Oblig 4

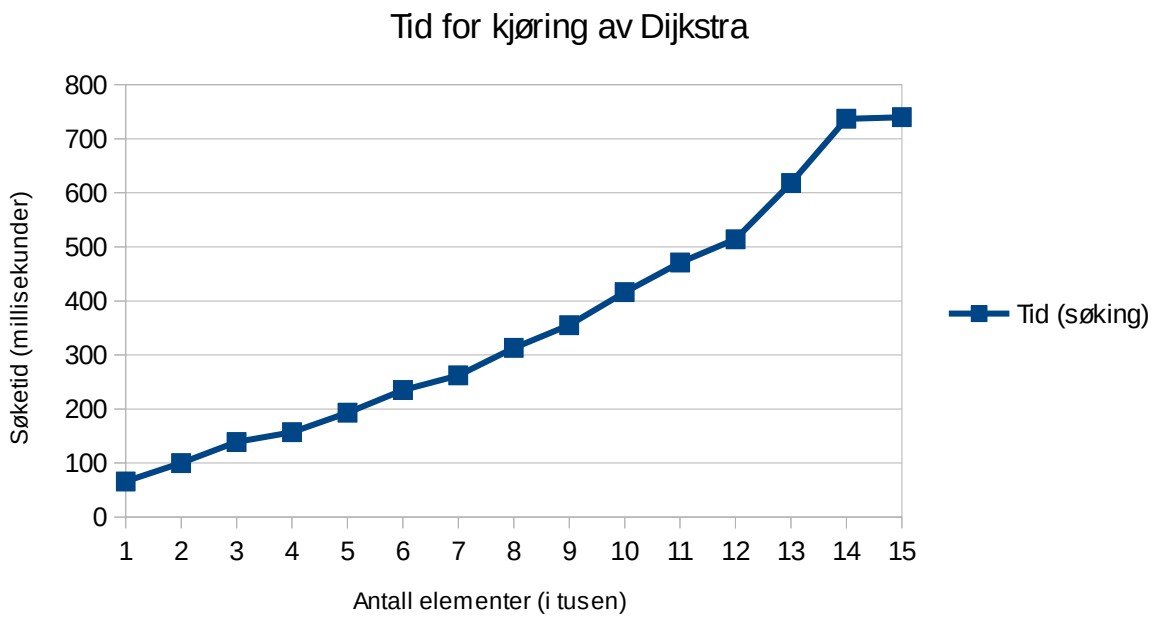
I oblig 4 skulle vi jobbe med Dijkstras algoritme for å finne korteste vei mellom 2 noder. Settet var delt inn I 2 oppgaver:
Oppgave 1)
Implementere Dijkstras algoritme og
Oppgave 2)
Søke etter flere noder og skrive ut stien mellom dem hvis det er en sti mellom disse nodene. Vi skulle også måle kjøretid og minnebruk.

For å løse oppgave 2, valgte jeg å lage «et nytt program» som istedenfor å lese fra fil og argument for å angi startnode, heller generer et tilfeldig antall noder og kanter, og ta utgangspunkt i node 0. Grunnen til at jeg hardkodet node 0 var for å unngå en mikroskopisk mulighet for at noden ikke skulle finnes.
I denne rapporten tok jeg utgangspunkt i kart med mellom 1000 og 15 000 noder. For hver kjøring øker jeg antall noder med 1000, slik at 1000 noder blir til 2000, 2000 blir til 3000, etc. Et problem som oppstod ved kjøring for fler enn 10 000 noder, var at heapstørrelsen var for liten, så det dukket opp en java.lang.OutOfMemoryError. Dette løste jeg ved å øke heapstørrelsen til 3 GB. Da antallet kom opp i 14 000, måtte størrelsen økes til hele 4 GB. Funnene listes i tabellen under: Bemerk at verdiene forandrer seg fra kjøring til kjøring (selv når jeg ber programmet opprette samme antall noder), så veldig sikker vitenskap er det nok ikke.

