Uppgift 1

Simon Moradbakti

HT 2024

1 Introduktion

Detta är den första rapporten för kursen ID1021 som ges på KTH. Syftet med kursen är att få en djupare förståelse för enkla algoritmer och datastrukturer. I första uppgiften skulle olika tider mätas, förklaras och presenteras från givna kodrader. Koderna innehåller 3 olika operationer, "Random access", "Search" och "Duplicates". Ett polynom skulle också tas fram för de 2 sistnämnda operationerna som beskriver dem så nära som möjligt. Denna rapport skrivs i Overleaf för att det ska ges en möjlighet att kunna bekanta sig med Latex.

2 Körtid - Random access

Våran första tilldelade uppgift var att mäta kör tiden för att köra en operation. Det skulle också tas reda på hur exakt denna mätning kan utföras. För att detta skulle kunna göras gavs följande kod:

```
for (int i = 0; i < 10; i++) {
long n0 = System.nanoTime();
long n1 = System.nanoTime();
System.out.println(" resolution " + (n1 - n0) + "
    nanoseconds");
}
```

Efter att koden körts fick vi följande resultat: första körningen gav 350 ns. Andra körningen gav 420 ns. Tredje körningen gav 510 ns. en slutsats drogs att det fanns en felmarginal på ungefär 50-100 ns. För att få ett så bra resultat som möjligt var det en möjlighet att ta fram ett medelvärde. Ett medelvärde blir mer representativt för hur lång tid en operation tar att avläsas desto mer data man har, men för att det ska fungera måste vi använda oss av operationen "Access random" för att förhindra att Cashing ska påverka resultatet.

Resultatet när koden:

```
int[] given = {0,1,2,3,4,5,6,7,8,9};
```

```
Random rnd = new Random();
int sum = 0;
long t0 = System.nanoTime();
for (int i = 0; i < 1000; i++) {
sum += given[rnd.nextInt(10)];
}
long t1 = System.nanoTime();
System.out.println(" one read operation in " + (t1 -
    t0)/1000 + " nanoseconds");
```

körs, blev: första körningen 793ns, andra körningen 714ns, tredje körningen 532ns, fjärde körningen 445ns. slutsatsen som kan tas är att vi mäter en arrays medelvärde genom att mäta 1000 gånger.

vidare i uppgiften gavs en kod. Denna kod var dock inte fullständig då syntax error generas vid testkörning i *Visual Studio Code*. Specifikt var det problematiskt med kodraderna innehållandes orden void, index och arr. Koden fick därför skrivas om för att den skulle vara körbar. När detta var gjort kördes den 5 gånger. Det minsta talet som gavs var 12500ns och största 52000ns, medelvärdet blev 21000ns. Som gavs i uppgiftbeskrivningen är dessa tal annourlunda än de vi jobbat med tidigare.

I nästa uppgift skulle vi köra en snutt kod som är av typen "Benchmark" ett par gånger. Som student fick vi välja hur vi skulle rapportera resultatet, antingen som medelvärde, median eller minsta värde. Skribenten av den här rapporten valde att undersöka minsta värde. Efter att koden körts ett par gånger kunde ett konstaterande göras. Det minsta värdet var alltid eller nära talet 440ns. Medans max värdet inte alls var satt i sten i samma grad, utan kunde variera mellan 6200ns till 12500ns. När den modifierade "Benchmarken" kördes, denna:

```
bench(n, 1000000);
for (int i = 0; i < k; i++) {
long t = bench(n, loop);
if (t > max) max = t;
if (t < min) min = t;
total += t;
}
```

resultatet av denna exekvering av kod gav intressanta resultat. Minsta värdet var detsamma, ca 440ns medans max värdet istället sjunkit till ungefär 520ns i snitt. Detta visualiserar hur stor påverkan JIT har.

3 Search

Den givna koden kördes, med en modifiering för att kunna ändra storleken på arrayen och dess "keys". Bland annat lades dessa kodrader till för att enkelt kunna bestämma resultaten:

```
    public static void main(String[] args) {
        int n = 50000;      // Storleken pa arrayen
        int loop = 100000;  // antalet keys (storre an
            n)
        long timeTaken = search(n, loop);

        System.out.println("tid tagen for sokning: " +
            timeTaken + " nanoseconds.");
    }
}
```

Talet ”n” antog värderna 1000, 10000 och 50000. antalet ”Keys” var i samtliga fall större än n med en faktor 10, med undantag för 50000 då det var 100000. Koden exekverades 5 gånger för respektive n värde för att ta medelvärdet av värderna.

tid (ms)	keys	n-värde
21	10000	1000
750	100000	10000
8000	1000000	500000

Table 1: resultat av exekvering

Resultatet lades in Excel för att kunna visualiseras och få fram ett passande polynom.

