DAT102 - Obligatorisk innlevering 1

Gruppemedlemmer:
Stephen Neba Fuh, Tord Johan Melheim,
Ebubekir Siddik Yuksel, Casper Eide Özdemir-Børretzen

Skjermbilder fra kjøring av Java program:

```
[ 01 ] [ 02 ]
[WARNING] Some problems were encountered while building the effective model for no.hvl.data102.filmarkiv:filmark
[WARNING] 'dependencies.dependency.version' for org.junit.jupiter:junit-jupiter:jar is either LATEST or RELEASE
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----------------< no.hvl.data102.filmarkiv:filmarkiv >-----------------
[INFO] Building filmarkiv 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.5.1:java (default-cli) @ filmarkiv ---

------------------------------------------------------------------------
FILMARKIV (Tabell)
------------------------------------------------------------------------
Ny film lagt til: [Film] nr: 0, tittel: A Poet, år: 2025, sjanger: DRAMA, filmselskap: Ocúltimo, produsent: Juan
Ny film lagt til: [Film] nr: 1, tittel: Harvest, år: 2024, sjanger: DRAMA, filmselskap: ARTE France Cinéma, prod
Ny film lagt til: [Film] nr: 2, tittel: Samsara, år: 2023, sjanger: DRAMA, filmselskap: Filmin, produsent: Leire
Ny film lagt til: [Film] nr: 3, tittel: The Apple, år: 1998, sjanger: DRAMA, filmselskap: Makhmalbaf Film House
Ny film lagt til: [Film] nr: 4, tittel: Life, and Nothing More…, år: 1992, sjanger: DRAMA, filmselskap: Kanoon,
Ny film lagt til: [Film] nr: 5, tittel: Kuroneko, år: 1968, sjanger: HORROR, filmselskap: Nihon Eiga Shinsha, pr
Ny film lagt til: [Film] nr: 6, tittel: To Be or Not to Be, år: 1942, sjanger: KOMEDIE, filmselskap: Romaine Fil

Filmarkiv søk etter film med tittel som inneholder "not":
[Film] nr: 4, tittel: Life, and Nothing More…, år: 1992, sjanger: DRAMA, filmselskap: Kanoon, produsent: Ali Rez
[Film] nr: 6, tittel: To Be or Not to Be, år: 1942, sjanger: KOMEDIE, filmselskap: Romaine Film Corporation, pro

Filmarkiv søk etter film med produsent som inneholder "en":
[Film] nr: 0, tittel: A Poet, år: 2025, sjanger: DRAMA, filmselskap: Ocúltimo, produsent: Juan Sarmiento G.
[Film] nr: 1, tittel: Harvest, år: 2024, sjanger: DRAMA, filmselskap: ARTE France Cinéma, produsent: Viola Fügen

Statistikk for filmarkiv:
Antall filmer: 7
Antall filmer i sjanger DRAMA: 5
Antall filmer i sjanger DOKUMENTAR: 0
Antall filmer i sjanger HORROR: 1
Antall filmer i sjanger KOMEDIE: 1

Legg til ny film:
Nr: 7
Tittel: No Other Land
År: 2024
Sjanger: dokumentar
Filmselskap: Antipode
Produsent: Bård Kjøge Rønning
Ny film lagt til: [Film] nr: 7, tittel: No Other Land, år: 2024, sjanger: DOKUMENTAR, filmselskap: Antipode, pro

------------------------------------------------------------------------
FILMARKIV (Lenket liste)
------------------------------------------------------------------------
