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: 20. desember 2021
Eksamenstid (fra–til): 09:00–14:00
Hjelpemiddelkode/Tillatte hjelpemiddel: A / Alle hjelpemidler tillatt.

Annen informasjon: Løsningsforslag

Målform/språk: bokmål
Antall sider (uten forside): 17
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.

I noen oppgaver må du bruke *siste siffer i kandidatnummeret ditt* i oppgaven. (Rett siffer burde være lett å finne. F.eks. er «7» siste siffer i «100247».) Hensikten er at ulike studenter skal få litt ulike oppgaver. Hvis *feil* siffer likevel brukes, underkjennes deloppgaven, og det gir mistanke om juks/plagiat.

Oppgave 1 — 24%

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

(Om det er vanskelig å skrive «Θ» på ditt tastatur, kan du skrive «Theta» i stedet».)

```
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) {
            sum += i*j;
            if (sum > b) return sum - b;
        }
    }
    return sum;
}
```

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

a: $\Omega(c)$, $O(ac)$

Dobbeltløkka kan brytes tidlig, pga. if-testen. Men første gang indre løkke kjøres, er $i = 0$, så summen forblir 0. Dermed kan ikke summen bli større enn den positive b, før i blir 1. Altså må den indre løkka kjøres minst én gang, før vi kan få et tidlig brudd. Derfor $T(a,c) \in \Omega(c) \notin \Omega(1)$.

b: $\Theta(qp + r)$

```
int prog_c(int q, int n, int [] tab) {

    int sum = 0;
    if (q > 0) {
       sum += prog_c(q/2, n/4, tab);
       for (int i=0; i<q; ++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-1, x);

}
```

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

    int sum = 0;
    if (a >= b) return -1;
    for (int i = a; i < b; i += c) sum += i*i;
    return sum;

}
```

```
int prog_f(int n) {

    if (n > 0) {
       int produkt = 1;
       for (int i = 1; i <= 6; ++i) {
          produkt *= prog_f(n/3);
       }
       return produkt;
    } else return 1;

}
```

c: $\Theta(q \log q)$ d: $\Theta(n)$ e: $\Omega(1)$, $O((b-a)/c)$ f: $\Theta\left(n^{\log_3(6)}\right) \approx \Theta\left(n^{1,63}\right)$

Oppgave 2 6%

La x være siste siffer i kandidatnummeret ditt.

a) Sett disse tallene $(9, 1, x, 8, 2, 5, 7, 6)$ inn i en min-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.

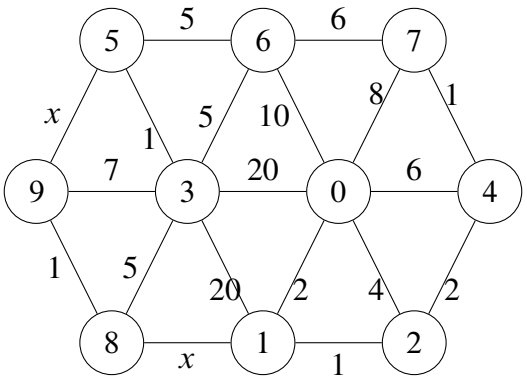
```
Viser bare den ferdige heapen i hvert tilfelle.
Heap med x=0        x=1          x=2          x=3          x=4
      _0_           _1_          _1_          _1_          _1_
     /   \         /   \        /   \        /   \        /   \
    2     1       2     1      2     2      2     3      2     4
   / \   / \     / \   / \    / \   / \    / \   / \    / \   / \
  6   8 5   7   6   8 5   7  6   8 5   7  6   8 5   7  6   8 5   7
 /             /            /            /            /
9             9            9            9            9

Heap med x=5        x=6          x=7          x=8          x=9
      _1_           _1_          _1_          _1_          _1_
     /   \         /   \        /   \        /   \        /   \
    2     5       2     5      2     5      2     5      2     5
   / \   / \     / \   / \    / \   / \    / \   / \    / \   / \
  6   8 5   7   6   8 6   7  6   8 7   7  6   8 8   7  6   8 9   7
 /             /            /            /            /
9             9            9            9            9

