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

Målform:	Bokmål
Eksamensdato:	3 desember 2013
Varighet/eksamenstid:	5 timer
Emnekode:	TDAT2005
Emnenavn:	Algoritmer og datastrukturer Løsning
Klasse:	2ING
Studiepoeng:	10
Faglærere:	Helge Hafting ☎ 73 55 95 44 Mildrid Ljosland ☎ 73 55 95 56
Kontaktperson(adm.)	
Hjelpemidler:	Ett stemplet A4-ark med valgfritt innhold
Oppgavesettet består av:	8 oppgaver og 9 sider inkludert forside
Vedlegg består av:	Ingen sider

Merknad:
Oppgavesettet 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 oppgavesettet, skal du gjøre dine egne antagelser og forklare dette i besvarelsen.
Lykke til!

Oppgave 1

Finn kjøretiden for de følgende programmene. Bruk Θ-notasjon om mulig, O og Ω ellers.

a)
```
public void a(int n, int [][]tab, int m) {
  for (int i=n; i>0; --i) {
    for (int j=0; j<m; ++j) {
      for (int k=j; k<m; ++k) {
        tab[i][k] += tab[k][j] * k;
      }
    }
  }
}
```

$\Theta\left(nm^2\right)$ Trippel løkke som avhenger av to variabler.

b)
```
public int b(int z, int []tab, int m) {
  if (m <= 0) return m*n;
  for (int i = 0; i < m; ++i) tab[i] += m*tab[n];
  int tmp = b(n/2+3, tab, m/4) + b(n/2+2, tab, m/4)
  return tmp + b(n/2, tab, m/4) + b(n/2+1, tab, m/4);
}
```

$\Theta\left(m \log m\right)$. Rekursiv metode med fire kall. Rekursjonen brytes når m blir 0, og m deles med 4 i hvert kall. En $\Theta\left(m\right)$-løkke inni den rekursive metoden. $T\left(m\right) = 4T\left(m/4\right)+m \Rightarrow T\left(m\right) \in \Theta\left(m \log m\right)$

c)
```
public int c(int n, int[]tab, int m) {
  int sum = 0;
  if (m<n) for (int i = 1; i < n; i += m) {
    sum += tab[i];
  }
  return sum;
}
```

$\Omega\left(1\right)$ (hvis m >= n), $O\left(n/m\right) \Rightarrow \Theta\left(n/m\right)$

d)
```
public void d(int z, int[][]tab, int m) {
  int halv=z/2;
  if (z == 0) return;
  for (int i=1; i<z; ++i) {
    for (int j=1; j<z; ++j) tab[i+m][j+m]++;
  }
  d(halv, tab, m+4);
  d(halv, tab, m);
  d(halv, tab, m-4);
  d(halv, tab, m-8);
}
```

$\Theta\left(z^2 \log z\right)$ Rekursiv metode med 4 kall. Rekursjonen brytes når z blir 0, og z halveres i hvert kall ved hjelp av variabelen `halv`. Inni metoden er det doble løkker som går fra 1 til z.

$T\left(z\right) = 4T\left(z/2\right) + cz^2 \Rightarrow T\left(z\right) \in \Theta\left(z^2 \log z\right)$

Oppgave 2

Gitt følgende hashfunksjoner: (m er tabellstørrelse, k er nøkkelen)

$h_a(k) = k \bmod m$

$h_b(k) = k \bmod (m-1) + 1$

$h_c(k) = (2|k| + 1) \bmod m$

a) Plukk ut to som egner seg for å implementere dobbel hashing. Begrunn valget.

b) Funksjonene stiller et krav til m. Hva er kravet? Hvorfor er det slik?

Det beste alternativet er a som hashfunksjon, og b for å velge hopplengde ved kollisjon. Dette fordi a sprer nøklene over hele tabellen, og b vil typisk gi ulike svar for nøkler som kolliderer i a-funksjonen. b produserer heller aldri 0.

c som hashfunksjon og b som hopplengde vil også fungere.

kravet til m er at m og hopplengdefunksjonen må være relativt prime. Når vi bruker b som hopplengdefunksjon, får vi til dette ved å la m være et primtall.

