NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Eksamensoppgave i Algoritmer og datastrukturer, IDATT2101

Faglig kontakt under eksamen: Helge Hafting
Tlf.: 924 386 56

Eksamensdato: 6. desember 2022
Eksamenstid (fra–til): 09:00–13:00
Tillatte hjelpemiddel: ett A4-ark med notater

Annen informasjon: Løsningsforslag

Målform/språk: bokmål
Antall sider (uten forside): 6
Antall sider vedlegg: 0

Informasjon om trykking av eksamensoppgave			
Originalen er:			
1-sidig	☐	**2-sidig**	☒
sort/hvit	☐	**farger**	☒
		Flervalgskjema?	☐

Kontrollert av

. . . .
Dato Sign

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

Oppgave 1 **20%**

Analyser disse programmene. Bruk Θ om mulig. Om ikke, bruk O og Ω. Alle parametre er positive.

```
int prog_a(int a, int b, int c) {
    int sum = 0;
    for (int i=0; i<a ; ++i) {
        for (int j=0; j<c; ++j) {
            for (int k=b; k>0; --k) {
                sum += i*j-k;
            }
        }
    }
    return sum;
}
```

```
int prog_b(int q, int p, int r) {
    int sum = 0;
    for (int i=0; i<q ; i+=r) {
        sum += i*(q-r);
        if (sum > p) return sum;
    }
}
```

---

a: $\Theta(abc)$

b: $O(q/r), \Omega(1)$

```
int prog_c(int q, int n, int [] tab) {
    int sum = 0;
    if (n > 0) {
        sum += prog_c(q/2, n/4, tab);
        for (int i=0; i<n; ++i) sum += tab[i*n+q];
        sum += prog_c(q/2, n/4, tab);
    }
    return sum;
}
```

```
double prog_d(int n, float x) {
    if (n == 0) return 1.0;
    else return x * prog_d(n/2, x);
}
```

```
int prog_e(int a, int b, int c) {

    int sum = 1;
    for (int i = 1; i < a; ++i) sum *= i;
    for (int j = 1; j < b; ++j) sum += j;
    return sum;

}
```

c: $\Theta(n)$ d: $\Theta(\log n)$ e: $\Theta(a+b)$

Oppgave 2 — 5%

a) Sett disse tallene $(5, 7, 1, 0, 8, 6, 2)$ inn i en max-heap. Tegn opp hvordan heapen ser ut etter hvert tall.

b) Sett de samme tallene inn i ett binært søketre. Tegn opp hvordan treet ser ut etter hvert tall.

En heap blir til:

```
5    7   7      7      8       _8_     _8_
    /   / \    / \    / \     /   \   /   \
   5   5   1  5   1  7   1   7     6 7     6
             /      / \     / \   / / \   / \
            0      0   5   0   5 1 0   5 1   2
```

Slik blir søketreet til slutt:

```
    _5_
   /   \
  1     7
 / \   / \
0   2 6   8
```

Oppgave 3 — 10%

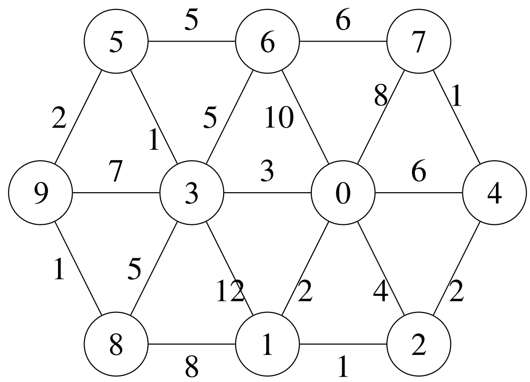
a) Fortell kort om Burrows-Wheeler transformasjonen. Hva er det den oppnår, som er nyttig for datakompresjon?

b) Jeg har brukt Burrows-Wheeler transformasjonen på et ord, og fått strengen *OHHO. «*» er slutt-tegnet, som kommer etter alle andre tegn i sorteringsrekkefølgen. Finn ut hva det opprinnelige ordet var, og vis hvordan du gjorde det.