Viser bare det ferdige søketreet i hvert tilfelle.
Med duplikater, tre muligheter:
Duplikat til venstre, ingen dup., eller dup. til høyre
Søketre x=0  x=1   x=1   x=1     x=2   x=2   x=2     x=3     x=4
      9       9     9     9       9     9     9       9       9
     /       /     /     /       /     /     /       /       /
    1       1     1     1       1     1     1       1       1
   / \     / \     \     \       \     \     \       \       \
  0   8   1   8     8     1       2     2     2       3       4
     /       /     /       \     / \     \     \     / \     / \
    2       2     2         8   2   8     8     8   2   8   2   8
     \       \     \       /     /     /     /       /       /
      5       5     5     2       5     5     2       5       5
       \       \     \     \       \     \     \       \       \
        7       7     7     5       7     7     5       7       7
       /       /     /       \     /             \     /       /
      6       6     6         7   6               7   6       6
                              /                   /
                             6                   6
```

```
Viser bare det ferdige søketreet i hvert tilfelle.
Duplikater gir tre muligheter:
Duplikat til venstre, ingen dup., eller dup. til høyre
  x=5       x=5      x=5        x=6        x=6        x=6
   9         9        9          9          9          9
  /         /        /          /          /          /
 1         1        1          1          1          1
  \         \        \          \          \          \
  _5_       5        5         _6_        _6_        _6_
 /   \     / \      / \       /   \      /   \      /   \
2     8   2   8    2   8     2     8    2     8    2     8
 \   /       /        /       \   /      \   /      \   /
  5 7       7        5         5 7        5 7        5 7
   /        /          \         \                     /
  6        6            7         6                   6
                       /
                      6
```

