NTNU

Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i TDAT2005 Algoritmer og datastrukturer

Faglig kontakt under eksamen: Dag Olav Kjellemo og Helge Hafting

Tlf.: 920 50 951 og 924 386 56

Eksamensdato: 10. desember 2019

Eksamenstid (fra–til): 09:00–14:00

Hjelpemiddelkode/Tillatte hjelpemidler: Ett stemplet A4-ark med valgfritt innhold

Annen informasjon: Løsninger på oppgavene 1, 2 og 3 står samlet etter oppgave 3. Løsning for oppgavene fra 4 og utover, står med blått sammen med oppgavene.

Målform/språk: bokmål

Antall sider (uten forside): 10

Antall sider vedlegg: 1

Informasjon om trykking av eksamensoppgave			
Originalen er:			
1-sidig	☐	**2-sidig**	☒
sort/hvit	☐	**farger**	☒
skal ha flervalgskjema			☐

Kontrollert av

....

Dato Sign

Merk! Studenter finner sensur i Studentweb. Har du spørsmål om din sensur må du kontakte instituttet ditt. Eksamenskontoret vil ikke kunne svare på slike spørsmål.

Oppgave 1 — 8%

La X bestå av de positive heltallene som deler 12.

Relasjonen R på X som er definert ved at aRb dersom a deler b.

a) Er relasjonen
   - Refleksiv?
   - Symmetrisk?
   - Antisymmetrisk?
   - Transitiv?

b) Tegn Hasse-diagrammet til R.

c) Beskriv eventuelle maksimale, minimale, største eller minste elementer.

d) Beskriv en topologisk sortering av X ordnet etter relasjonen R.

Oppgave 2 — 12%

Det regulære språket L er definert over alfabetet $\Sigma = \{x, 0, 1\}$ ved det regulære uttrykket $((xx)^*0 \mid x(xx)^*1)^*$.

a) Lag tre forskjellige strenger som er i L.

b) Beskriv med ord språket L.

c) Lag en endelig automat som aksepterer språket L.
(Hint: Det kan gjøres med fire tilstander.)

Språket M er definert ved den kontekstfrie grammatikken

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow aSb \mid ab \mid \varepsilon \\ B &\rightarrow bA \end{aligned}$$

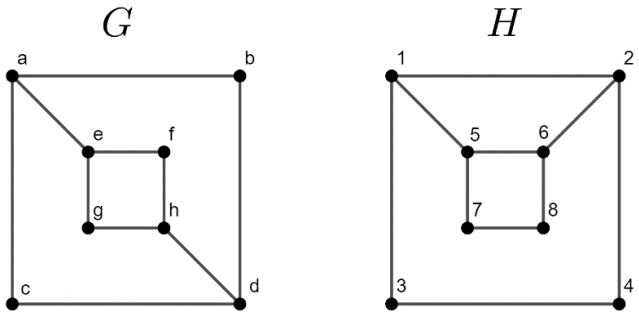
($S \rightarrow A \mid B$ er en skrivemåte for å skrive to regler, $S \rightarrow A$ og $S \rightarrow B$, på én linje.)

d) En av de følgende strenger er med i M. Angi hvilken, sammen med en produksjon av strengen:
   1. $aaba$
   2. $abaab$
   3. $baabb$

e) Vis at det ikke finnes noen endelig automat som aksepterer M.

Oppgave 3 **10%**

To grafer, G og H, er beskrevet ved diagrammene under:



a) Finn en Eulersti eller en Eulerkrets i hver av grafene G og H, eller begrunn at det ikke finnes.

b) Finn en Hamiltonkrets i hver av grafene, eller begrunn at det ikke finnes.

c) Gi en isomorfi mellom G og H, eller begrunn at grafene ikke er isomorfe.

Løsningsforslag Eksamen Høst 2019 TDAT2005 DM

(Poeng angitt i parantes er veiledende. En helhetsvurdering av forståelsen av hver oppgave må gjøres.)

Oppgave 1

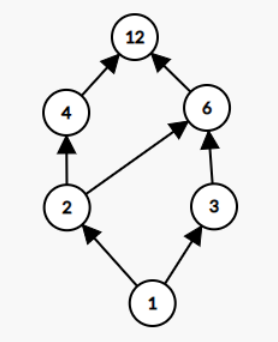
$X = \{1,2,3,4,6,12\}$, $aRb \Leftrightarrow a \mid b$

a) (2 poeng) Definisjonen $a \mid b$ er at det finnes heltall k slik at $b = k \cdot a$.

