-12
LagF:     13          21-21
LagC:     13          22-23
LagD:     12          23-25
LagB:     12          20-24
```

Vedlegg 2

Graf for oppgave 3