Ny film lagt til: [Film] nr: 0, tittel: A Poet, år: 2025, sjanger: DRAMA, filmselskap: Ocúltimo, produsent: Juan
Ny film lagt til: [Film] nr: 1, tittel: Harvest, år: 2024, sjanger: DRAMA, filmselskap: ARTE France Cinéma, prod
Ny film lagt til: [Film] nr: 2, tittel: Samsara, år: 2023, sjanger: DRAMA, filmselskap: Filmin, produsent: Leire
Ny film lagt til: [Film] nr: 3, tittel: The Apple, år: 1998, sjanger: DRAMA, filmselskap: Makhmalbaf Film House
Ny film lagt til: [Film] nr: 4, tittel: Life, and Nothing More…, år: 1992, sjanger: DRAMA, filmselskap: Kanoon,
Ny film lagt til: [Film] nr: 5, tittel: Kuroneko, år: 1968, sjanger: HORROR, filmselskap: Nihon Eiga Shinsha, pr
Ny film lagt til: [Film] nr: 6, tittel: To Be or Not to Be, år: 1942, sjanger: KOMEDIE, filmselskap: Romaine Fil

Filmarkiv søk etter film med tittel som inneholder "not":
[Film] nr: 4, tittel: Life, and Nothing More…, år: 1992, sjanger: DRAMA, filmselskap: Kanoon, produsent: Ali Rez
[Film] nr: 6, tittel: To Be or Not to Be, år: 1942, sjanger: KOMEDIE, filmselskap: Romaine Film Corporation, pro

Filmarkiv søk etter film med produsent som inneholder "en":
[Film] nr: 0, tittel: A Poet, år: 2025, sjanger: DRAMA, filmselskap: Ocúltimo, produsent: Juan Sarmiento G.
[Film] nr: 1, tittel: Harvest, år: 2024, sjanger: DRAMA, filmselskap: ARTE France Cinéma, produsent: Viola Fügen

Statistikk for filmarkiv:
Antall filmer: 7
Antall filmer i sjanger DRAMA: 5
Antall filmer i sjanger DOKUMENTAR: 0
Antall filmer i sjanger HORROR: 1
Antall filmer i sjanger KOMEDIE: 1

Legg til ny film:
Nr: 99
Tittel: All We Imagine as Light
År: 2024
Sjanger: drama
Filmselskap: Pulpa Film
Produsent: Govinda Van Maele
```

```
~/git/studies/01-dataing/s2/dat102/oblig1/oppg3 › make
make build
make[1]: Entering directory '/home/lzzrhx/git/studies/01-dataing/s2/dat102/oblig1/oppg3'
javac -encoding utf8 *.java
make[1]: Leaving directory '/home/lzzrhx/git/studies/01-dataing/s2/dat102/oblig1/oppg3'
make run
make[1]: Entering directory '/home/lzzrhx/git/studies/01-dataing/s2/dat102/oblig1/oppg3'
java Program

 - - - - - - - - - - - - - - - - - - - - - - - -
 OPPGAVE 3
 - - - - - - - - - - - - - - - - - - - - - - - -

Oppgave 3b:
n = 10  -->  8
n = 100  -->  14
n = 1000  -->  20
n = 10000  -->  28
n = 100000  -->  34

Oppgave 3c:
n = 10  -->  100
n = 100  -->  1600
n = 1000  -->  22000
n = 10000  -->  300000
n = 100000  -->  3600000

Oppgave 3e:
n = 10  -->  45
n = 100  -->  4950
n = 1000  -->  499500
n = 10000  -->  49995000
n = 100000  -->  704982704

Oppgave 3g:
n = 10000000  -->  5.2 ms
n = 100000000  -->  46.7 ms
n = 1000000000  -->  467.0 ms

make[1]: Leaving directory '/home/lzzrhx/git/studies/01-dataing/s2/dat102/oblig1/oppg3'
~/git/studies/01-dataing/s2/dat102/oblig1/oppg3 ›
```

