Løsningsforslag eksamen i Algoritmer og datastrukturer desember 2014

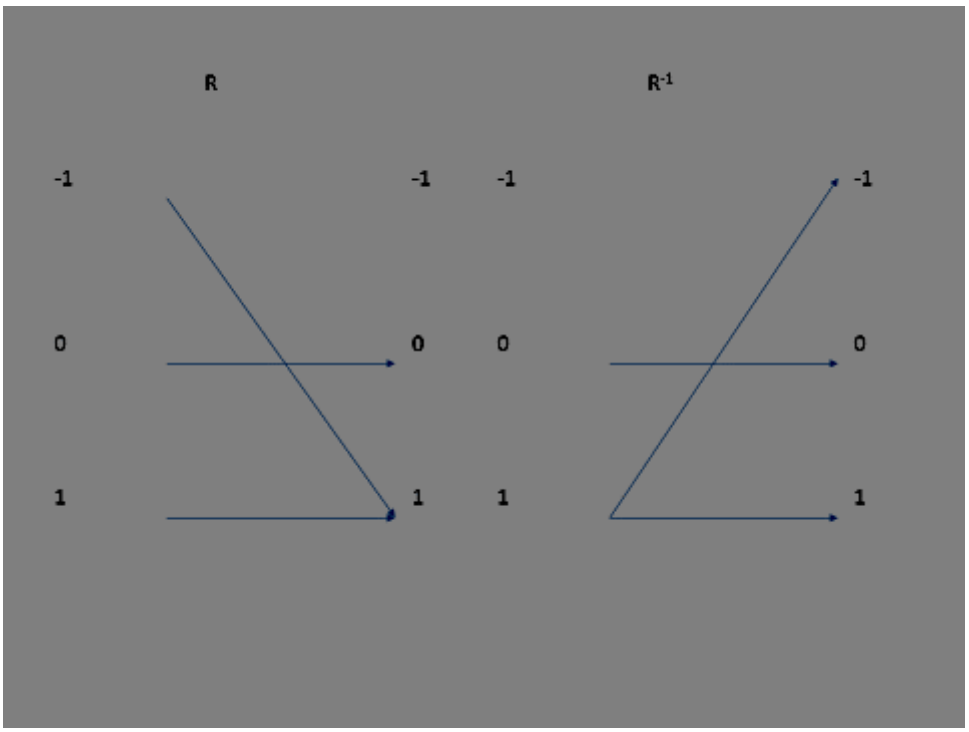
Oppgave 1

a)

Hvis m og n > 0 : Θ(mn)
Hvis m=0: Θ(n)
Hvis n=0: Θ(m)
Totalt: O(mn), Ω(n+m)
b)
Hvis m < n: Θ(m)
Hvis m>n: Θ(n)
Hvis m=n: Θ(1)
Totalt: O(m+n) eller O(minimum(m,n)), Ω(1)
c)
$T_m=2T_{m/2}+cm$
Θ(mlogm)

Oppgave 2

a)



R er en funksjon siden alle elementer i M har en pil fra seg.

R^{-1} er ikke en funksjon siden -1 ikke har noen pil fra seg (og 1 har to piler fra seg).

b) R er ikke refleksiv (mangler (-1,-1)), ikke symmetrisk (har (-1,1) men ikke (1, -1), transitiv ((-1,1) og (1,1), derfor (-1,1), antisymmetrisk ((-1,1) men ikke (1,-1))

c) Partiell ordning: Refleksiv, antisymmetrisk og transitiv. Trenger å lage den refleksiv. S={(-1,1), (0,0), (1,1), (-1,-1)}

d) Noe uheldig oppgaveformulering.Det står: Finn… i S. Det burde stått: Finn…. i M når du bruker S. Makimale elementer: 0 og 1. Minimale elementer: -1 og 0. Største element og minste element finnes ikke. (0 er inkompatibel med alle andre.)

Oppgave 3

a) Hvis to grafer er isomorfe, er følgende egenskaper like i de to grafene (enten har begge den, eller ingen av dem):
har n noder
har m kanter
har en node med grad k
har m noder med grad k
har en krets med lengde k
har en enkel krets med lengde k
har m enkle kretser med lengde k
er sammenhengende
har en Euler-krets
har en Hamilton-krets

b)

	i	ii	iii
Antall noder	6	6	6
Antall kanter	7	7	8
Antall enkle kretser med lengde 3	0	1	2
Antall enkle kretser med lengde 4	2	0	1

Siden de har ulikt antall kretser med lengde 3 (eller 4), kan de ikke være isomorfe.

c)