En annen løsning her, er å bruke a og c, og la m være en toerpotens. Da er imidlertid ikke a en god hashfunksjon, modulus-operasjonen kaster vekk mye av nøkkelen når m er en toerpotens.

Det viste seg at spørsmålet var litt løst formulert, noen tolket det slik at jeg spurte om «hvilke to funksjoner som egner seg som hopplengdefunksjon i dobbelt hashing.» I så fall må b og c velges, for a kan bli 0 og 0 er ikke en brukbar hopplengde. b vil kreve at m er et primtall, c vil kreve at m er en toerpotens.

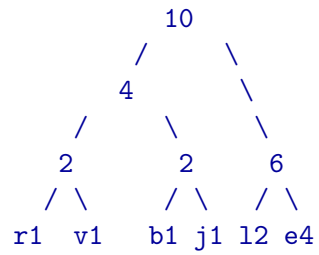
Oppgave 3

a) Tegn opp et huffmantre basert på bokstavene i ordet «REVEBJELLE»

Mange korrekte løsninger, ingen fasit. Frekvenstabellen blir slik:

Tegn	Frekvens
R	1
E	4
V	1
B	1
J	1
L	2

Det er til sammen 10 tegn. Ett av mange mulige trær:



b) Skriv binærkoden for ordet «REVELJE», basert på ditt huffmantre

Svaret vil variere med hvordan treet ser ut. For full uttelling må bitstrengen stemme med det treet kandidaten har laget. Med mitt tre, og 0 til venstre, 1 til høyre:

00011001111001111

c) Dekod «11111100000010» ved hjelp av huffmantreet ditt. (Se bort fra eventuelle bits som blir til overs til slutt.)

Svarene her avhenger helt av hvordan treet ser ut; ingen løsningsforslag er mulig. Uttelling i den grad dekodingen stemmer med treet.

Løsningsforslag med mitt tre: «EEERR»

Oppgave 4

Prioritetskøer kan implementeres som henholdsvis usortert tabell, heap og fibonacci-heap. Fortell om fordeler og ulemper ved hver av dem.

Usortert tabell

Fordeler Enkel å implementere. Prioritetsendring $O(1)$

Ulempe Å finne eller hente ut minimum tar $O(n)$ tid

heap

Kjapp, men noe mer komplisert enn usortert tabell. Å finne minimum tar $O(1)$ tid. Å hente ut minimum eller endre prioritet tar $O(\log(n))$ tid.

fibonacci-heap

Kjøretider som for heap, bortsett fra at prioritet endres i $O(1)$ tid. Ulempen er at strukturen er mer komplisert å programmere.

Oppgave 5

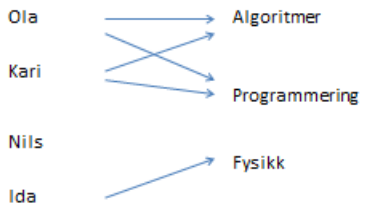
Vi definerer mengdene S og F ved S={Ola, Kari, Nils, Ida} og F={Algoritmer, Programmering, Fysikk}.

La relasjonen R være en delmengde av $S \times F$ definert ved at den viser sammenhengen mellom hvilke studenter som tar hvilke fag. Ola tar fagene Algoritmer og Programmering, Kari tar Algoritmer og Fysikk, Ida tar Fysikk, mens Nils ikke tar noen av disse fagene.

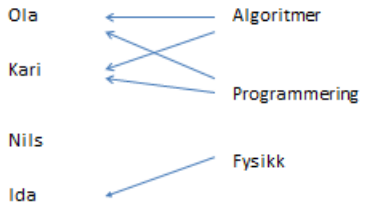
a) Tegn R og R^{-1}. Er noen av disse funksjoner? Hvorfor, eller hvorfor ikke?