Oppgave 3

a)

$$
\begin{aligned}
&i.\ 4n^2 + 50n - 10 && \rightarrow\ O(n^2)\\
&ii.\ 10n + 4\log_2 n + 30 && \rightarrow\ O(n)\\
&iii.\ 13n^3 + 22n^2 + 50n + 20 && \rightarrow\ O(n^3)\\
&iv.\ 35 + 13\log_2 n && \rightarrow\ O(\log n)
\end{aligned}
$$

b) Vi teller antall tilordninger (=), antall ganger (=) brukes.

1) Utenfor løkken: 1 tilordning `sum = 0;`

2) I "for"-løkken: 1 tilordning `int i = n;`

3) I "for"-løkken: 1 tilordning `i = i/2;`

4) I "for"-løkken: 1 tilordning `sum = sum + i;`

$\Rightarrow$ Total antall iterasjoner kan bli beskrevet som "$\log_2(n)$" ganger.

Inn i løkken "`sum = 0`" = 1, "`i = n`" = $1 \rightarrow 2 \cdot log_2(n)$

$\Rightarrow$ Vi har 2 iterasjoner til, dermed $2 + 2log_2(n)$

$\Rightarrow$ Vi fokuserer bare på konstanter, og ser vekk fra det "raskeste", "log(n)", er den minst raske, så $O\log(n)$.

c) Teller hver gang (=) brukes.

1) "`sum = 0;`" $\rightarrow$ 1. Tilordning.

2) "`int i = 1;`" $\rightarrow$ 1. Tilordning.

3) "`i++`" $\rightarrow$ Skjer n ganger, pga skjer hele tiden.

4) "`int j = 1`" $\rightarrow$ Avhengig av ytre-løkke, så den skjer en gang per ytre iterasjon. n ganger.

5) "`j = j * 2`" $\rightarrow$ Avhengig av ytre-løkke, skjer $\log_2(n)$ ganger per ytre iterasjon pga. multiplikasjon. $n \cdot \log_2(n)$.

6) "`sum = sum + i \cdot j`" $\rightarrow$ $n \cdot \log_2(n)$ ganger.

$\Rightarrow$ Summerer alle tilordningene.

$1 + 1 + n + n + 2n \cdot \log_2(n) + n \cdot \log_2(n) = 2 + 2n + 2n\log_2(n)$.

Konstanter og lavere ordens ledd fjernes: $O(n \log n)$

d)

$$
\begin{aligned}
&\pi r^2 \ \rightarrow\ O(n^2)\\
&2\pi r \ \rightarrow\ O(n)
\end{aligned}
$$

e) Den ytre løkken har en tellevariabel "indeks" som går fra 0 til $n-2$ og inkrementeres i hver iterasjon av løkken. Den ytre løkken kjøres derfor $n-1$ ganger. Den indre løkken har en tellevariabel "igjen" som går fra indeks + 1 til $n-1$. Antall ganger den indre løkken kjøres er derfor avhengig av "indeks" verdien. Ved indeks = 0 må verdien som "indeks" refererer til sammenlignes med alle andre verdier i tabellen og den indre løkken kjøres $n-1$ ganger. Ved indeks = $n-2$ har alle verdier i tabellen allerde blitt sammenlignet, bortsett fra den siste og nest-siste verdien, så den indre løkken kjøres bare 1 gang for å sammenligne de to siste verdiene.

Det verste tilfelle for algoritmen er når tabellen ikke inneholder duplikater som fører til at alle verdier i tabellen må sammenlignes.

Eksempel ved $n = 10$:

indeks:	Ant. ganger indre løkke kjøres:	Totalt antall sammenligninger:
0	$(10-1) = 9$	9
1	$(10-2) = 8$	17
2	$(10-3) = 7$	24
3	$(10-4) = 6$	30
4	$(10-5) = 5$	35
5	$(10-6) = 4$	39
6	$(10-7) = 3$	42
7	$(10-8) = 2$	44
8	$(10-9) = 1$	45

Ved å lage et enkelt Java program kan antall sammenligner beregnes for andre n verdier:

```
public class Program {
    public static void main(String[] args) {
        System.out.println(" - - - - - - - - - - - - - - - - - - - - - - - - ");
        System.out.println(" OPPGAVE 3e ");
        System.out.println(" - - - - - - - - - - - - - - - - - - - - - - - - ");
        int[] nTab = new int[] { 10, 100, 1_000, 10_000, 100_000};
        for (int n : nTab) {
            print(n,oppg3e(n));
        }
    }

    private static void print(int n, int k) {
        System.out.printf("n = %d --> %d%n", n, k);
    }

    private static int oppg3e(int n) {
        int[] tab = new int[n];
        for (int i = 0; i < n; i++) {
            tab[i] = i;
        }
        // boolean harDuplikat( ... )
        // v = indeks for venstre tall i sammenligning
        // h = indeks for høyre tall i sammenligning
        // if (tab[v] == tab[h]) return true
        int c = 0; // c brukes for å telle antall sammenligninger
        for (int v = 0; v <= n - 2; v++) {
            for (int h = v + 1; h <= n - 1; h++) {
                c++; // c += 1
                if (tab[v] == tab[h]) {
                    return c; // return true her (to like tall funnet)
                }
            }
        }
        return c; // return false her (ingen like tall funnet)
    }

}
```