- **Refleksiv** fordi $a \mid a$, $(a = 1 \cdot a)$
- **Ikke symmetrisk**, fordi $1 \mid 2$, mens $\neg\ (2 \mid 1)$
- **Antisymmetrisk** fordi $a \mid b$ og $b \mid a$ impliserer $a = b$ når a og b er positive.
- **Transitiv** fordi $a \mid b$ og $b \mid c$ impliserer at $a \mid c$.
  $(b = k_1 \cdot a \ \wedge \ c = k_2 \cdot b \implies c = k_2(k_1 a) = (k_2 k_1) \cdot a\ )$

  Halvt poeng for hvert punkt. Ikke trekk hvis $X = \{12,24, \dots\}$ men de har svart på ellers rett.

b) (2 poeng)



c) (2 poeng)1 er minst og minimalt, 12 er størst og maksimalt

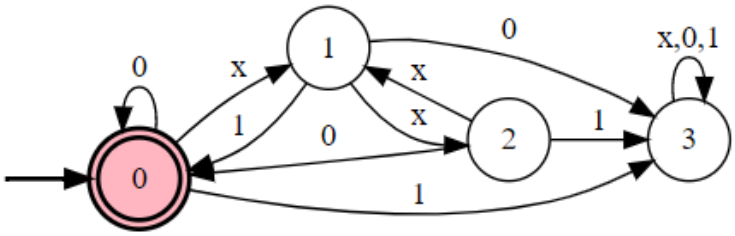
d) $1 < 2 < 3 < 4 < 6 < 12$ eller $1 < 2 < 4 < 3 < 6 < 12$ eller $1 < 3 < 2 < 4 < 6 < 12$ eller $1 < 3 < 2 < 6 < 4 < 12$.

Oppgave 2

a) (2 poeng) $\varepsilon, 0, x1, 0x1, xx0, xxx1, xx0x1$

b) (2 poeng) Strenger i L er enten den tomme strengen, eller en sammensetning av en eller flere strenger på følgende form: et partall antall x etterfulgt av 0, eller et oddetall antall x etterfulgt av 1.

c) (3 poeng)



d) (2 poeng)3. $S \to B \to bA \to baSb \to baabb$

e) (3 poeng)Språket inneholder alle strengene $a^n b^n$, $n \geq 0$. Vi ser også at alle strengene i M må inneholde minst like mange b'er som a'er, så strengene $a^m b^n$ er ikke med i M hvor $m < n$. Fordi en endelig automat har kun endelig mange tilstander, så må det finnes m og n med $m < n$ slik at den er i samme tilstand etter å ha lest a^m og a^n. Da vil den også være i samme tilstand etter å ha lest $a^m b^n$ og $a^n b^n$. Hvis dette er en aksepterende tilstand, så vi den akseptere $a^m b^n$, som ikke er i M. Hvis det ikke er en aksepterende tilstand, så vil den ikke akseptere $a^n b^n$ som er i M. Så en slik automat eksisterer ikke.

Det kan gis ett poeng hvis det ikke gis et bevis, men bare en forklaring at endelig automater ikke har minne.

Oppgave 3

a) I oppgaven skulle det stått Eulerspor. Poenget med Eulerspor er at den skal ha med alle kantene, og dermed må den også ha med alle hjørnene. Siden det er et spor, kan den ikke repetere kanter. Definisjonen av en sti er at den ikke kan repetere hjørner, og derfor heller ikke kanter. En Eulersti er derfor også et Eulerspor. (Det er ikke vanskelig å se at de eneste Eulerstiene er sammenhengende grafer hvor alle hjørnene har grad 1 eller 2.)

Begge tolkninger er aksepterte.

Begge grafene har fire hjørner av grad 3, så ved Eulers teorem, så har ingen av dem noen Eulerspor. Da finnes det heller ikke Eulerstier eller Eulerkretser.

b) G har ingen Hamilton-krets: stiene abd og acd må være med, men da kan ikke kantene ae eller dh være med, og da blir gjenværende graf ikke sammenhengende.

