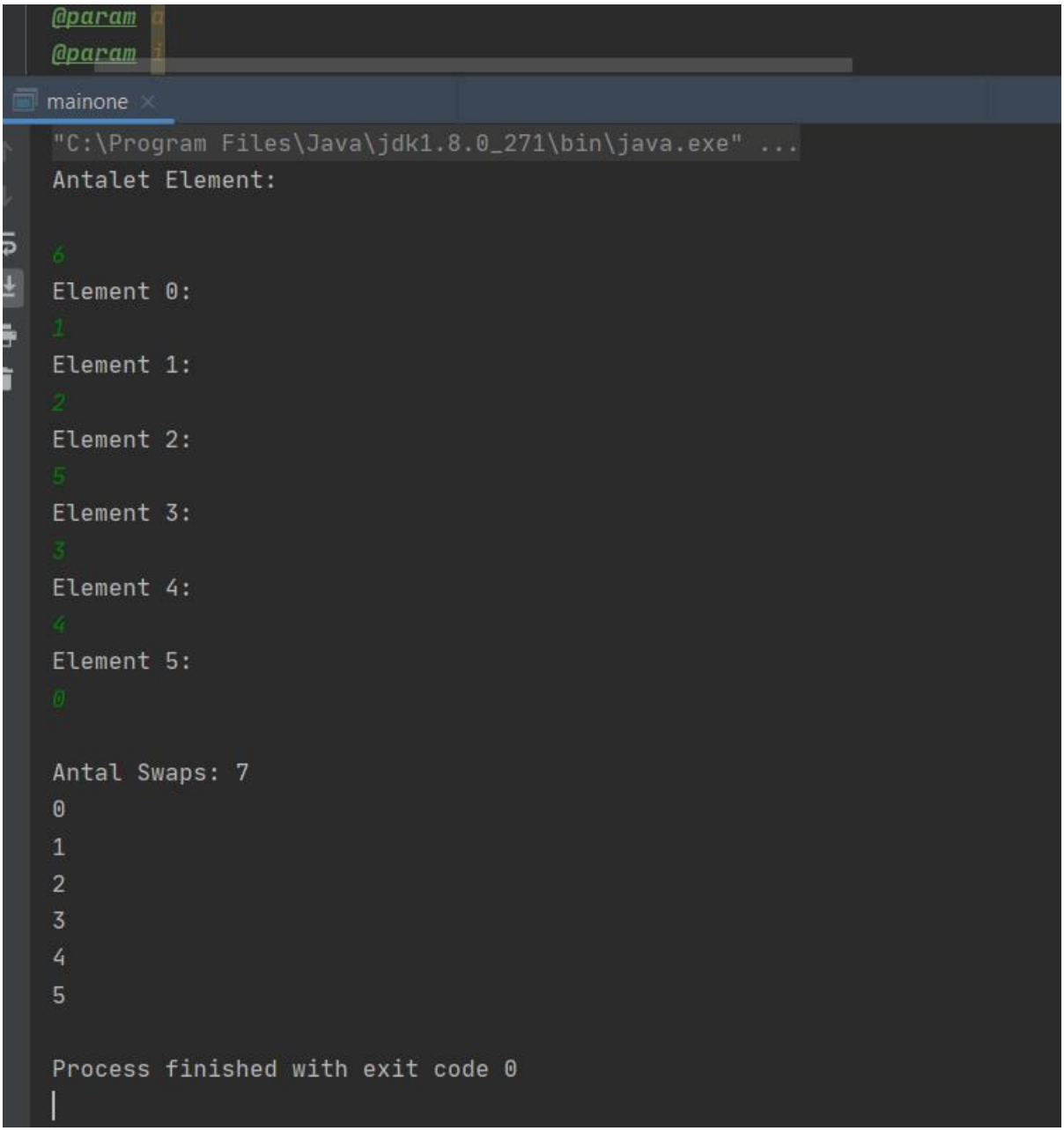