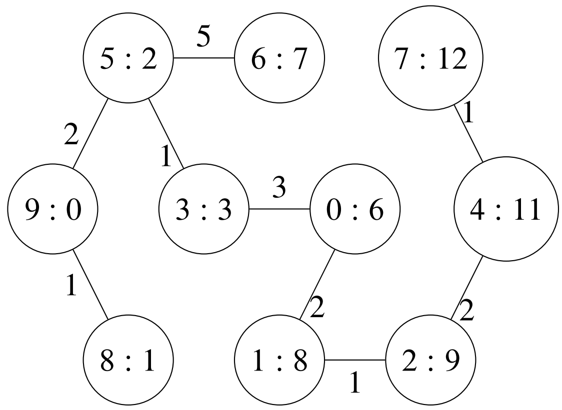
Transformasjonen komprimerer ikke selv data. Men hvis inndata inneholder repetisjoner (f.eks. ord som brukes flere ganger), vil utdata inneholde repeterte *tegn*. Repeterte tegn kan lett komprimeres med run-length coding.

Det opprinnelige ordet var «HOHO*». Her må man også vise hvordan man fant ut det. Metoden fra forelesningen, eller lignende metoder fra nettet eller ad hoc.

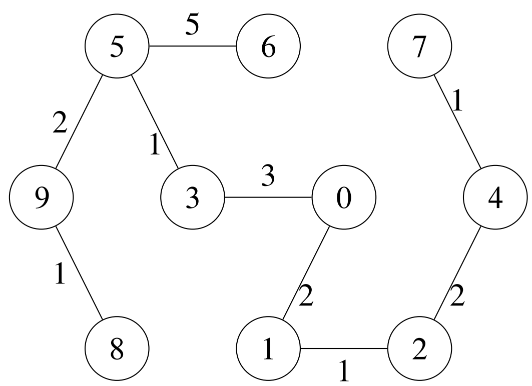
Oppgave 4 **20%**



a) Finn korteste vei fra node nr. 9 til alle andre noder. Tegn korteste-vei treet, og skriv opp avstanden til hver node.

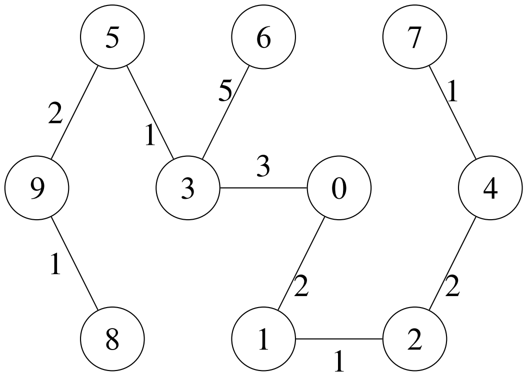


b) Finn og tegn et minimalt spenntre for grafen. Skriv opp den totale vekta for spenntreet.



Vekten av spenntreet er 18.

c) Finn og tegn et annet minimalt spenntre for grafen, eller forklar hvorfor det ikke er mulig.



Vekten av spenntreet er 18 her også. Å bytte om løsningene i b) og c) er også en korrekt løsning. Det er ingen andre minimale spenntrær i denne grafen.

Oppgave 5 — 25%

a) Grafen G har 8 noder. Hvis vi sorterer den med topologisk sortering, får vi enten rekkefølgen 1, 3, 0, 7, 2, 5, 4, 6 eller rekkefølgen 1, 3, 0, 7, 5, 2, 4, 6.

Hvor mange sterkt sammenhengende komponenter er det i grafen G?

Siden grafen kan sorteres topologisk, har den ingen rundturer. Dermed har den ingen komponenter med mer enn én node. Altså er det 8 sterkt sammenhengende komponenter, nemlig nodene i grafen.

b) Heapsort og innsettingssortering er ikke de mest populære sorteringsalgoritmene. Men når kan det være lurt å bruke disse?

Heapsort er fint når man trenger garantert $O(n \log n)$ kjøretid, og ikke har råd til quicksorts verste tilfelle. Kan også brukes for å speede opp quicksort, når man oppdager at rekursjonen går for dypt.

Innsettingssortering er god på små datasett. Den er derfor egnet når vi vet at det bare blir små datasett å sortere. (Avhenger av hva vi driver med.) Innsettingssortering kan også brukes for å speede opp siste del av quicksort, når intervallene har blitt små nok.

c) Lag frekvenstabell og huffmanntre med utgangspunkt i ordet «settekasse». Skriv binærkoden for «settekasse» med utgangspunkt i treet du lagde. Hvor mange bits trengte du?

