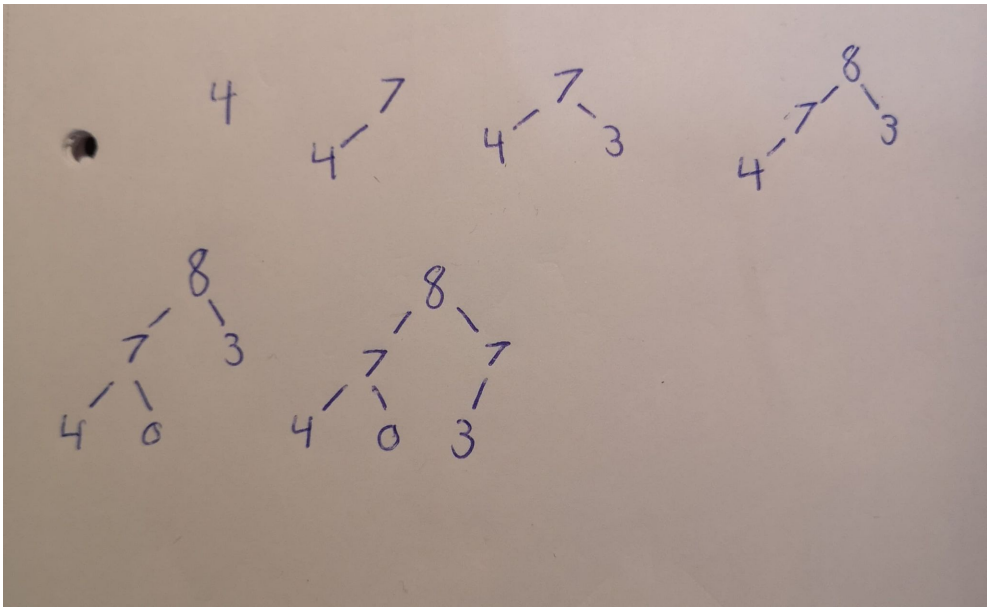
Eksamen H 2020 IDATT2101

1)

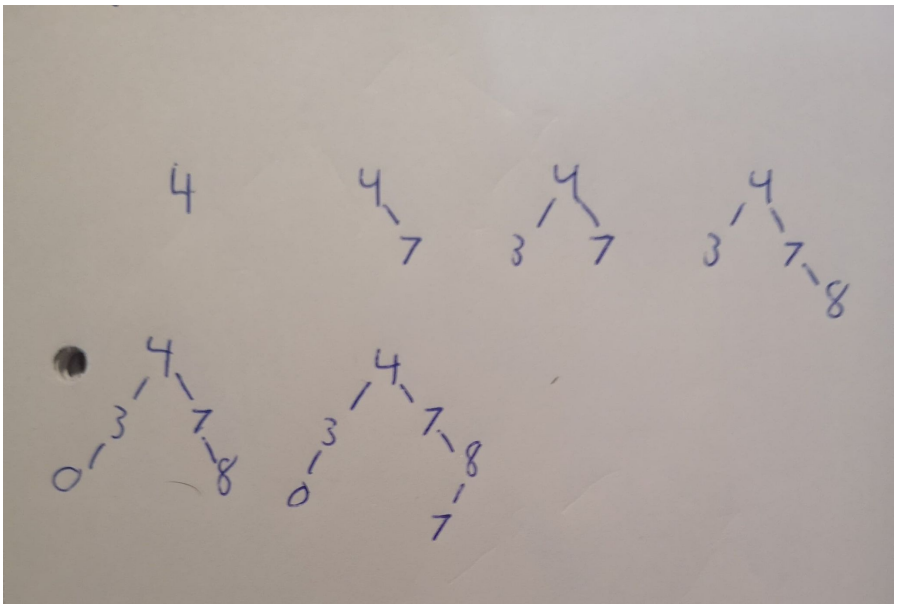
10081*47 = 473 807 (Tall som skal brukes i oppgavene under)

a)

Steg for steg sette inn nye tall i max-heap. Tar ikke med å legge inn fra bunn og så flytte opp siden dette da blir mange flere tegninger og blir også mere å sjekke gjennom for sensor.



b)
Steg for steg setter inn tallene i et binært tre



2)

a)

Siden det er 2 løkker hvor i den første går i til m, og i den andre går j (med samme verdi som m) til 0. Dette kan ses på som 2 løkker som går til m og kan derfor ses på som m^2. Svaret er da:

$\underline{T(m) \in \Theta(m^2)}$

b)

Her ser vi en if-statement som returnerer med en gang hvis n >= p. Pga dett vet vi at raskeste mulige gjennomkjøring blir 1. I løkken ser vi at løkken kjører så lenge i < p hvor i adderes med n for hver gjennomgang. Dette gir verste kjøretid p/n. Svaret blir da:

Kjøretid: $\underline{\Omega(1) \text{ eller } O(p/n)}$

c)

Her bruker jeg master metoden for å regne ut tidskompleksiteten for den rekursive metoden. Jeg får da verdiene a = 1, b = 2, og $c^k = n^1 = n$.