```
Viser bare det ferdige søketreet i hvert tilfelle.
Duplikater gir tre muligheter:
Duplikat til venstre, ingen dup., eller dup. til høyre
 x=7      x=7       x=7       x=8 x=8 x=8      x=9 x=9 x=9
  9        9         9         9   9   9        9   9   9
 /        /         /         /   /   /        /   /   / \
1        1         1         1   1   1        1   1   1   9
 \        \         \         \   \   \        \   \   \
  7        7       _7_         8   8   8        9   8   8
 / \      / \     /   \       /   /   / \      /   /   /
2   8    2   8   2     8     8   2   2   8    8   2   2
 \        \       \   /     /     \   \      /     \   \
  5        5       5 7     2       5   5    2       5   5
   \        \       \       \       \   \    \       \   \
    7        6       6       5       7   7    5       7   7
   /                          \     /   /      \     /   /
  6                            7   6   6        7   6   6
                              /                 /
                             6                 6
```

Oppgave 3 25%

La x være siste siffer i kandidatnummeret ditt. To kanter i grafen har denne vekta.



a) Finn korteste vei fra node nr. x 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.

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

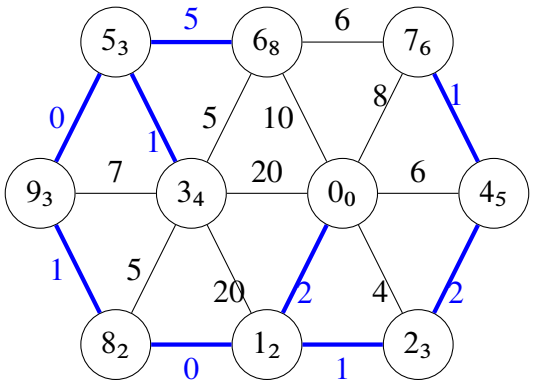
d) Dijkstras algoritme og Bellman-Ford algoritmen løser samme problem, men med ulik kompleksitet. Grei ut om når vi bruker den ene algoritmen, og når vi bruker den andre.

e) I analysen av Prims algoritme, står det at $O\left((N+K)\log N\right)$ kan forenkles til $\mathrm{O}(K\log N)$. Forklar hvorfor dette stemmer.

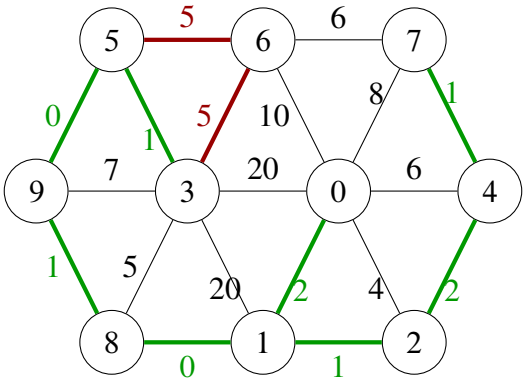
Løsninger på oppgave 3a. Kanter i korteste vei-trær vises med blått. Der det er flere rette løsninger, vises alternative kanter med rødt.

Spenntrær vises med grønne kanter. Røde kanter viser alternative spenntrær; bruk halvparten av de røde kantene for å ha et gyldig spenntre.

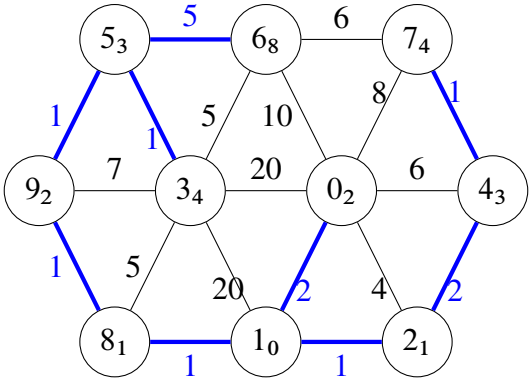
Løsning på oppgave 3a for $x = 0$



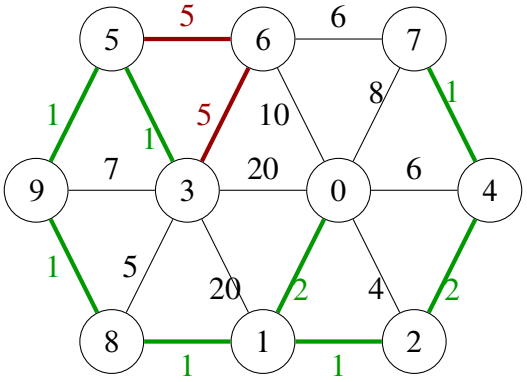
Løsning på oppgave 3b for $x = 0$. To mulige minimale spenntrær, bruk én av de to av de røde kantene. Vekt av spenntreet er 13.