Frekvenstabell: s:3 e:3 t:2 k:1 a:1

Det er mange mulige huffmantrær. derfor ikke noe løsningsforslag for dette. Ulike trær gir ulik kode, men alle muligheter gir like mange bits for «s», like mange for «e», osv.

Mitt tre gav disse kodene:

s 00
e 01
t 10
k 110
a 111

Andre kan ha fått andre bitkombinasjoner, men «s», «e», «t» må være to-bits, «k» og «a» må være tre-bits. Med mine koder blir settekasse 0001101001110111000001. Andre kan ha fått et annet bitmønster, men det må være 22 bits.

Gav også full uttelling for en variant som hadde med et «end»-tegn. Sånt kan jo være nyttig når en bruker huffman i praktiske anvendelser. I det tilfellet ble det 27 bits.

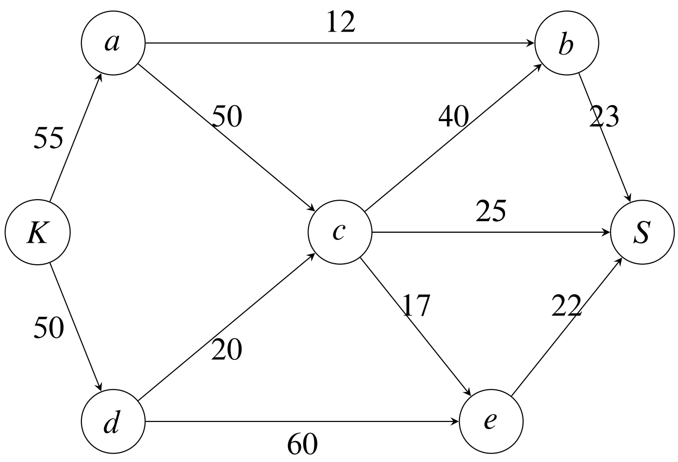
d) Hva vil det si, at et problem er NP-komplett? Gi også eksempel på et slikt problem, og fortell hva problemet går ut på.

Etter definisjonen: Problemer i **NP** har en løsning som kan *sjekkes* i polynomisk tid. (Det er ikke et krav at løsningen kan *finnes* i polynomisk tid.) NP-komplette problemer (NPC) er en gruppe tyngre problemer i NP. Ethvert NP-problem transformeres til et NPC-problem på polynomisk tid. Så en løsning for *ett* NPC-problem, kan brukes for å løse alle NP-problemer. Vi kjenner ikke polynomiske løsninger for noe NPC-problem.

Mange problemer fra pensum, som travelling salesman, ryggsekkproblemet, delsumproblemet, og andre. For full uttelling må det også stå hva det presenterte problemet går ut på.

Oppgave 6 20%

Bruk denne grafen i de to første deloppgavene.



a) Finn maksimal flyt fra K til S. Nytt flytøkende veier, og skriv opp hver vei og hvor mye flyt du legger til langs veien.

Mulige flytøkende veier:

```
Økning : Flytøkende vei
    22   K d e S
    20   K d c S
     5   K a c S
    12   K a b S
    11   K a c b S
Maksimal flyt ble 70
```

Det er mulig å ha andre rekkefølger, men summen må bli 70.

b) Sorter grafen topologisk, eller forklar hvorfor det ikke er mulig.

Det er fire korrekte rekkefølger for topologisk sortering. Alle begynner med K og slutter med S.

a og d før c

c før b og e

KadcbeS
KdacbeS
KadcebS
KdacebS

c) Tegn et binærtre med tre noder. Det skal være slik at vi får ulike rekkefølger, om vi skriver nodene i post-orden, pre-orden eller in-orden.

Skriv nodene i pre-orden, post-orden, in-orden og nivå-orden.

Min løsning:

```
     a
    / \
   b   c
```

postorden: bca
preorden: abc
inorden: bac
nivåorden: abc

Alternativ løsninger:

```
     a                   p
    /                     \
   b                       q
    \                     /
     c                   r
```

postorden: cba	rqp
preorden: abc	pqr
inorden: bca	prq
nivåorden: abc	pqr