Oppgave 2

a)

i) O(n^2) er størrelsesorden på $4n^2 + 50n - 10$
ii) O(n) er størrelsesorden på $10n + 4\ log2\ n + 30$
iii) O(n^3) er størrelsesorden på $13n3 - 22n2 + 50\ n + 20$
iiii) O(log2n) er størrelsesorden på $35 + 13\ log2\ n$

b) Den ytterste løkken har størrelsesorden O(n) og den innerste er O(n^2), antall tilordninger er bestemt av O(n) ganger med n antall ganger. Den innerste løkken kjøres *n* ganger og den andre gangen kjøres den sum= sum + 1. Det er ikke konstanten som er det viktig men det er selveste vekstfunksjonen som vi kan se så er effektiviteten i løkken O(n^2) .

c)

	Algoritme A	*Algoritme B*	*Algoritme C*
Tilordninger =	*n*	$1 + 2 + \ldots + n = n(n+1)/2$	
Addisjoner +	0		
*Multiplikasjon **			
Divisjon /			
Totalt antall operasjoner	2		

d)

t(n)	*t(10^6)/6*
log2 1000 || ln 10000/ln 2	*19,3 sekunder*
n	*1 sek på 10^6 per*
n log2 n	*19 300000 sek per*
n^2	11 574 074 dager
n^3	$3.16887646 \times 10^{10}$ years p

e)

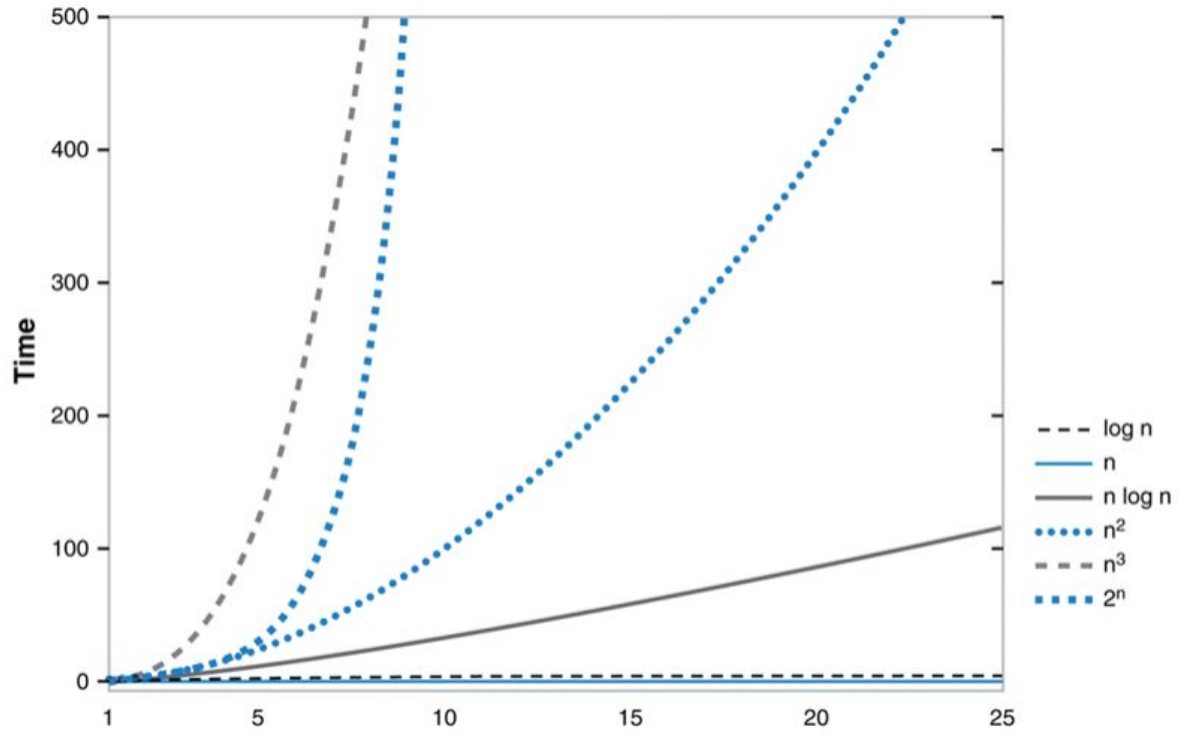
```
Boolean harDuplikat (int tabell[], int n) {
// n er antall elementer
for (int indeks= 0;  indeks <=2; indeks++) {
for(int igjen=indeks+1; igjen <=n-1; igjen++) {
if(tabell[indeks]==tabell[igjen])
return true
}
}
return false;
}
```

f)

i) O-notasjonen er O(n^3)
ii) O-notasjonen er log2n
iii) O-notasjonen er O(n)
iiii) O-notasjonen er O(log2n)

Fra størst til minst

Den mest effektive algoritme er n^3 som har den største veksten, den andre er n log2n som du kan se på grafen og den siste er log2n som er den minst effektive algoritmen.



Oppgave 3

Tidskomplekisteten er O(n) fordi at den går gjennom cdarkivet engang.

Oppgave 4

Metoden tester om stabelen er tom ved å putte in et element og ta det ut.
Dette skjer ved å catche en EmptyCollectionException med meldingen,
"Pop failed unexpectedly 'Ekstra info fra getMessage() metoden'".
Under er det enda en metode som sjekker om stabelen er faktisk tom ved ta ut et element.
Deretter sende ut et EmptyCollectionException hvis den er tom.