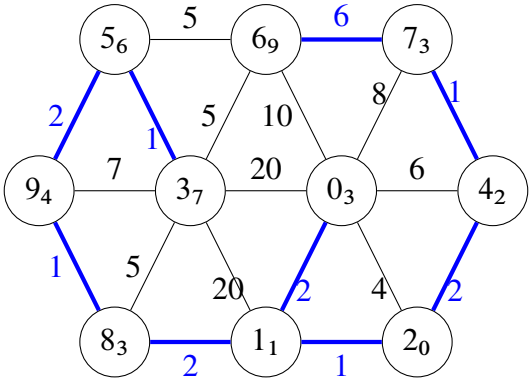
Løsning på oppgave 3a for $x = 1$



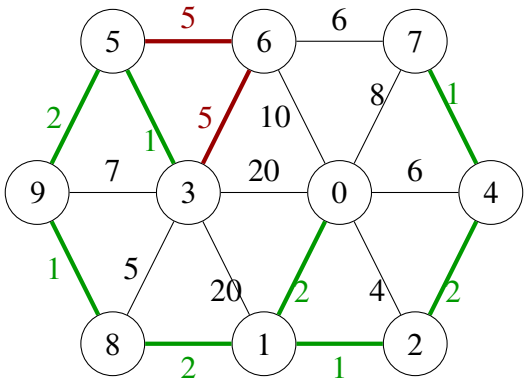
Løsning på oppgave 3b for $x = 1$. Vekt av spenntreet er 15.



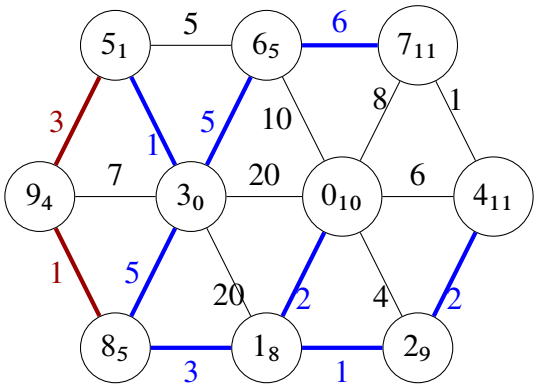
Løsning på oppgave 3a for $x = 2$



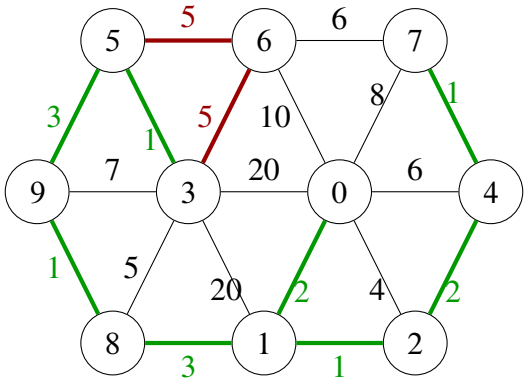
Løsning på oppgave 3b for $x = 2$. Vekten av spenntreet er 17.



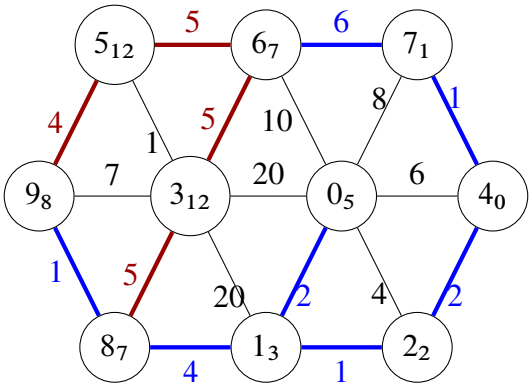
Løsning på oppgave 3a for $x = 3$. Det er to rette løsninger, bruk bare en av de to røde kantene.



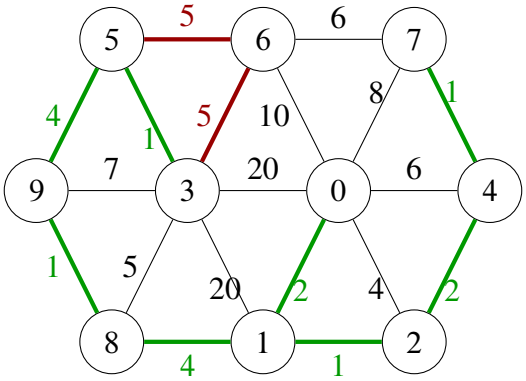
Løsning på oppgave 3b for $x = 3$. Vekt av spenntreet er 19.



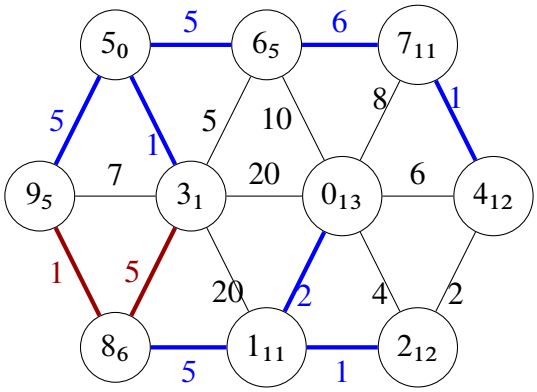
Løsning på oppgave 3a for $x = 4$. Det er fire rette løsninger. Bruk én av de røde kantene koblet til node 5, og en av de røde kantene koblet til node 3.



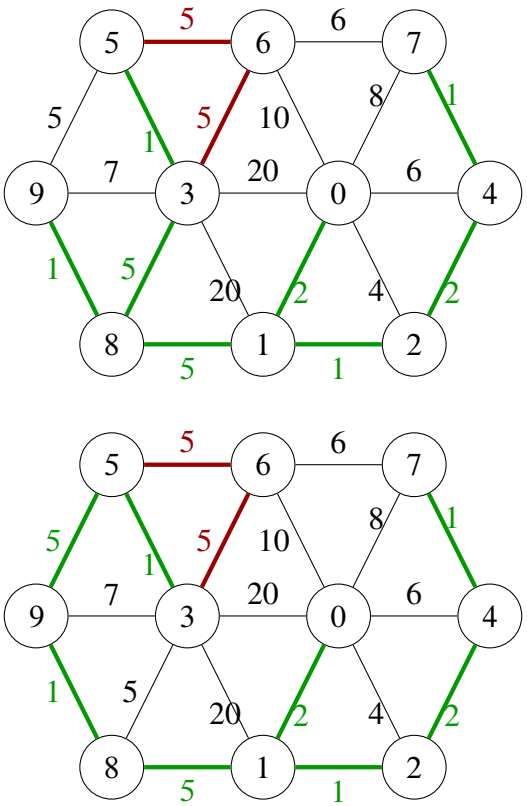
Løsning på oppgave 3b for $x = 4$. Vekten av spenntreet er 21.



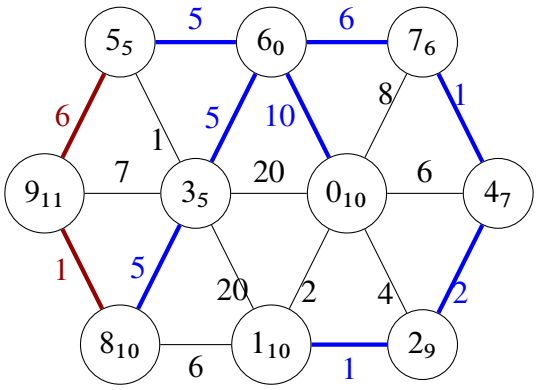
Løsning på oppgave 3a for $x = 5$. Det er to rette løsninger, bruk bare en av de røde kantene koblet til node 8.



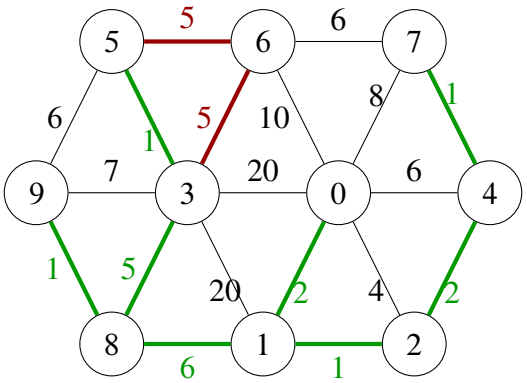
Løsning på oppgave 3b for $x = 5$. Vekten av spenntreet er 23. Fire mulige løsninger. Hver tegning gir to muligheter ved å beholde en av de røde kantene.

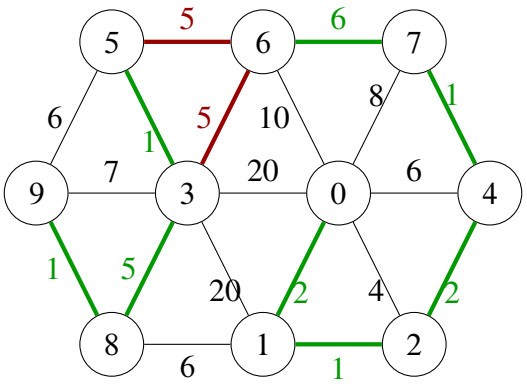


Løsning på oppgave 3a for $x = 6$. Det er to rette løsninger. Bruk bare en av de røde kantene som er koblet til node 9.

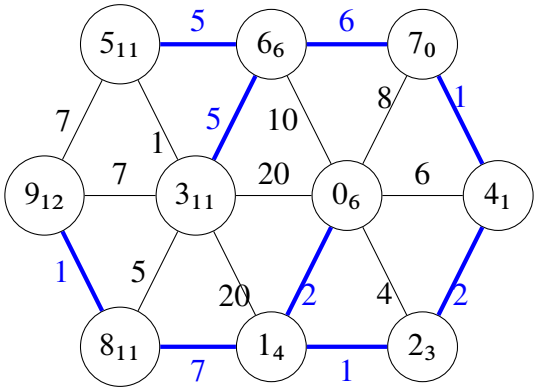


Løsning på oppgave 3b for $x = 6$. Vekt av spenntreet er 24. Det er fire mulige minimale spenntrær, i hver av tegningene kan man få to spenntrær ved å beholde en av de røde kantene.

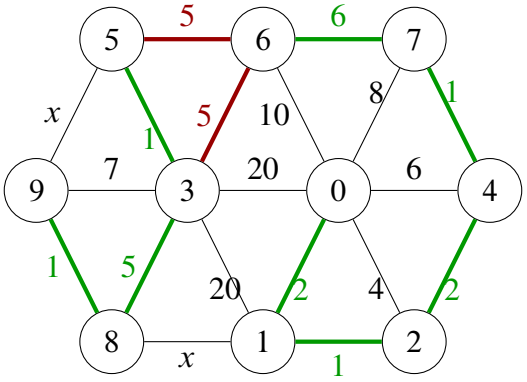




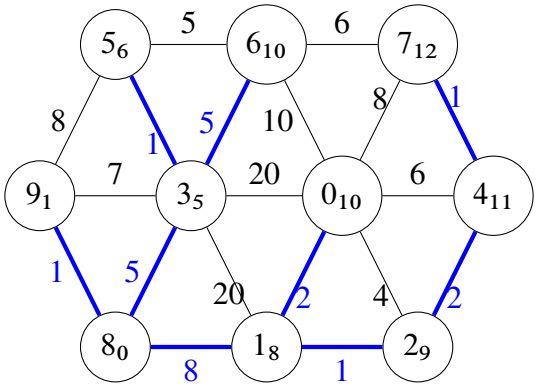
Løsning på oppgave 3a for $x = 7$



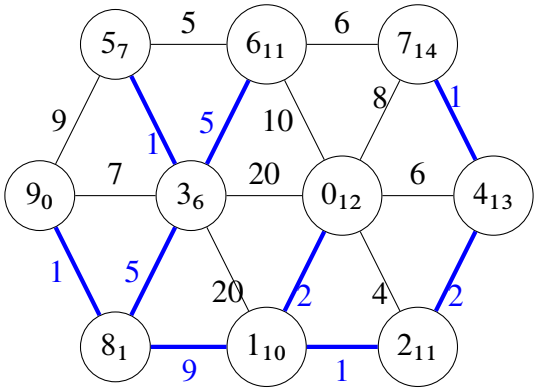
Løsning på oppgave 3b for $x = 7, x = 8, x = 9$. Vekt av spenntreet er 24.



Løsning på oppgave 3a for $x = 8$



Løsning på oppgave 3a for $x = 9$



Løsning oppgave 3c

Løsningen for oppgave 3b viser to (eller fire) mulige minimale spenntrær, ved å enten bruke kanten 5–6, eller kanten 3–6. Det er altså mulig å finne minst ett alternativt minimalt spenntre i denne oppgaven.