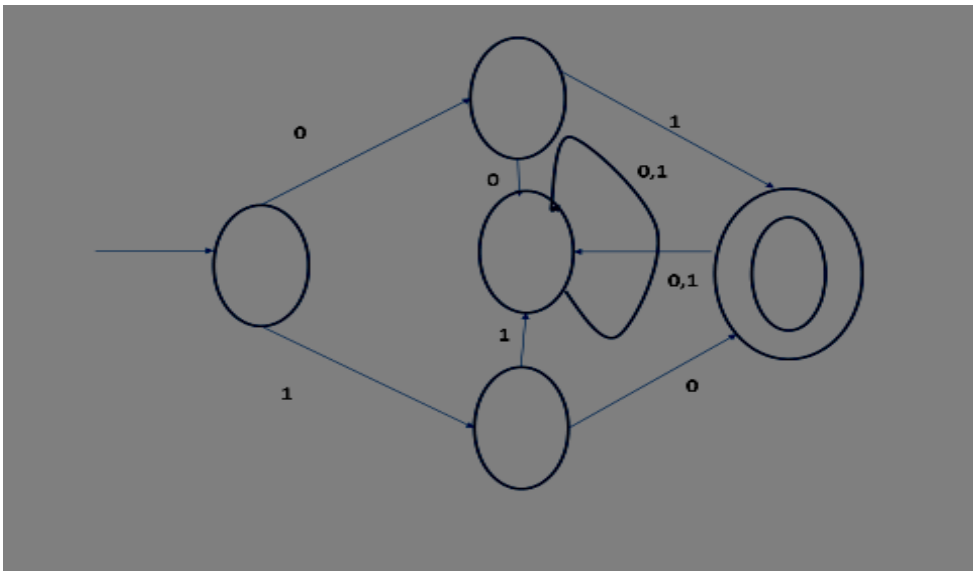
	a	b	c	d
Antall noder	6	6	6	6
Antall kanter	8	8	7	7
Antall enkle kretser med lengde 3	2	2	1	1
Antall enkle kretser med lengde 4	1	2	0	0

Ingen er isomorf med i.
c og d kan være (og er) isomorf med ii (og hverandre).
a kan være (og er) isomorf med iii.
b er ikke isomorf med noen.

Oppgave 4

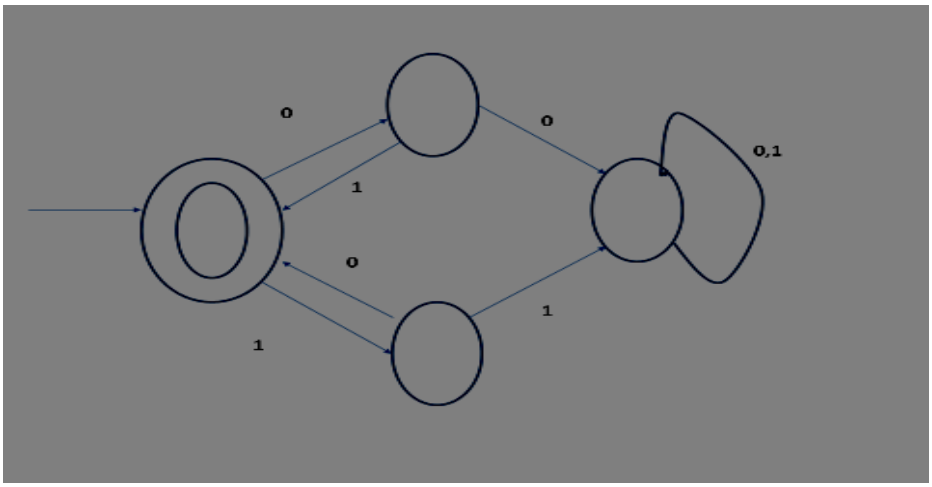
a) L_3={01, 10}

b)



c) Med: F.eks. den tomme strengen, 01, 10, 0110, 0101, 1010
Ikke med: F.eks. 00, 11, 000, 101

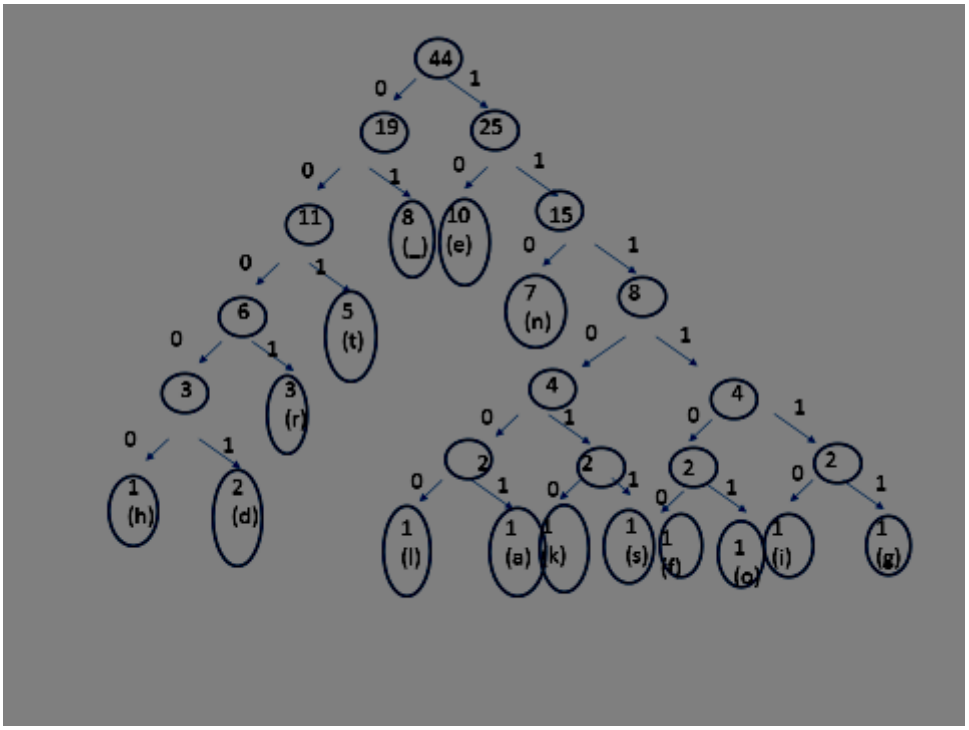
d)



