Obligatorisk innlevering 1 – Oppgave 2, Erik, Jørgen, Birger, Martin – Gruppe 2 Førde

- Oppgave 2 a)

  Hva er størrelsesorden uttrykt i O-notasjon (dvs. vi behøver ikke finne c og N) for algoritmen når vekstfunksjonene er gitt som;

  i) 4n2 + 50n – 10
  i utrykk) O(n^2)

  ii) 10n + 4 log2 n + 30
  ii utrykk) O(n)

  iii) 13n3 – 22n2 + 50 n + 20
  iii uttrykk) O(n^3)

  iv) 35 + 13 log2 n
  iv uttrykk) O(log n)

- Oppgave 2 b)

  Gitt følgende algoritme:

```
for (int i = 1; i <= n; i++){
  for (int j = 1; j <= n; j++) {
    sum = sum + 1;
      }
   }
```

  Finn antall tilordninger (=) for algoritmen og effektiviteten uttrykt i O-notasjon. Begrunn svaret. Vi ser kun på løkkekroppen når vi analyserer løkker!

  Effektiviteten uttrykt: O(n^2)
  Antall tilordninger: n^2

  Begrunnelse: Antall tilordninger i denne løkken vil være det samme som notasjonens uttrykk, fordi hver gang ytre løkke kjører, vil indre løkke kjøre n ganger.

- Oppgave 2 c)

Algoritme A:

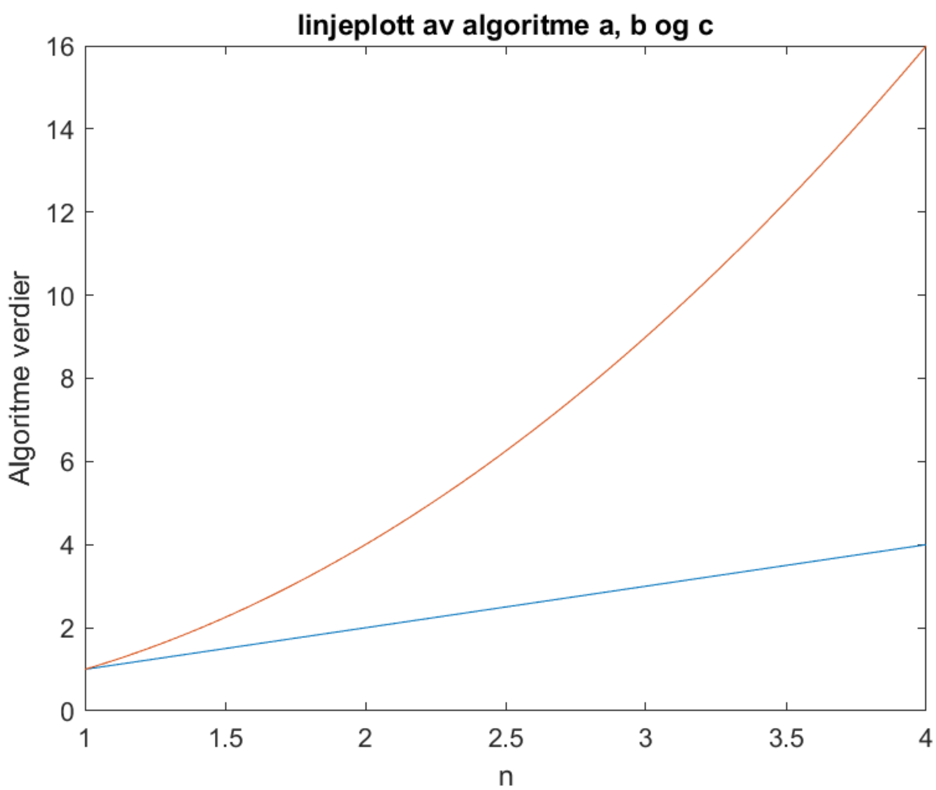
Uttrykk:	O(n)
Tilordninger:	1+n
Operasjoner:	2 (utgangspunkt i at vi ikke tar med løkkeindeks)
Addisjoner:	1
Multiplikasjoner:	0
Divisjoner:	0

Algoritme B:

Uttrykk:	O(n^2)
Tilordninger:	(n*(n + 1)/2)+1
Operasjoner:	2 (utgangspunkt i at vi ikke tar med løkkeindeks)
Addisjoner:	1
Multiplikasjoner:	0
Divisjoner:	0

Algoritme C:

Uttrykk:	O(1)
Tilordninger:	1
Operasjoner:	1
Addisjoner:	1
Multiplikasjoner:	1
Divisjoner:	1



Oppg 2 d)

Obligatorisk innlevering 1 – Oppgave 2, Erik, Jørgen, Birger, Martin – Gruppe 2 Førde

t(n)	t(10^6)/10^6
log2 n	19.93 sekunder
n	1,000,000 sekunder
n log2 n	19931568.57 sekunder
n^2	11,574,074 dager
n^3	31,688,764,615.4 år

Oppg 2 e)

Effektivitet: O(n^2)

Begrunnelse:

Den ytterste forløkka itererer fra indeks 0 til n-2, og for hver skritt blir hele tabellen iterert igjennom av den indre forløkka som går fra 0 til n-1.

Antall sammenligninger i verste tilfellet blir altså at den teller hele tabellen helt til n, dersom den ikke finner noe duplikat.

Antall sammenligninger blir:
n*(n + 1)/2

Oppg 2f)

i) Effektivitet: O(n^3)
ii) Effektivitet: log n
iii) Effektivitet: n log n
iv) Effektivitet: O(n)

Det er algoritme t1 (8n+4n^3) som er minst effektiv og mest tidkrevende.
Denne algoritmen er kubisk, og vokser eksponensielt.

Algoritme t2 (10 log n + 20) er algoritmen som er minst tidkrevende og som vokser minst av disse, og vil derfor være mest effektiv i tilfellet hvor n er veldig stor. Den har lavest tidskompleksitet.

Obligatorisk innlevering 1 – Oppgave 2, Erik, Jørgen, Birger, Martin – Gruppe 2 Førde

Oppg 2g)

```
tid(100000000L);
```

Her varierte resultatet mellom ca. 29 til 36

```
tid(1000000000L);
```

Her varierte resultatet mellom ca. 294 til 306

```
tid(10000000000L);
```

Her varierte resultatet mellom ca. 3046 til 3076

Resultatene stemmer slik at for hver 0 som blir lagt til i tid, så øker også resultatene med faktor på n, men med et større slingringsmoment som er like stor relativt til n.