Løsning oppgave 3d

Dijkstras algoritme håndterer ikke negative kanter, så for slike grafer er vi nødt til å bruke Bellmann-Ford algoritmen.

Ellers foretrekker vi lavest kompleksitet. Dijkstras algoritme har kompleksitet $O\left((N+K)\log N\right)$, Bellman-Ford er $O\left(KN\right)$. For de fleste grafer (med positiv vekting) vil dermed Dijkstras algoritme være å foretrekke. Unntaket er hvis $K < \log N$. For en slik graf, (som har for få kanter til å henge sammen), vil Bellman-Ford algoritmen være raskere.

Løsning oppgave 3e

Vise at $O\left((N+K)\log N\right)$ kan forenkles til $O\left(K\log N\right)$ for Prims algorime: Prims algoritme finner et minimalt spenntre. Den brukes derfor bare på grafer som henger sammen, for ellers finnes ikke noe spenntre. For at grafen skal henge sammen, må $K \geqslant N-1$. I asymptotisk analyse kan vi se bort fra «-1», og sitter igjen med $K \geqslant N$. Når K er størst, kan vi fjerne N fra $K+N$, dermed forenkles uttrykket til $O\left(K\log N\right)$.

Kan også bruke definisjonen på O:

$0 \leqslant (N+K)\log N \leqslant c \cdot K\log N$. (For $N \geqslant N_0$ og $K \geqslant K_0$)

Deler med $K\log N$ på begge sider:

$0 \leqslant \frac{N+K}{K} = \frac{N}{K} + 1 \leqslant c$

Så vet vi at $K \geqslant N-1$. For større K blir $\frac{N}{K}$ mindre, så setter inn det verste tilfellet som er $K = N-1$. Da får vi:

$0 \leqslant \frac{N}{N-1} + 1 \leqslant c$. Fra matematikken vet vi at $\lim_{N\to\infty} \frac{N}{N+1} = 1$

Så for tilstrekkelig høye N, har vi $0 \leqslant 1+1 \leqslant c$, som ser greit ut.

Ulikheten går opp f.eks. for $c = 3$, og alle $N \geqslant N_0 = 2$.

Oppgave 4 25%

a) Fortell om sterke og svake sider ved quicksort og innsettingssortering. Gi eksempel på når du vil bruke den ene sorteringa, og når du vil bruke den andre.

	Quicksort	**Innsettingssort**
Sterke	Store datasett	Små datasett, $n < 100$ Nesten/helt sorterte sett
Svake	Datasett laget spesielt for n^2 worst case	Store tabeller
Når bruke	For det meste	Hjelp for quicksort, datasett vi vet er små/passende

b) Når vi implementerer quicsort, er det lett å gjøre feil som gir unødvendig dårlig ytelse. Fortell om slike problemer, og hvordan vi unngår dem.