Dette gir følgende resultat:

```
 - - - - - - - - - - - - - - - - - - - - - - - -
 OPPGAVE 3e
 - - - - - - - - - - - - - - - - - - - - - - - -
n = 10 --> 45
n = 100 --> 4950
n = 1000 --> 499500
n = 10000 --> 49995000
n = 100000 --> 704982704
```

Antall sammenligninger i forhold til n kan beskrives som en aritmetisk rekke $s_m = \frac{m \cdot (a_1 + a_m)}{2}$. Der m er antall ganger den ytre løkken kjøres $(n-1)$. a_1 er første tall i rekken $(n-1)$. Og a_n er siste tall i rekken (1). Det gir $s_{n-1} = \frac{(n-1) \cdot ((n-1)+1)}{2} = \frac{n^2-n}{2}$, som i O-notasjon er $O(n^2)$.

f)

i. $t_1(n) = 8n + 4n^3 \rightarrow O(n^3)$
ii. $t_2(n) = 10\log_2 n + 20 \rightarrow O(\log n)$
iii. $t_3(n) = 20n + 2n\log_2 n + 11 \rightarrow O(n\log n)$
iv. $t_4(n) = 4\log_2 n + 2n \rightarrow O(2^n)$

Rangert etter effektivitet:

1) $O(\log n)$ $[t_2]$
2) $O(n\log n)$ $[t_3]$
3) $O(n^3)$ $[t_1]$
4) $O(2^n)$ $[t_4]$

g) Hvis vi bruker antall tilordninger som måleenhet for algoritmen slik som tidligere, ser vi at det alltid gjennomføres to tilordninger for `long k = 0` og `long i = 1`. For-løkken kjøres $n-1$ ganger, og for hver iterasjon gjennomføres to tilordninger, `i++` og `k = k + 5`.

Antall tilordninger, for $n > 0 \rightarrow 2 + 2(n-1) = 2n$. Som er $O(n)$ i O-notasjon.

Ved å lage et enkelt Java-program kan vi måle tiden for $n = 10^7, 10^8$ og 10^9:

```
public class Program {
    public static void main(String[] args) {
        System.out.println(" - - - - - - - - - - - - - - - - - - - - - - - - ");
        System.out.println(" OPPGAVE 3g ");
        System.out.println(" - - - - - - - - - - - - - - - - - - - - - - - - ");
        long[] nTab = new long[] { 10_000_000, 100_000_000, 1_000_000_000};
        for (long n : nTab) {
            int ant = 20;
            long res = 0L;
            for (int i = 0; i < ant; i++) {
                long start = System.currentTimeMillis();
                long k = 0;
                for (long j = 1; j <= n; j++) {
                    k = k + 5;
                }
                res += System.currentTimeMillis() - start;
            }
            System.out.printf("n = %d --> %.1f ms%n", n, (double)res / (double)ant);
        }
    }
}
```

Dette gir følgende resultat:

```
 - - - - - - - - - - - - - - - - - - - - - - - -
 OPPGAVE 3g
 - - - - - - - - - - - - - - - - - - - - - - - -
n = 10000000 --> 5.0 ms
n = 100000000 --> 47.0 ms
n = 1000000000 --> 465.7 ms
```

Resultatene viser en lineær økning i tiden det tar å gjennomføre metoden, som stemmer overens med hva vi hadde forventet ved $O(n)$.

Vekstfunksjonen i dette tilfelle er $T(n) = cn$, der $c \approx 0.0000005\ ms$. c vil variere avhengig av faktorer som hvilken maskinvare og operativsystem programmet kjører på. I tillegg til en rekke andre faktorer slik som om det er andre prossesser som kjører samtidig, hvilken versjon av Java som brukes og om det skal gjennomføres søppeltømming underveis.