3a)

1. =
2. = , sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.
3. = , sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.
4. = , sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.

3b)

Vi ser på

sum = 0;  
for (int i = n; i > 1; i = i/2) {  
sum = sum + i;}

der tilordningane er *sum=0 , i=n, i>1, i= 1/2* \*i og sum=sum+1, som gir oss 5 tilordningar. Effektiviteten til desse er:

.

3c) Vi ser på  
sum = 0;  
for (int i = 1; i <= n; i++) {  
for (int j = 1; j <= n; j = j \* 2) {  
sum += i \* j;  
} }

der tilordningane er *sum=0*, , , , , og *sum+=i\*j*, som gir oss 7 tilordningar. Effektiviteten til desse er:

sum=

, sidan og er heiltal.

sum+=, sidan og er heiltal.

3d)

P sirkel = sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.

A sirkel = sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.

3e)

Vi ser på

boolean harDuplikat(int tabell[], int n) {  
for (int indeks = 0; indeks <= n - 2; indeks++) {  
for (int igjen = indeks + 1; igjen <= n - 1; igjen++) {  
if (tabell[indeks] == tabell[igjen]){  
return true;  
} } }  
return false;  
}

max n=2147483647 (integer limit)

Hvert element vil sammenligne med hvert element som står på venstre side av bordet. Det betyr at hver indeks på listen skal ha antall sammenligninger som den forrige indeksen har, men med -1 sammenligning.

=> ∑ = 5.802646038×10^18

=>

3f)

i) sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.

ii ) sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.

iii) sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.

iv) sidan den operasjonen inneheld den største vekstgraden av alle dei andre operasjonane.

=> ii < iv < iii < i

3g)

Vi ser på

public static void tid(long n) {  
long k = 0;  
for (long i = 1; i <= n; i++) {  
k = k + 5;  
}  
}

For (long i = 1; i <= n; i++) betyr at vår sum virker n ganger og hver gang vi legg til + 5 på vår variabel, så er vår sum Vektfunksjonen er

Vektfunksjonen er en linær funksjon så hvis vi utfører , må , tiden øke med 10 (fordi vår n øke med 10 også). I praksis er det ikke en perfect linær funksjon fordi andre faktorer kan påvirke resultatene (fe: CPU, load av systemet). Med flere ganger skal vi få mer nøyaktige resultater.