- n^2 skjevdeling ved sortering av sorterte data. unngå ved å velge midterste element som delingstall, evt. median av 3/5/7/…
- n^2 skjevdeling ved sortering av duplikater. <,> i indre løkker, heller enn <= og >=
- for dyp rekursjon. Test dybden og bytt til heapsort, eller iterer på det største intervallet
- skjevdeling med 0 og n elementer, som gir uendelig løkke. Unngås med median3sort, som gir oss ihvertfall ett «lite» og ett «stort» tall. Unngås også av andre metoder som ikke sorterer delingstallet omigjen.
- Bedre ytelse ved å bytte til innsettingssort når intervallene blir tilstrekkelig små.

c) Kan vi bruke dynamisk programmering for å lage en enkel og effektiv sorteringsalgoritme?

Hvis ja, foreslå en slik algoritme. Hvis nei, forklar hvorfor dette er vanskelig.

Sannsynligvis ikke? Dynamisk programmering hjelper når vi deler opp et stort problem, for så å løse *samme* delproblem flere ganger. Sortering kan deles opp, slik f.eks. quicksort og flettesortering gjør. Men det er lite sannsynlig at vi ser samme delproblem flere ganger. Å sjekke om en deltabell matcher en tidligere sortert tabell, vil lett ta mer tid enn det sorteringsarbeidet vi sparer.

d) Hva vil det si, at et problem er i kompleksitetsklassen **NP** ? Gi også eksempel på et problem som er i **NP**, og fortell hva problemet går ut på.

Etter definisjonen: Problemer i **NP** har *løsning* som kan sjekkes i polynomisk tid (der n er størrelsen på problemet.) Problemet kan ikke nødvendigvis *løses* på polynomisk tid.

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

e) Pristilsynet vil ha oversikt over 10 000 dagligvarer. De vil ha varane i en hashtabell, så de kan slå opp via pris. De vil for eksempel finne «alle varer som koster 200 kr» for å sjekke om butikkene følger regelverket. Tilsvarende for alle andre priser. Mange varer har priser som ender i enten 0 eller 9.

Foreslå tabellstørrelse, hashfunksjon og kollisjonshåndtering så dette blir effektivt. (Du trenger ikke finne konkrete primtall/toerpotenser. Om du f.eks. trenger et primtall over 1000, skriv «la p være neste primtal høyere enn 1000.» Så kan du bruke p i svaret ditt.)