H har Hamiltonkrets: F.eks. $1 \to 3 \to 4 \to 2 \to 6 \to 8 \to 7 \to 5 \to 1$

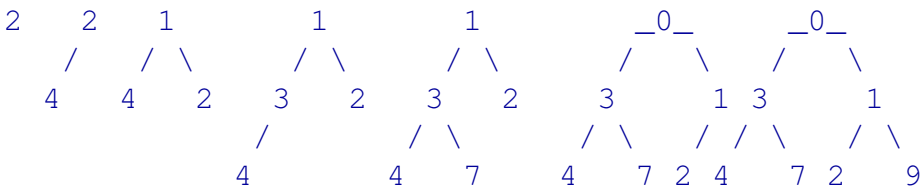
c) Noen alternative begrunnelser for at grafene ikke er isomorfe:

- H har Hamiltonkrets, mens G ikke har det.
- H har en krets med lengde fire, hvor alle hjørnene har grad 3, mens i G danner ikke hjørnene med grad 3 noen krets.
- I H finnes det kanter mellom to hjørner av grad 2, mens det ikke gjør det i G.

Oppgave 4 — 10%

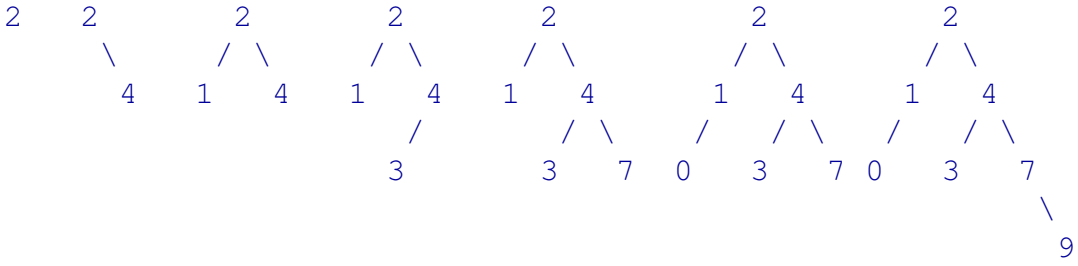
a) Sett tallene 2, 4, 1, 3, 7, 0, 9 inn i en min-heap. Sett dem inn i den rekkefølgen de står, og tegn opp heapen en gang for hvert tall du setter inn.

Her er det bare en korrekt løsning:



b) Sett de samme tallene inn i et binært søketre. Sett dem inn den rekkefølgen de står, og tegn opp søketreet en gang for hvert tall du setter inn.

Her er det bare en korrekt løsning:



Oppgave 5 — 15%

a) Demonstrer Burrows-Wheeler-transformasjonen på ordet «auauau»

Skriver først opp alle rotasjoner. Deretter, rotasjonene sortert alfabetisk. Tegnet «*» viser slutten på teksten. Normalt sorterer vi så «*» kommer etter alle andre tegn:

ROTASJONER	SORTERT
auauau*	auauau*****
*auauau	auau*a**u**
u*auaua	au*aua**u**
au*auau	uauau*a****
uau*aua	uau*au**a**
auau*au	u*auau**a**
uauau*a	*auaua**u**

Siste kolonne i i den sorterte lista, er BWT for ordet «auauau». Transformasjonen blir altså «*uuaaau» (mange klarte denne)

To andre brukbare løsninger fins. Hvis man sorterer «*» foran «a» heller enn etter «u», får man:

```
SORTERT
*auaua u
au*aua u
auau*a u
auauau *
u*auau a
uau*au a
uauau*a
```

Med slik sortering blir transformasjonen «uuu*aaa». En enda merkeligere sortering er å sette «*» etter «a» men foran «u». Transformasjonen er fortsatt reversibel, så også denne sorteringen er brukbar:

```
SORTERT
auauau *
auau*a u
au*aua u
*auaua u
uauau*a
uau*au a
u*auau a
```

Som vi ser, blir transformasjonen «*uuuaaa» med en slik sorteringsrekkefølge.

b) Beskriv halting-problemet, og forklar hvorfor det ikke er løsbart.

Beskrivelse: Halting-problemet går ut på å kunne si hvorvidt et program vil stoppe eller ikke, med en gitt input. Et program som løser dette problemet, vil altså ta et annet program som input.