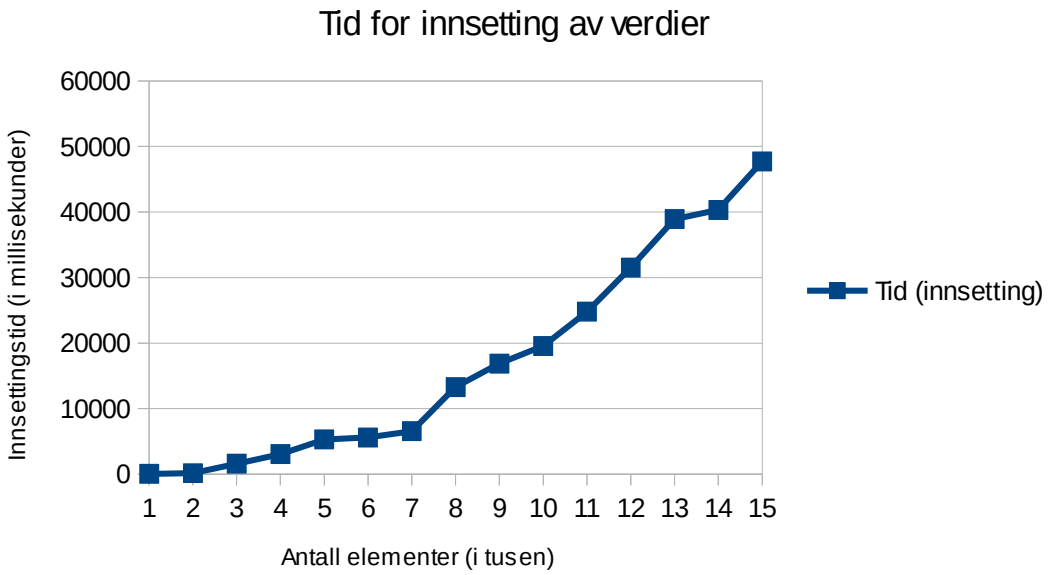
Tabell 1.1

Antall	Tid (innsetting)	Tid (søking)	Minne (innsetting)
1000	56 ms	66 ms	13,82 MB
2000	155 ms	100 ms	49,92 MB
3000	1 596 ms	139 ms	110,87 MB
4000	3 050 ms	157 ms	196,43 MB
5000	5 301 ms	193 ms	306,88 MB
6000	5 596 ms	235 ms	440,77 MB
7000	6 554 ms	262 ms	600,02 MB
8000	13 292 ms	313 ms	783,69 MB
9000	16 874 ms	355 ms	989,55 MB
10000	19 559 ms	416 ms	1 223,12 MB
11000	24 785 ms	471 ms	1 480,33 MB
12000	31 528 ms	514 ms	1 759,58 MB
13000	38 918 ms	618 ms	2 067,42 MB
14000	40 318 ms	737 ms	2395,72 MB
15000	45 875 ms	740 ms	2 749,83 MB
Gjennomsnitt	16 987 ms	347 ms	1 011.19 MB

I tabellen ovenfor er kolonnene «Tid (søking)» og «Minne (søking)» det mest interessante med tanke på selve kjøringen av Dijkstras algoritme.



Figur 1 viser en graf over hvor lang tid det tok å kjøre Dijkstras algoritme på mellom 1000 og 15000 noder



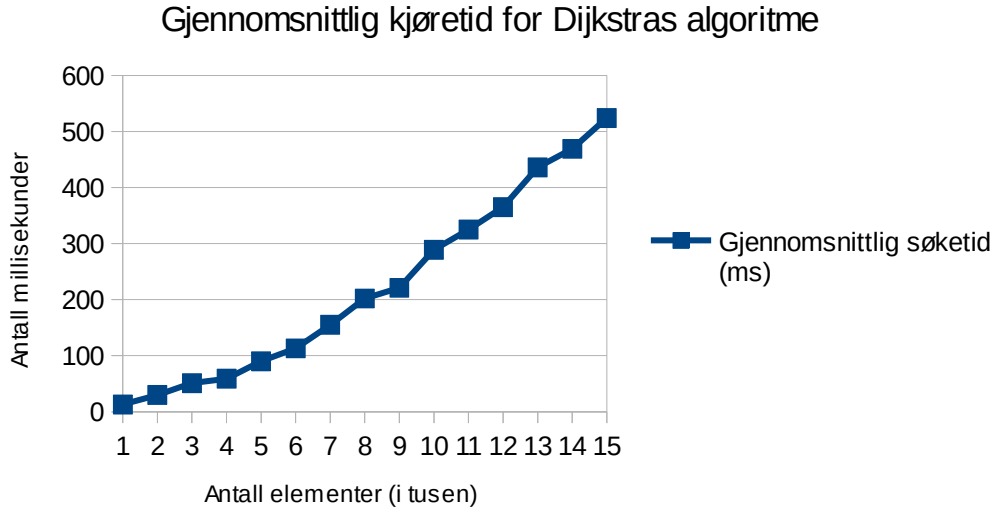
Figur 2 viser en graf over hvor lang tid det tok å sette inn mellom 1000 og 15000 noder. Merk at innsettingstiden i Y-aksen øker med 10000 for hver gang, mens «innsettingstiden» i Y-aksen på figur 1 bare øker med 100.