Løsning:

R:



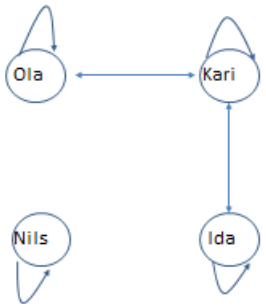
R^{-1}:



Ingen av dem er funksjoner. For at en relasjon skal være en funksjon, må alle elementer i definisjonsmengden (mengden S for R og mengden F for R^{-1}) ha en og bare en pil fra seg.

b) La videre relasjonen Q være definert på S ved at (xQy) hvis og bare hvis student x tar et fag som også student y tar. Tegn Q og avgjør om den er refleksiv, symmetrisk, antisymmetrisk og/eller transitiv.

Løsning:



Alle har samme fag som seg selv (også Nils som har ingen fag), så relasjonen er refleksiv. Hvis en person har samme fag som en annen, har den andre samme fag som den første, så relasjonen er symmetrisk. Man siden Ola har samme fag som Kari, og Kari har samme fag som Ida, men Ola ikke har samme fag som Ida, er relasjonen ikke transitiv.

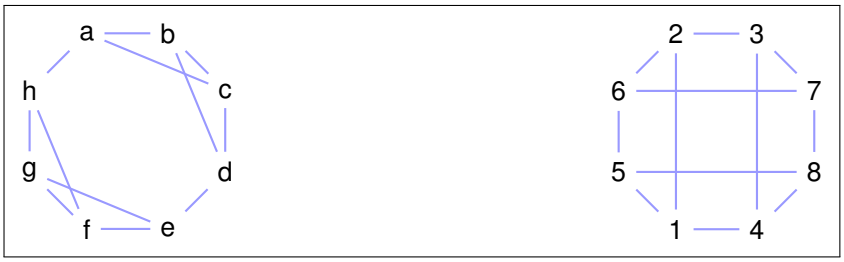
c) La A være mengden av alle mulige matriser. Definer relasjonen V på A ved at xVy hvis og bare hvis $y = x^T$ (dvs y er den matrisen vi får hvis vi bytter om rekker og kolonner i x). Er V en ekvivalensrelasjon?

En ekvivalensrelasjon er refleksiv, symmetrisk og transitiv. Siden det ikke er slik at $x^T = x$ for alle matriser, er relasjonen ikke refleksiv. Den er derfor ikke en ekvivalensfunksjon.

Oppgave 6

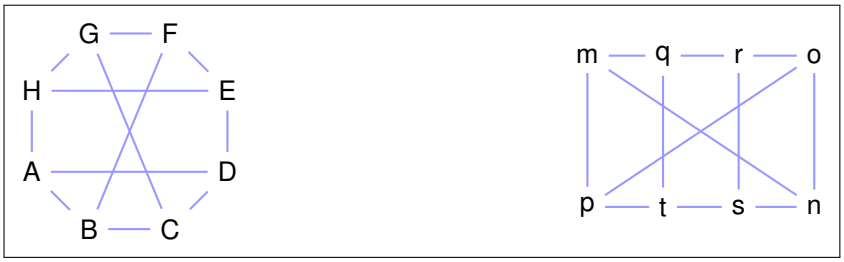
Finn ut om følgende grafer er parvis isomorfe. Hvis ja, gi en eksplisitt isomorfi, hvis nei, forklar hvorfor.

a)



Nei, ikke isomorf. I den første grafen har vi kretser av lengde 3, det har vi ikke i den andre. Der er korteste krets av lengde 4.

b)



Ja, isomorf. Det finnes mange muligheter, her er en av dem:

```
ABCDEFGH
ronstpmq
```

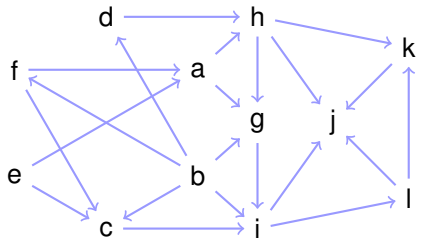
Oppgave 7

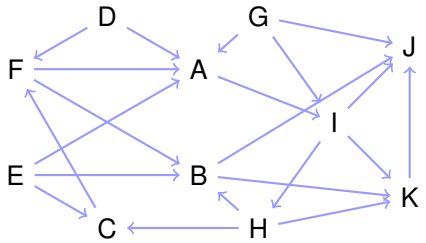
a) Fortell om forskjellene mellom A*-algoritmen og Dijkstras algoritme.