Fra dette ser vi at $a < b^k$.

Vi vet at $T(n) \in \Theta(n^k)$ hvis $a < b^k$ $(1 < 2)$.

Dette gir => $T(n) \in \Theta(n^1) = \underline{\Theta(n)}$

d)

Her bruker jeg master metoden for å regne ut tidskompleksiteten for den rekursive metoden. Jeg får da verdiene a = 2, b = 2, og $c^k = m^2$.

Fra dette ser vi at $a < b^k$ $(2 < 2^2)$.

Vi vet at $T(n) = \Theta(m^k)$ hvis $a < b^k$.

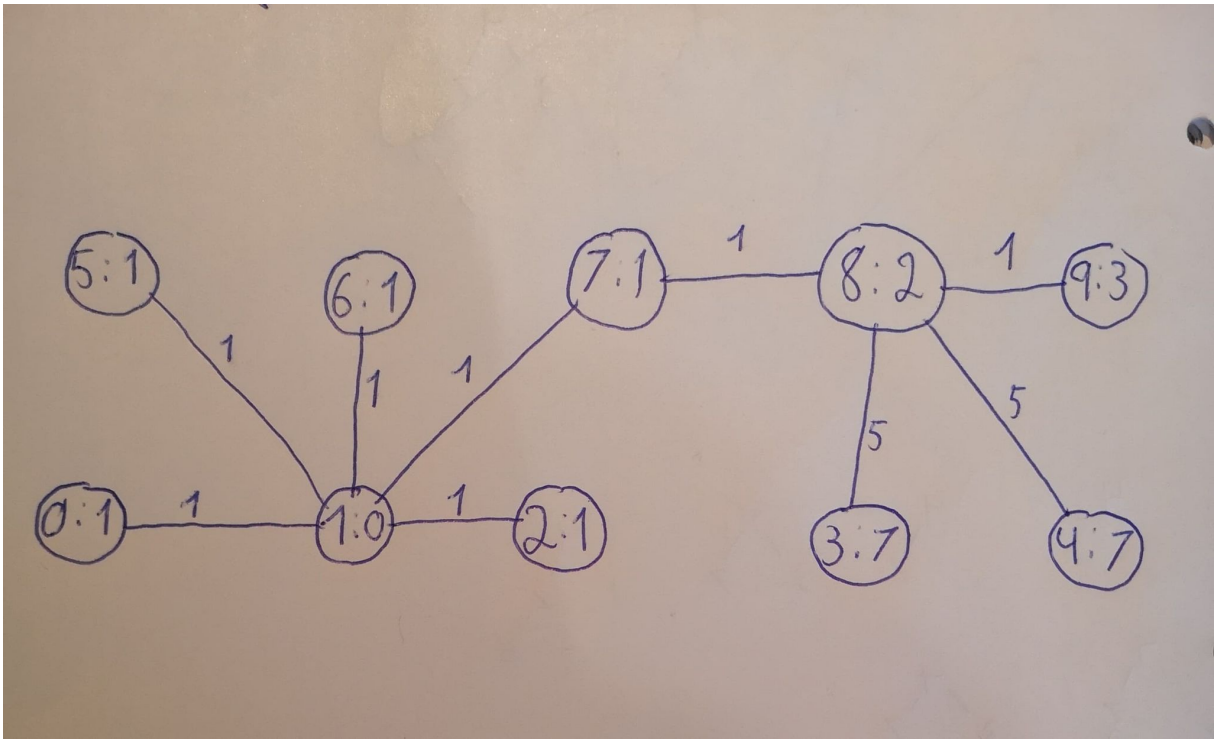
Dette gir => $\underline{T(n) \in \Theta(m^2)}$

3)

a)

Siste siffer i kandidatnummeret mitt er 1. Derfor skal jeg finne minste avstand fra node 1 til resten av nodene i grafen.

0 = 1 => 0, vekt = 1
2 = 1 => 2, vekt = 1
3 = 1 => 7 => 8 => 3, vekt = 1+1+5 = 7
4 = 1 => 7 => 8 => 4, vekt = 1+1+5 = 7
5 = 1 => 5, vekt = 1
6 = 1 => 6, vekt = 1
7 = 1 => 7, vekt = 1
8 = 1 => 7 => 8, vekt = 1+1 = 2
9 = 1 => 7 => 8 => 9, vekt = 1+1+1 = 3