Her fins mange løsninger.

Det nevnes spesielt å finne alle varer med samme pris. Da er kollisjonshåndtering med lenka lister praktisk. Varer med samme pris hasher til samme sted, og havner på samme liste. Multiplikativ hashfunksjon kan brukes uten føringer for tabellstørrelse, når vi bruker lenka lister. Restdivisjon kan brukes, da er det en fordel å dele med et primtall (eller ihvertfall noe som ikke er tierpotens, siden mange priser ender på samme tall.) Tabellen bør vare stor nok til at *ulike* priser sjelden kolliderer. Det står ikke nok i oppgaven om hvor mange *like* priser det er, men om lag 10 000 vil vel være brukbart. Noe mindre, hvis det er *mange like priser.* For restdivisjon velger man et litt større primtall.

Åpen adressering er også en mulighet. Da trenger vi et overhead på 20% for å få god ytelse. Tabellstørrelsen bør altså være 12 000 eller mer.

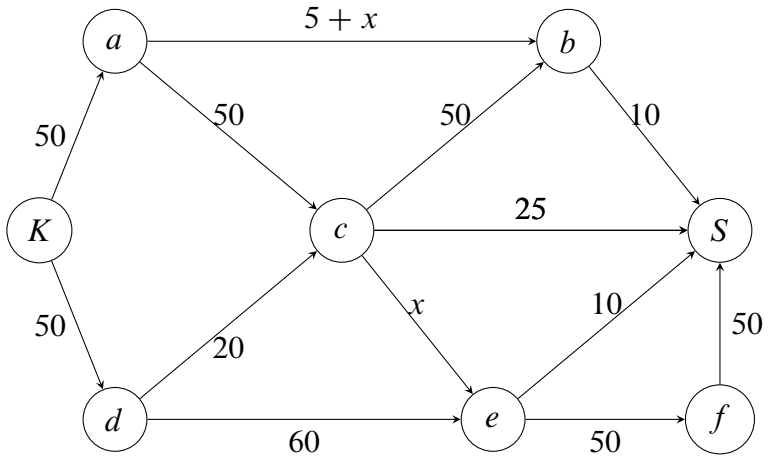
Åpen adressering med restdivisjon: Tabellstørrelse blir første primtall etter 12 000. Første hashfunksjon blir restdivisjon med tabellstørrelsen, andre hashfunksjon blir restdivisjon med (tabellstørrelse-1), hvor vi deretter legger til 1 for å unngå 0. Restdivisjon med mindre tall kan også brukes, fordi kortere hopp kan være bedre for cache.

Åpen adressering med multiplikativ hash: alle tabellstørrelser kan brukes, men hashfunksjon nr 2 kan stille krav til tabellstørrelsen. Divisjonsbaserte funksjoner vil ofte kreve en primtallstørrelse. Eventuelt kan man gå for rask heltallsbasert multiplikativ hash, tabellstørrelse som er en toerpotens (16384), og oddetallsfunksjonen som hashfunksjon nr 2.

Andre gode løsninger er mulig, og må vurderes for seg.

Oppgave 5 — 20%

La x være siste siffer i kandidatnummeret ditt. Legg merke til at x brukes i noen av kantvektene i grafen:



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.

b) Forklar hva fenomenet «flytkansellering» går ut på, når vi jobber med maksimum flyt.

c) Tegn et binærtre med fire noder. Det skal være slik at enten vi skriver ut nodene i postordenrekkefølge eller in-orden, så får vi samme rekkefølge.

Løsningsforslag. Andre flytøkende veier er mulig.

Oppgave 5a, for $x = 0$