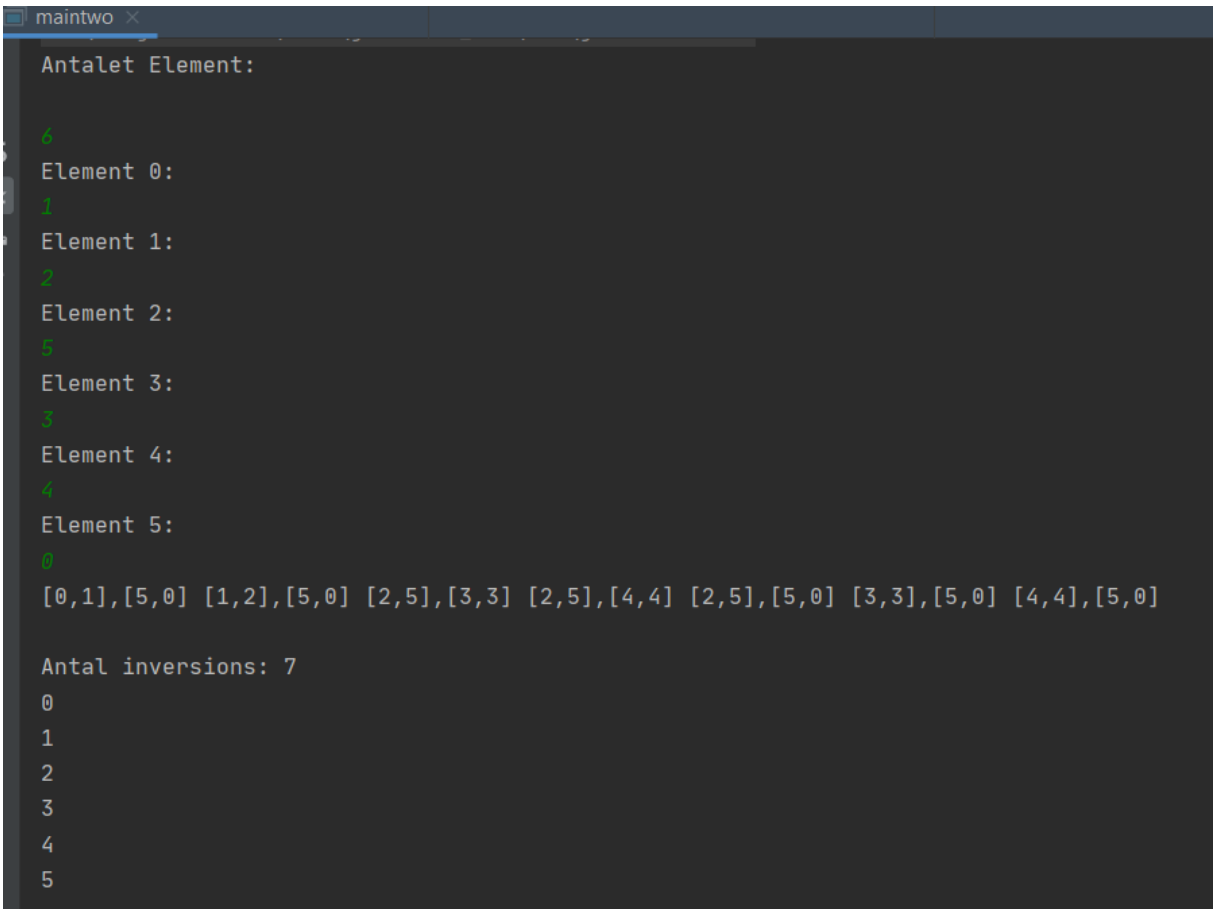
Labb2