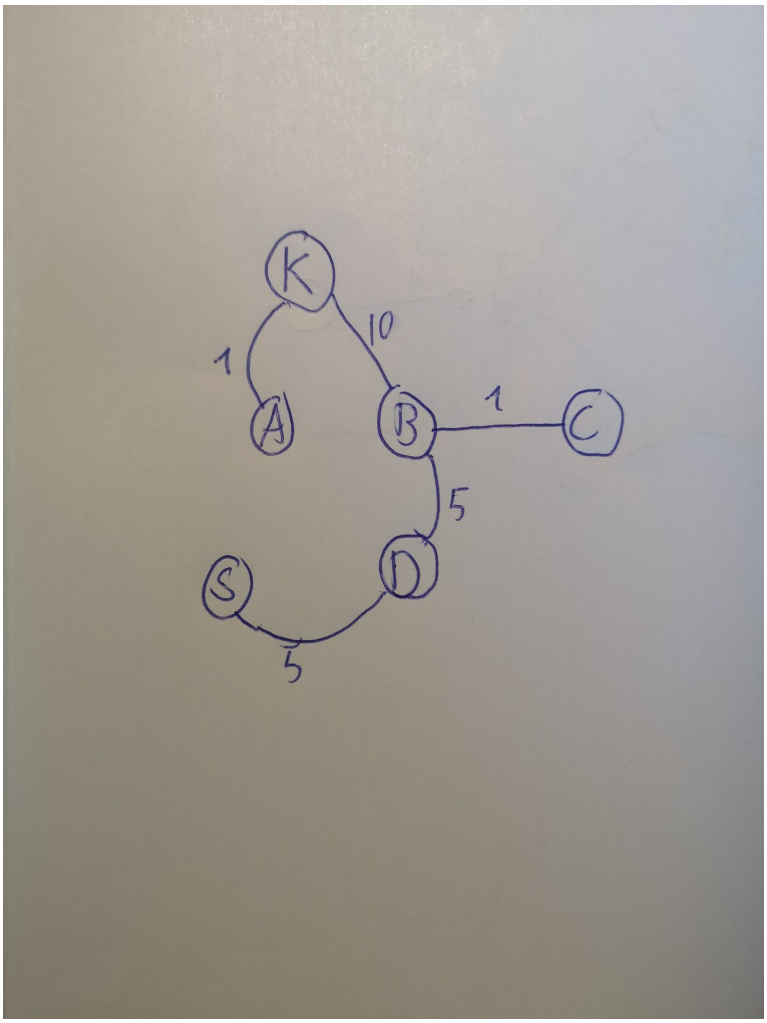


b)

Dijkstras algoritme fungerer ikke riktig på grafer med negative kanter fordi en negativ kant vil gå i motsatte retninger og besøke nøder som allerede har blitt besøkt og vi er ferdige med. Dette kan gi de allerede besøkte nodene kortere vei enn det de originalt hadde og blir derfor feil. DIjkstra fungerer med antagelsen at alle noder som er besøkt har funnet korteste vei. Når vi da går tilbake med en negativ vekted kant vil grunnmuren til Dijkstra være ødelagt.

4)

a)



Totalvekten til det minimale spenntreet er 22.

b)

De flytøkende veiene jeg brukte var:

K => A => S = 1 økning
K => B => A => S = 10 økning
K => C => D =>S = 10 økning
K => C => B => A => S = 1 økning
K => C => D => B => A => S = 5 økning

Maks flyt er da 1+10+10+1+5 = <u>27</u>

5)

a)

Et program/problem er i kompleksitetsklassen P hvis programmet/problemet kan løses i polynomisk tid ($O(n^k)$).

Et eksempel på et program som er i kompleksitetsklassen P er feks sorteringsalgoritmen «Selection sort» som har tidskompleksiteten $\Theta(n^2)$. Dette stammer fra sorteringsproblemet og blir derfor et eksempel på et slikt problem.

b)

Et problem er i kompleksitetsklassen NP hvis svaret til problemet kan sjekkes i polynomisk tid. Dette betyr altså at du enkelt kan sjekke svaret til et problem i polynomisk tid, men det er ikke nødvendigvis at man kan løse problemet i polynomisk tid. Dette ligner på kompleksitetsklassen P, men i NP kan løsningen sjekkes i polynomisk tid mens i P kan programmet løses i polynomisk tid.

c)

Siden man vil bruke størrelsen på tabellen til å gjøre restdivisjonen med (verdi%tabellstørrelse) vil altså restdivisjonen bli gjort som verdi%100. For tall som da slutter på 0 vil dette ikke bli effektivt. Generelt kan man si at for alle tall som slutter på 0 vil restverdien bli de to siste sifrene (feks 1340%100 og 1740%100 vil bege gi 40 som svar). Siden det er mange postnumre som har 0 som siste tall vil dette føre til mange kollisjoner og derfor mye mer arbeid som må gjøres med lenkede lister eller dobbel hashing.

Som en standard er det kjent at å gjøre restdivisjon med tabeller av formen 2^x eller 10^x er kjent for å være dårlige og bør derfor unngås. Et bedre alternativ er derfor å bruke tabellstørrelser av primtall (som ikke er for nærme en toerpotens) slik at man unngår disse problemene. Dette gir flere varierte hashede verdier og burde derfor i større grad unngå kollisjoner bedre.

For Kåre vil det derfor (i hans eksempel med tabellstørrelse på 100) være bedre å bruke hashing basert på multiplikasjon, eller endre tabellstørrelsen til et primtall hvis størrelsen på tabellen ikke var viktig.