Dijkstras algoritme finner korteste vei gjennom grafen ved å se på hvordan grafen henger sammen. Dijkstra prioriterer kun ved hjelp av avstand til startnoden, og leter dermed i alle retninger.

A* er en utvidelse av Dijkstras algoritme. Den er i stand til å bruke hint om hvilken vei det er lurest å gå, f.eks. ved å se på hvilke noder som er nærmere målnoden. Dermed leter A* mer i riktig retning, og mindre i feil retning. Dermed søker A* gjennom et mindre område enn Dijkstra, før den finner den korteste veien.

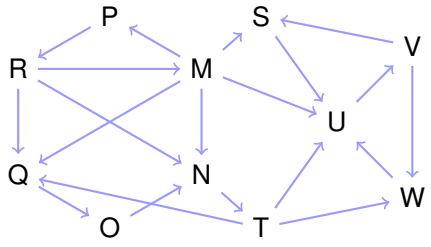
b) Finn en topologisk sortering for hver av de to grafene under, eller forklar hvorfor det ikke er mulig.





En mulig rekkefølge på den første: bfecadhgilkj. (Det fins mange flere, f.eks. ebfdahgcilkj) På den andre er det ikke mulig, fordi den har rundturen FAIHCF.

c) Finn de sterkt sammenhengende komponentene i grafen under:



PRM, QONT, SUVW.

d) Kan en graf ha flere ulike minimale spenntrær? Lag et eksempel, eller forklar hvorfor dette er umulig.

Det er mulig. Det enkleste eksempelet er en trekantgraf, hvor alle kantene har samme vekt. Her kan vi få tre ulike minimale spenntrær ved å fjerne én kant.

e) Dijkstras algoritme, Bellman Ford-algoritmen og Bredde-først søk er alle sammen algoritmer som løser ulike typer korteste-vei problem. Fortell om de ulike typene grafer de kan brukes på.

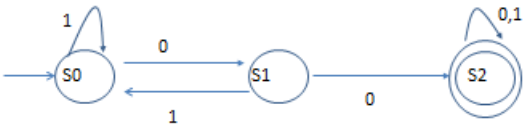
Bredde-først søket brukes på en uvektet graf. De andre to brukes på vektede grafer. Dijkstras algoritme forutsetter at kantene har positive vekter, Bellman-Ford algoritmen håndterer også grafer med negative vekter.

Oppgave 8

Gitt alfabetet 0,1.

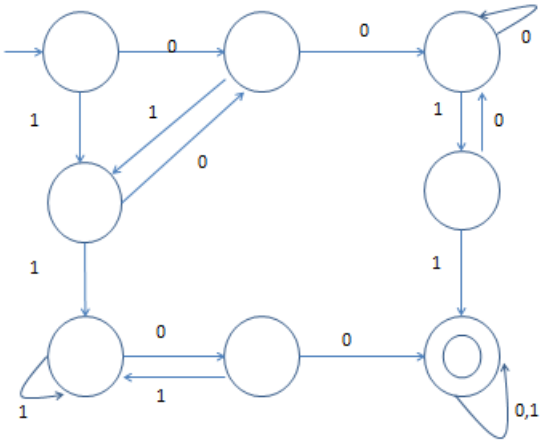
a) Lag en endelig automat som aksepterer alle bitstrenger som har minst én forekomst av bitsekvensen 00. Skriv opp et regulært uttrykk som genererer det samme språket.

Løsning:



(1*(01)*)*00(0|1)* eller (0|1)*00(0|1)*

b) Beskriv med ord (på liknende måte som automaten i a) er beskrevet) det språket som aksepteres av følgende automat:



Dette er en endelig automat som aksepterer alle bitstrenger som inneholder både 00 og 11, i vilkårlig rekkefølge.