Noel Tesfalidet

2.1



2.2



2.3

en temp variable används för att byta plats på element.
Sedan går man igenom hela listan och letar element vilket ger O(N) time complexity. Om ett negativ tal hittas så placeras den på första plats och nästa på andra osv. Memory complexity blir O(1); i bästa fall görs inga byten och i värsta så byter man alla element skulle säga att det blir O(N) complexity.

2.4

Merge sort körningstider

```
C:\WINDOWS\system32\cmd.exe

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourmerge < 10test.txt
Exekverings tiden blir: 0 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourmerge < 100test.txt
Exekverings tiden blir: 0 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourmerge < 1000test.txt
Exekverings tiden blir: 2 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourmerge < test10lax.txt
Exekverings tiden blir: 2 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourmerge < test100lax.txt
Exekverings tiden blir: 12 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourmerge < test500lax.txt
Exekverings tiden blir: 51 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourmerge < test750lax.txt
Exekverings tiden blir: 76 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourmerge < testmill.txt
Exekverings tiden blir: 104 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>
```

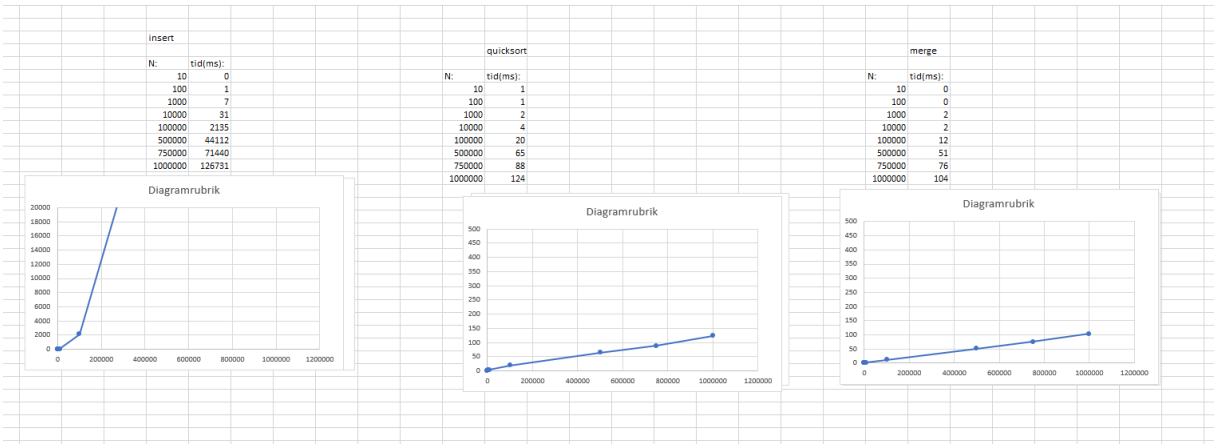
Insert sort körningstider

```
C:\WINDOWS\system32\cmd.exe

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourinsert < 10test.txt
Exekverings tiden blir: 0 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourinsert < 100test.txt
Exekverings tiden blir: 1 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourinsert < 1000test.txt
Exekverings tiden blir: 7 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourinsert < test10lax.txt
Exekverings tiden blir: 31 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourinsert < test100lax.txt
Exekverings tiden blir: 2135 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourinsert < test500lax.txt
Exekverings tiden blir: 44112 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourinsert < test750lax.txt
Exekverings tiden blir: 71440 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourinsert < testmill.txt
Exekverings tiden blir: 126731 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>
```

Quicksort körningstider

```
C:\WINDOWS\system32\cmd.exe

C:\Users\noete\IdeaProjects\Labb2ALD\src>javac fourquick.java

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourquick < 10test.txt
Exekverings tiden blir: 1 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourquick < 100test.txt
Exekverings tiden blir: 1 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourquick < 1000test.txt
Exekverings tiden blir: 2 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourquick < test10lax.txt
Exekverings tiden blir: 4 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourquick < test100lax.txt
Exekverings tiden blir: 20 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourquick < test500lax.txt
Exekverings tiden blir: 65 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourquick < test750lax.txt
Exekverings tiden blir: 88 ms

C:\Users\noete\IdeaProjects\Labb2ALD\src>java fourquick < testmill.txt
Exekverings tiden blir: 124 ms
```