Oversikten over er basert på startnode 0. Ifølge oppgavetesten skulle vi teste ut ulike startnoder. Tabellen nedenfor viser en oversikt over tiden det tar å sette inn N elementer, den gjennomsnittlige tiden det tar å søke gjennom «kartet» fra 10 ulike noder, samt minnebruken for innsetting av data. Denne tabellen er ment for å løse den delen av oppgaven som sier at programmet skal testes for ulike antall noder samt varierende startnoder.

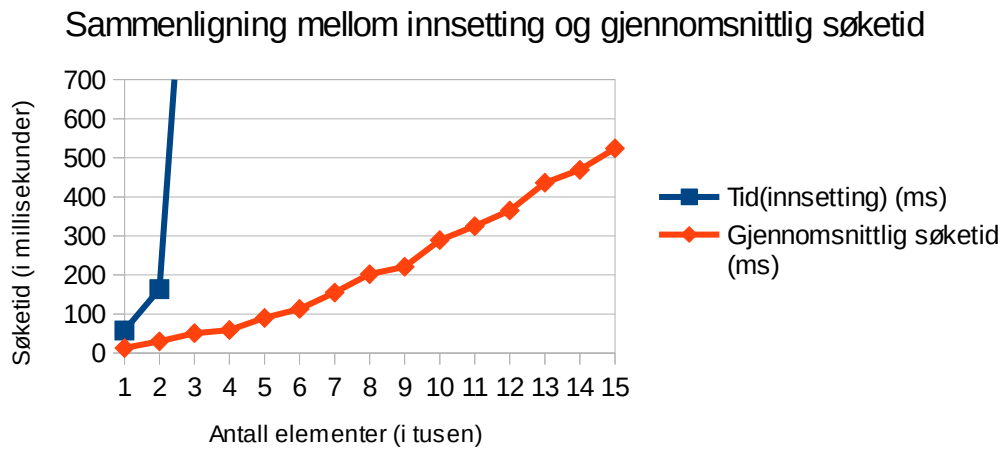
Tabell 1.2: Oversikt over tidforbruk av innsetting av noder og gjennomsnittlig søketid (Dijkstra) gjennom noder, samt minnebruk etter innsetting av elementer.

Elementer	Tid (innsetting)	Gjennomsnittlig søketid (10 noder)	Minne (innsetting)
1000	58 ms	13 ms	13,24 MB
2000	164 ms	30 ms	49,59 MB
3000	1427 ms	51 ms	111,05 MB
4000	2401 ms	59 ms	196,36 MB
5000	4233 ms	90 ms	306,47 MB
6000	4701 ms	113 ms	441,02 MB
7000	5357 ms	155 ms	600,35 MB
8000	8687 ms	202 ms	782,53 MB
9000	11876 ms	221 ms	990,23 MB
10000	14932 ms	289 ms	1222,85 MB
11000	21764 ms	325 ms	1478,35 MB
12000	21007 ms	365 ms	1760,87 MB
13000	27732 ms	436 ms	2067,05 MB
14000	32617 ms	469 ms	2396,02 MB
15000	38443 ms	524 ms	2748,25 MB

Figuren nedenfor viser dataene fra den gjennomsnittlige kjøretiden ovenfor i en graf.

