Programmering uppgift 1

1dt907 Algoritmer

Emil Ulvagården (eu222dq)

Run the program

Börja med att förflytta dig in i mappen "Algorithmer". Kör sedan:

```
javac *.java
```

Förflytta dig sedan ut ur "Algorithmer" men stanna i mappen "lab1"

Här efter kör du:

```
javac *.java
```

Nu kan du tillslut köra programmet med:

```
java Main.java
```

Uppgift 4

Jag valde att göra weighted quick union find i uppgift två. Tidtagningen i uppgift fyra tas för 9 olika mängder unions och över 9 olika mängder element. Antalet unions går från 10 000 till 90 000 med inkrement på 10 000. Antalet element går fårn 100 000 till 900 000 med inkrement på 100 000. Python programmet "plot.py" visar tiden för de olika testerna och för att uppdatera datan, koppiera listorna som skrivs ut ifrån Main.java in i respektive variabel. Följande m och b värden gavs från curve fiten för de olika plotterna:

Plot	UF Slope (m)	UF Intercept (b)	WQUF Slope (m)	WQUF Intercept (b)
100 000	0.04	-655.18	0.01	-189.65
200 000	0.05	-639.05	0.01	-201.71
300 000	0.06	-439.96	0.01	-197.07
400 000	0.08	-592.16	0.01	-193.52
500 000	0.09	-415.94	0.01	-187.33
600 000	0.11	-412.52	0.01	-197.36
700 000	0.12	-379.33	0.01	-195.38

Plot	UF Slope (m)	UF Intercept (b)	WQUF Slope (m)	WQUF Intercept (b)
800 000	0.14	-512.44	0.01	-189.93
900 000	0.16	-509.79	0.01	-188.26

Weighted quick union find är mycket snabbare än union find när antalet element och antalet random unions är samma. Det är då förväntat och det syns i tabellen ovan då m värdet från WQUF är mindre än m värdet från UF.

Uppgift 7

Från uppgift 6 har jag gjort two pointer lösningen. Jag kör 3 sum med Arrays i storlekar från 500 till 5000 med increment på 500. Dessa får då random värden från -5 gånger arrayns storleken till +5 gånger arrayns storlek. Detta körs 10 gånger för att få bättre värden. Python programmet "plot_sum3.py" visar tiden för de olika testerna och för att uppdatera datan, koppiera listorna som skrivs ut ifrån Main.java in i respektive variabel.

Plot	sum3 a fitted	sum3 b fitted	sum3 a computed	sum3 b computed	sum3 two pointer a fitted	sum3 two pointer b fitted	sum3 two pointer a computed	sum3 two pointer b computed
1	0.04	1.33	-4.66	1.33	0.00	1.80	-16.82	1.80
2	0.00	2.89	-23.67	2.89	0.00	2.02	-19.66	2.02
3	0.00	2.93	-24.17	2.93	0.00	2.04	-19.96	2.04
4	0.00	2.92	-24.04	2.92	0.00	1.94	-18.79	1.94
5	0.00	2.93	-24.20	2.93	0.00	1.88	-18.03	1.88
6	0.00	2.91	-23.97	2.91	0.00	2.11	-20.77	2.11
7	0.00	2.94	-24.29	2.94	0.00	2.07	-20.31	2.07
8	0.00	2.96	-24.52	2.96	0.00	2.05	-20.10	2.05
9	0.00	2.86	-23.30	2.86	0.00	1.98	-19.16	1.98
10	0.00	2.88	-23.53	2.88	0.00	1.94	-18.68	1.94

Tabellen ovan visar de a och b värden som framkommer i de olika plottarna av både curve fitted och computed för 3sum och 3sum two pointer. Värdena för curve fitten fås fram genom formeln

```
a * x**b
```

Värdena för computed fås genom formeln

```
x * 2**(a / b)**b
```

Om vi bortser från rad 1 i tabellen ser vi att b värdet för 3 sum är kring 3 vilket medför en tidskomplexitet på ca O(N^3).

Om vi sedan kollar på värdena för 3 sum two pointer ser vi att b värderna i tabellen ligger kring 2 om vi utesluter rad 1. Detta tyder på en tidskomplexitet kring O(N^2).

I bilden nedan kan man se skillnaden i hur snabbt three sum växer sett mot three sum two pointer, för samma antal tal.