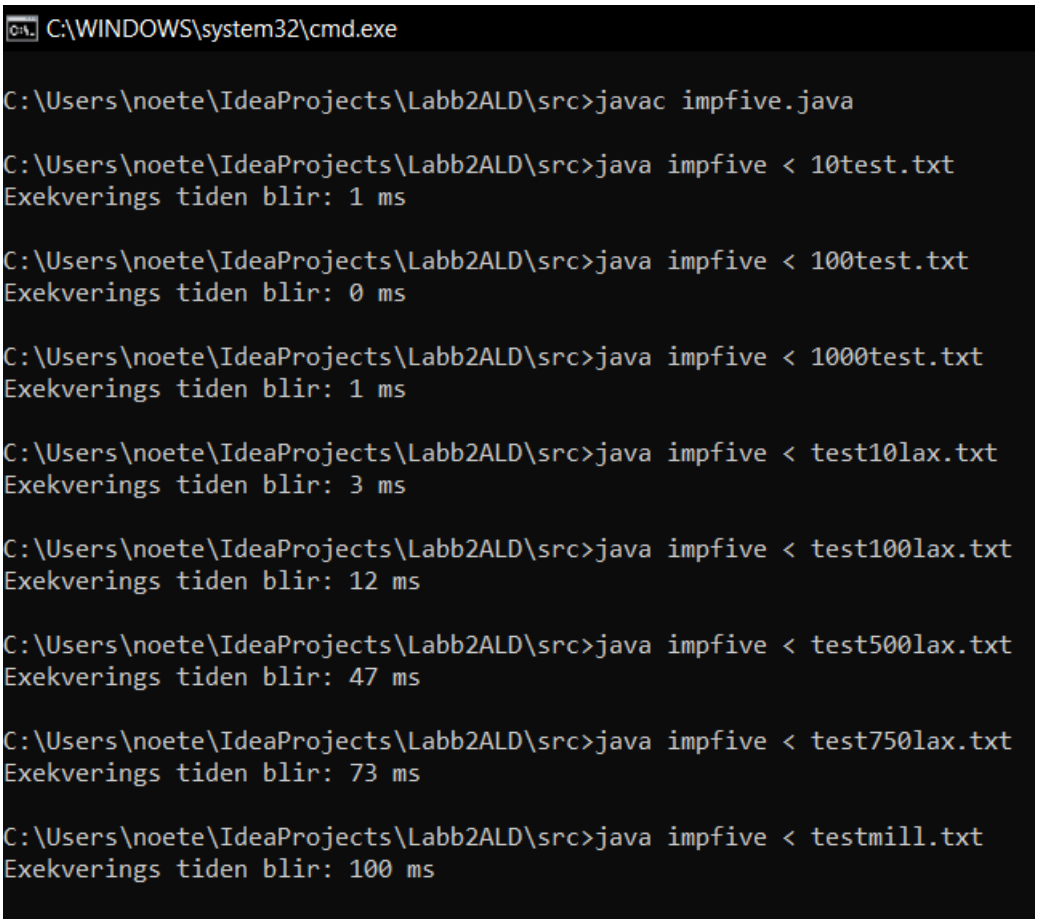
Grafer



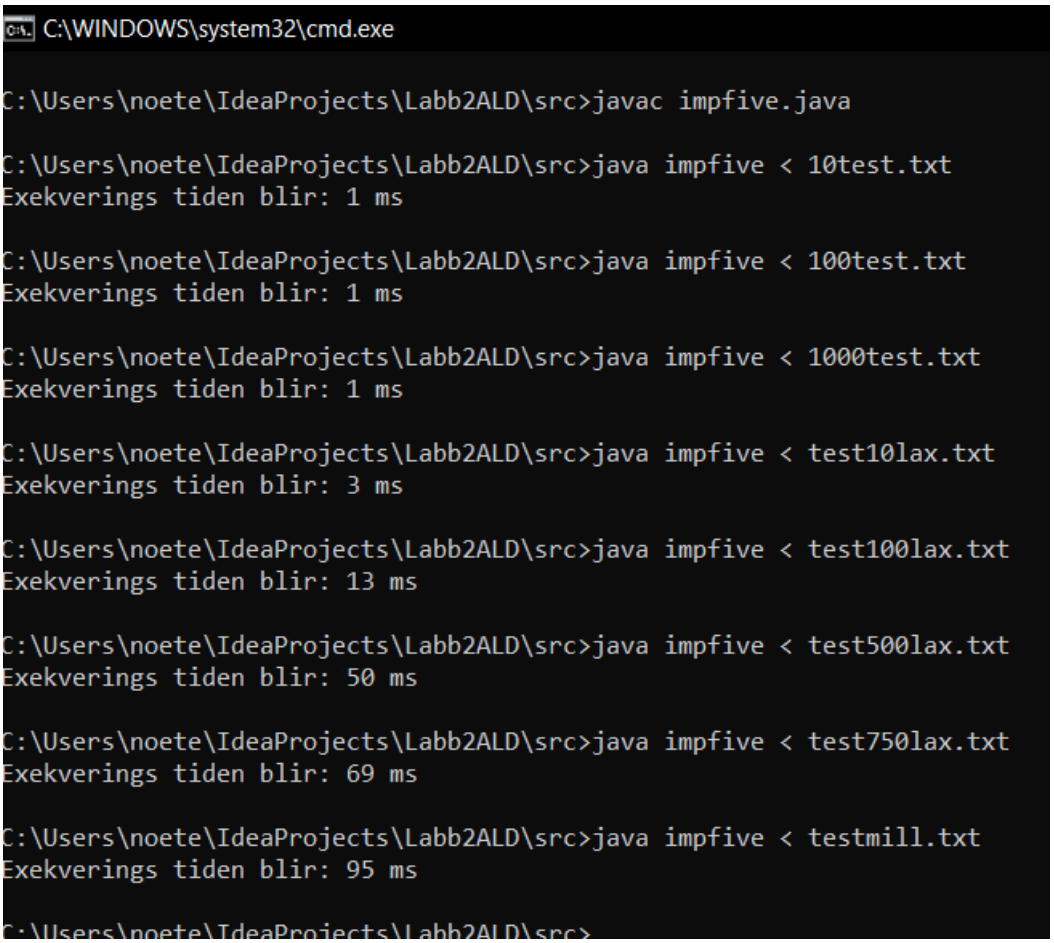
Insert är extremt långsam jämfört me de andra metoderna. Merge sort va lite snabbare än quicksort för mina testmätningar.