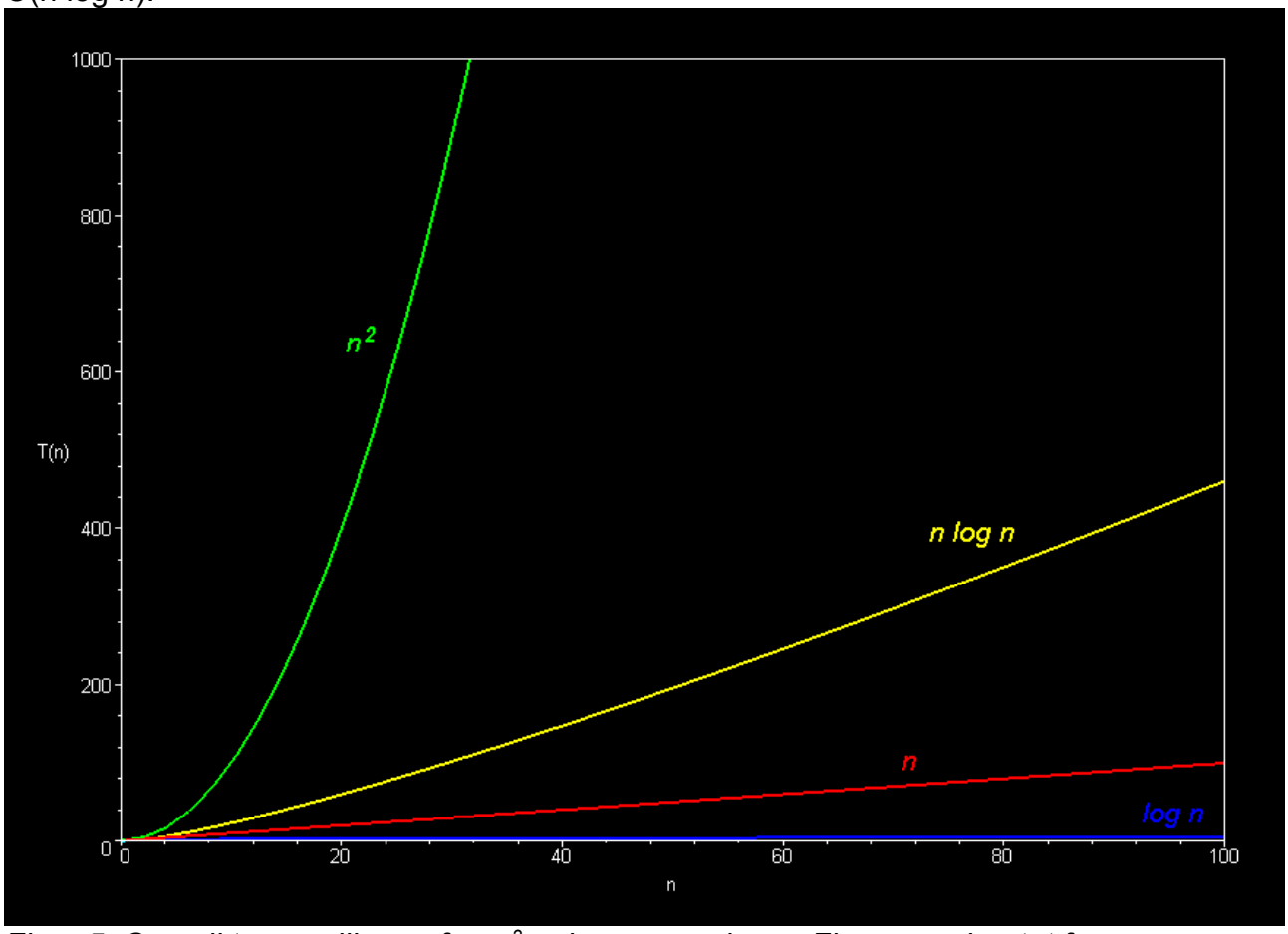


Figur 3: en graf som viser den gjennomsnittlige kjøretiden for Dijkstras algoritme for ,tilfeldige startnoder.



Figur 4 viser en sammenligning mellom kompleksiteten av innsetting av data $O(n^2)$ og den gjennomsnittlige kompleksiteten av av kjøring av Dijkstras algoritme.

Ifølge teorien er Dijkstras algoritme en O(E log V)-algoritme. Funnene jeg har viser at dette ser ut til å stemme. Samrmenlignet med bildet nedenfor (figur 5), mener jeg, ut ifra grafene jeg har produsert basert på dataene jeg har kommet frem til (se figur 1, 3 og 4), at kompleksiteten i Dijkstras algorirtme er den samme som i teorien, altså O(E log V), eller O(n log n).



Figur 5: Oversikt over ulike grafer: n^2, n log n, n og log n. Figuren er hentet fra http://science.slc.edu/~jmarshall/courses/2002/spring/cs50/BigO/.