Det er generelt uløselig, fordi: La oss si at vi har et program H, som løser halting-problemet. Da kan vi lage et program P,som tar et program X som input. Ved hjelp av H sjekker P om X vil stoppe eller ikke når det får sin egen kildekode som input. Hvis H sier X vil stoppe, kjører P en uendelig løkke. Hvis H sier X stopper ikke, vil P avslutte.

Hvis vi kjører P med sin egen programkode som input, får vi en umulig situasjon. Altså er programmet H umulig.

7poeng totalt 3 for definisjonen, 4 for forklaringen.

Her fikk mange problemer. Forklaringen er ikke enkel, dette var ment som en vanskelig oppgave. Mulig å få delvis uttelling for å huske hva problemet går ut på. Mye rot med irrelevant stoff (om P,NP) som jeg gikk gjennom på samme forelesning.

Oppgave 6 — 20%

Analyser de følgende programmene og finn kjøretiden. Regn med at alle parametre er større enn eller lik 0. Bruk Θ om mulig, ellers O og Ω.

Et rett svar er best. Delvis uttelling er mulig for et galt svar hvor kandidaten viser beregningen, og deler av den er korrekt. Trekk for feil notasjon. Det heter $\Theta\left(n^3\right)$, og *ikke* $\Theta = n^3$...

```
public void oppg_a(int n, int m, int [][] tab) {
    for (int i=0; i <= n-1; ++i) {
        for (int j=0; j<m*m; j++) {
            for (int k=1; k<i; ++k) tab[i][k] *= j;
        }
    }
}
```

$T(m,n) \in \Theta(m^2n^2)$. Evt. $\Theta\left((mn)^2\right)$

```
public void oppg_b(int n, int m, int [] tab) {
    if (n > m) return;
    for (int i=0; i <= n-1; ++i) {
        for (int j=0; j<m; j+=3) {
            tab[i+j*n] = 5;
        }
    }
}
```

$\Omega(1), O(nm)$

```
public void oppg_c(int n, int[][] tab, int x) {
    for (int j=0; j < n-1; ++j) {
        for (int k = n; k>0; --k) tab[j][k]+=x;
    }
    if (n>0) oppg_c(n/2, tab, x/3);
}
```

$T(n) \in \Theta(n^2)$, i følge mastermetoden.

```
public void oppg_d(int n, int[] tab, int x) {
    tab[x] += n;
    if (n>0) {
        oppg_d(n/2, tab, x/3+1);
        oppg_d(n/2, tab, x/3-1);
        oppg_d(n/2, tab, 2*x/3+1);
        oppg_d(n/2, tab, 2*x/3-1);
```

```
        }
}
```

$T(x) \in \Theta(n^2)$. etter mastermetoden

Oppgave 7 10%

Det er 968 postnumre i Nord-Norge. Jeg vil lagre disse i en hashtabell med dobbel hashing. Foreslå hashfunksjoner og tabellstørrelse så jeg får effektive oppslag og rimelig bruk av plass. Forklar valg du gjør.

Se vedlegget for noen tall som kan være nyttige.

Dobbel hashing bruker to hashfunksjoner. En H_1 som sprer tallene jevnt utover hele tabellen, og en H_2 som brukes ved kollisjoner. H_2 bør kunne gi *ulike* resultater for postnumre som H_1 hasher til samme posisjon. Derfor, trekk hvis H_2 bruker H_1 som eneste input.

Mange greie måter, f.eks. restdivisjon med primtall, eller multiplikativ hash med toerpotenser. h2 må vare brukbar for tabellstørrelsen. (aldri 0, relativt prim til tabellstørrelsen) h1 må gi god spredning. funksjonene bør kunne beregnes raskt.

Tabellstørrelsen kan gi overhead på 20–25%, ikke så mye mer.

Toerpotens 1024. Mindre er ubrukelig, høyere toerpotenser gir for mye overhead. Primtall: 1151 og 1171 er gode. Alle over 968 er brukelige, men ikke så gode. 971 gir en veldig full tabell med mange kollisjoner. 1583 gir mye overhead.

Løsning 1:

Tabellstørrelse primtallet p: $H_1(x) = x \mod p$, $H_2(x) = x \mod (p-1) + 1$

I begge tilfeller er x postnummeret.