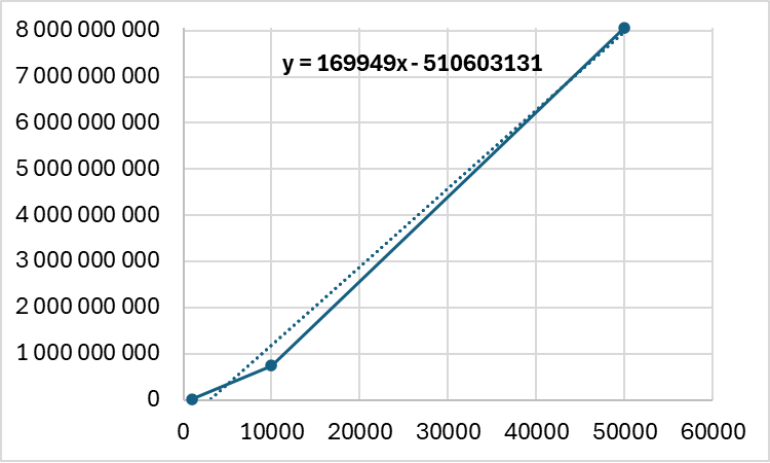

Figure 1: Excel graf med polynom, tid angiven i ns.

Med hjälp av funktionerna i excel gick det att bevisa att vi har en linjär

relation, då vi får fram det linjära polynomet: y = 169949x - 510603131. Vår funktion är mycket nära trendlinjen som är en perfekt linjär linje. Med mer data och fler referens punkter hade förhållandet mellan trendlinjen och polynomet blivit väldigt nära 1:1.

4 Duplicates

likt föregående uppgift, kördes den givna koden efter att modifiering skett, för att lättare bestämma resultaten. Följande rader lades bland annat till:

```
public static void main(String[] args) {
    int n = 100000;
    long timeTaken = duplicates(n);
    System.out.println("Time taken: " + timeTaken
        + " nanoseconds");
```

Koden exekverades 5 gånger för respektive n värde för att ta medelvärdet av värderna. Denna gång med en ökning av n-värdet med faktor 10.

tid (ms)	n-värde
0.2	100
8	1000
183	10000

Table 2: resultat av exekvering

Resultatet lades ännu än gång in i Excel för att kunna visualiseras och få fram ett passande polynom.

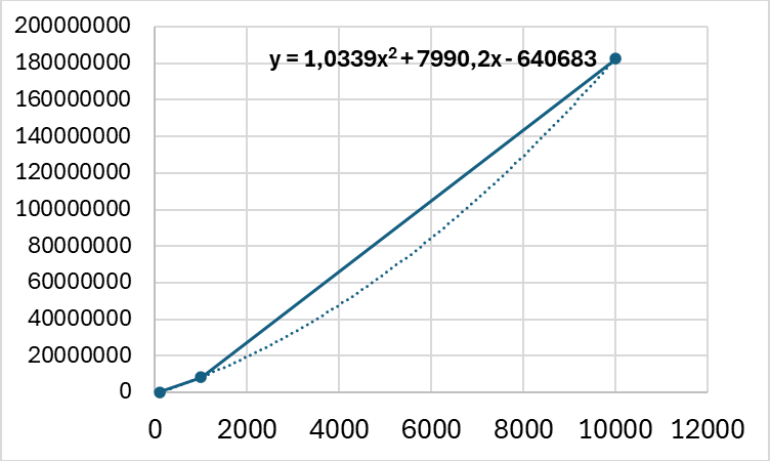

Figure 2: Excel graf med polynom, tid angiven i ns.

Med hjälp av funktionerna i excel gick det att bevisa att vi har en kvadratisk relation, då vi får fram det linjära polynomet: y = 1,0339x^2 + 7990,2x - 640683. Vår funktion är ganska nära trendlinjen som är en perfekt kvadratisk linje, projecerad på den här grafen. Avvikelsen är av större vikt denna gång än tidigare för den här uppgiften. Anledningen till det skulle kunna vara att inte tillräcklit mycket data och referenspunkter in men också eventuella data utliggare som påverkar medelvärdesberäkningen. En annan orsak kan vara olika mängder brus som påverkar olika mycket vid de olika mättillfällerna.