```
Økning : Flytøkende vei
    10   S d e K
    20   S d c K
     5   S a c K
     5   S a b K
    20   S d e f K
     5   S a c b K
    20   S a c d e f K
Max flyt fra K til S ble 85
```

Oppgave 5a, for $x = 1$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
     6   K a b S
    20   K d e f S
     4   K a c b S
     1   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 86
```

Oppgave 5a, for $x = 2$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
     7   K a b S
    20   K d e f S
     3   K a c b S
     2   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 87
```

Oppgave 5a, for $x = 3$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
     8   K a b S
    20   K d e f S
     2   K a c b S
     3   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 88
```

Oppgave 5a, for $x = 4$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
     9   K a b S
    20   K d e f S
     1   K a c b S
     4   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 89
```

Oppgave 5a, for $x = 5$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
    10   K a b S
    20   K d e f S
     5   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 90
```

Oppgave 5a, for $x = 6$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
    10   K a b S
    20   K d e f S
     6   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 91
```

Oppgave 5a, for $x = 7$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
    10   K a b S
    20   K d e f S
     7   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 92
```

Oppgave 5a, for $x = 8$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
    10   K a b S
    20   K d e f S
     8   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 93
```

Oppgave 5a, for $x = 9$

```
Økning : Flytøkende vei
    10   K d e S
    20   K d c S
     5   K a c S
    10   K a b S
    20   K d e f S
     9   K a c e f S
    20   K a c d e f S
Max flyt fra K til S ble 94
```

Oppgave 5b

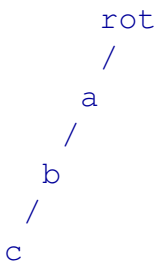
Flytkansellering: Når vi tidligere har lagt på flyt gjennom en kant, kan det hende at en senere flytøkende vei går gjennom samme kant i motsatt retning. Da legger vi ikke på flyt i motsatt retning, men kutter egentlig ut flyt som viste seg å gå i feil retning. Algoritmen kansellerer altså flyt som har blitt lagt på tidligere.

Oppgave 5c

Ved in-orden utskrift, behandles først nodens venstre barn. Så skrives nodens eget innhold, og til slutt behandles nodens høyre barn.

Ved postorden utskrift behandles først nodens venstre barn, så nodens høyre barn, og til slutt skrives nodens eget innhold.

Forskjellen ligger i behandling av høyre barn, så hvis postorden og in-orden skal gi samme resultat, må vi ha et tre hvor alle barn er til venstre:



Både in-orden og postorden vil skrive «c b a rot»