Løsning 2:

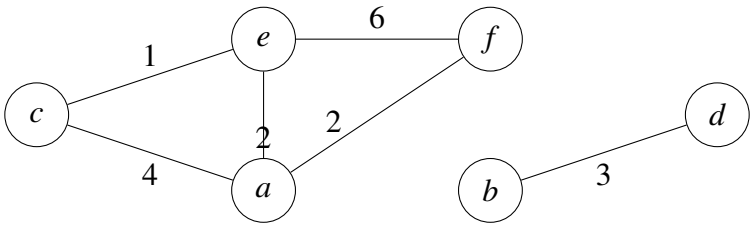
Tabellstørrelse 1024: $H_1(x) = (xA - \lfloor xA \rfloor) * 1024$; A er et passende desimaltall, f.eks. $\left(\sqrt{5} - 1\right)/2$. (A fra læreboka.) $H_2(x) = (2 * |x| + 1)$, som gir oddetall. (Oddetall har ingen felles faktorer med 1024, som bare har tallet 2 som faktor.)

Mange fant frem til løsning 1, men avverget deretter full uttelling ved å gå inn for at x i H_2 skulle være verdien fra H_1 (i stedet for postnummeret). Jeg vet ikke hvor de har fått den idéen fra, men det ødelegger hele poenget med dobbel hashing, som er at H_2 skal kunne produsere *ulike* verdier (hopplengder) når H_1 har hashet to *ulike* postnumre til samme sted. Men hashfunksjonen H_2 kan ikke lage *ulike* verdier hvis dens input er output-verdiene fra H_1, som er *like* når det dreier seg om en kollisjon! Trist å måtte trekke mange for dette. Mange ville også at x, argumentet til H_2, skulle være et *primtall*. Også problematisk/absurd; vi kan hverken kreve at postnumrene eller output fra H_1 skal være primtall. Da blir det i så fall mange postnumre som ikke kan lagres i denne hashtabellen. Hvor kommer idéen fra?

Det er mange andre gode løsninger også, men få fant frem til dem. Noen fikk trekk for å bruke en H_2 som gir tall som har felles faktorer med tabellstørrelsen. Mye av poenget med å bruke primtall eller toerpotens som tabellstørrelse, er at det da blir lettere å lage en brukbar H_2. Hvis tabellstørrelsen f.eks. er 1200, så har 1200 faktorene 2, 3 og 5. Restdivisjoner (med 1200, eller et primtall i nærheten) kan gi alle disse tallene, og er dermed ikke egnet. Oddetallsfunksjonen kan gi 3 og 5 som resultat, og er dermed heller ikke egnet for en slik tabellstørrelse. Restdivisjon med noe annet enn tabellstørrelsen gir ujevn spredning, og passer derfor ikke for H_1.

Oppgave 7 15%

Gitt denne grafen:



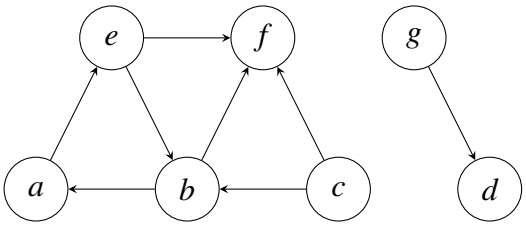
a) Finn og tegn et minimalt spenntre med vekt, eller forklar hvorfor det ikke er mulig.

Et spenntre er ikke mulig, fordi grafen ikke henger sammen.

b) Finn korteste vei fra c til de andre nodene. Tegn korteste-vei treet.

Veiene, med lengde: ce:1, cea:3, ceaf:5. b og d får uendelig avstand.

c) Gitt denne grafen:



Finn de sterkt sammenhengende komponentene, eller forklar hvorfor det ikke er mulig.

Komponenter: aeb, f, c, g, d For full uttelling må alle komponentene være med, ikke bare den første (som er den eneste komponenten med flere noder).

Vedlegg

Noen primtall

971, 991, 1013, 1031, 1051, 1087, 1109, 1151, 1171, 1193, 1213, 1237, 1283, 1301, 1361, 1399, 1433, 1481, 1523, 1583

Noen toerpotenser

256, 512, 1024, 2048, 4096, 8192, 16384