Regulært uttrykk: (01|10)*

Oppgave 5

a)



(Ikke viktig at man har telt tegnene riktig. Det viktige er at huffman-treet er riktig i forhold til de verdiene man har funnet dvs. alltid kobler sammen de to minste. Det finnes mange riktige løsninger.)

b) 000011000100110011000010110110010000010111000001010
111001110110101100100110111010111011001011011011001000011011001
111100111101000100011111101111111 0

c) Med 15 ulike tegn trenger vi minst 4 bits per tegn hvis vi skal ha et konstant antall bits per tegn. Da trenger vi totalt 44*4=176 bits. Med huffman-koding bruker vi 8*6+(1+2)*5+3*4+(5+7)*3+(8+10)*2=147bits. Vi sparer altså 29 bits (16%).

d) Lavest mulig: Komplett binærtre med 15 løvnoder. Det krever høyde 4. Høyest mulig: Alle indre noder har (minst) ett barn som er en løvnode. 15 løvnoder gir høyde 14.

Oppgave 6

a)

P: Mengden av alle problemer som kan løses i polynomisk tid.

NP: Mengden av alle problemer der svaret kan sjekkes i polynomisk tid.

NP-komplett: Mengden av alle problemer som er i NP og er minst like vanskelig som et hvilket som helst NP-problem. Alle NP-problem kan omformes (reduseres) til et NPC-problem i polynomisk tid.

NP-hard: Mengden av alle problemer som er slik at det finnes et NPC-problem som kan omformes (reduseres) til dette problemet i polynomisk tid.

b) NP-komplett. Med n tall finnes det 2^n mulige delsummer som må sjekkes. Men når et svar er funnet, kan det sjekkes på O(n) tid.

Oppgave 7

a) Shellsort har kompleksitet et sted mellom O(n) og $O(n^2)$. I labforsøk har vi målt ca. $O(n^{1.1})$

b) Det er ikke mulig, fordi enhver generell sortering må se på alle elementer for å finne ut om de står rett. Minste kjøretid for sortering blir da O(n), som er mer enn kvadratroten av n.

c) Quicksort velger et av tallene som «pivot». Deretter gjøres en delvis sortering ved å flytte tall som er mindre enn pivot til venstre del av tabellen, mens de som er større enn pivot havner til høyre. Pivot settes på rett plass, mellom de to mengdene. Til slutt sorteres høyre og venstre del ved rekursive kall til quicksort. Rekursjon fortsetter til vi får del-tabeller med størrelse 1. (Et tall alene kan ikke stå i feil rekkefølge) Demo:

d) 3 1 2 5 8 6 9 7 (pivot 5)
1 2 3 5 6 7 9 8 (pivot 2 venstre og 7 høyre)
1 2 3 4 5 7 8 9 (Bare en deltabell igjen, pivot 9)

Oppgave 8

a) Dijkstras algoritme ser bare på avstand fra startnoden, når den velger hvilken node som skal behandles. Dermed sprer søket seg ut som en voksende sirkel, sentrert på startnoden.

A* ser heller på summen av avstand fra startnoden, og estimert avstand til målet når den velger neste node. (Vi må ha et estimat som aldri blir for høyt.) Et slikt estimat baseres gjerne på rettlinjet avstand fra noden til målet, og krever dermed informasjon om nodenes plassering på kartet. Dette spisser søket; det får form av en ellipse som peker mot målet. Ofte behandles færre noder på dette viset.

b) Her fins mange korrekte flytøkende veier, her er én mulighet:
KABS:40
KDES:60
KDCES:10
KACES:10
KABCES:10
Den totale flyten blir 130. Det fins andre sett med flytøkende veier, men alle riktige løsninger må ende med flyt på 130.

c) Vi kan tegne opp restnettet for grafen, og se at det ikke lenger fins noen vei fra kilde til sluk.