2.5

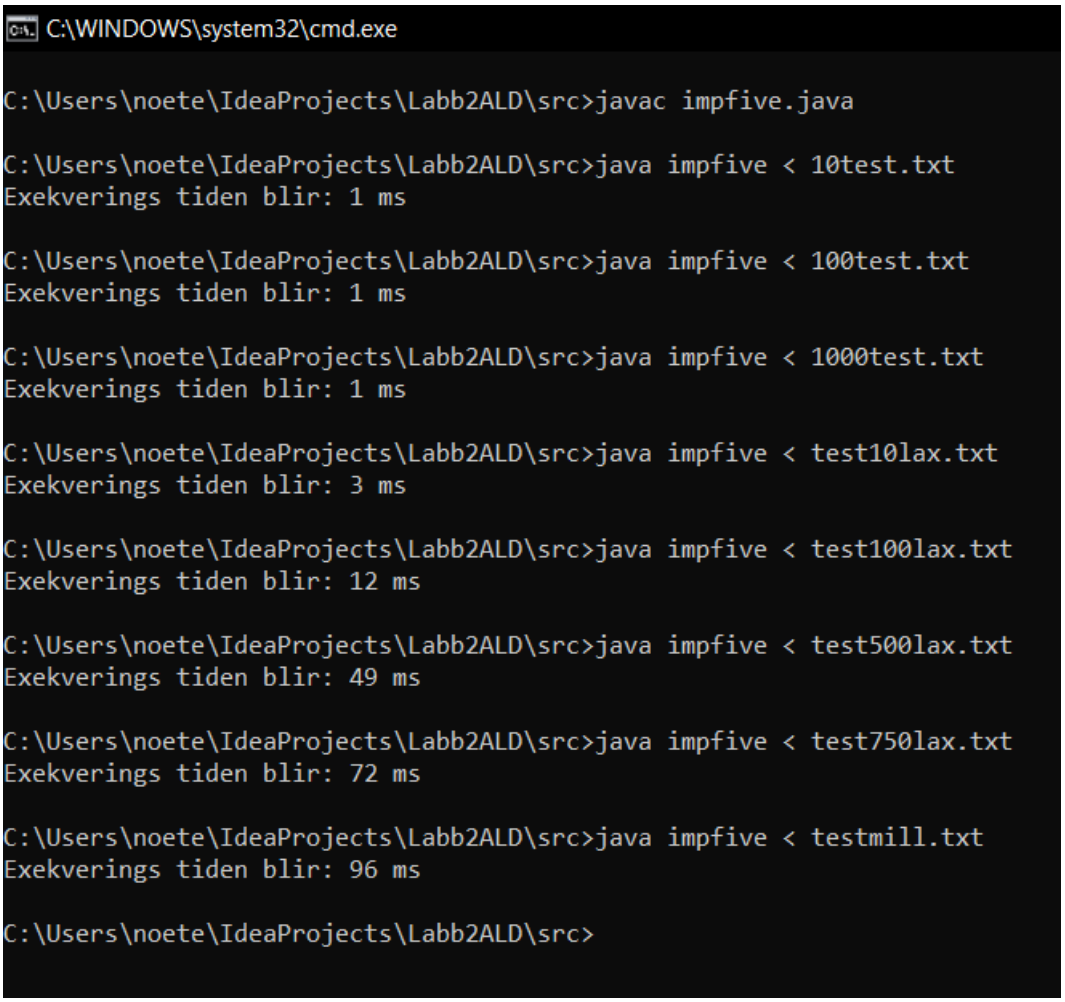
Cutoff = 7


C:\WINDOWS\system32\cmd.exe
C:\Users\noete\IdeaProjects\Labb2ALD\src>javac impfive.java
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 10test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 100test.txt
Exekverings tiden blir: 0 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 1000test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test10lax.txt
Exekverings tiden blir: 3 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test100lax.txt
Exekverings tiden blir: 12 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test500lax.txt
Exekverings tiden blir: 47 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test750lax.txt
Exekverings tiden blir: 73 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < testmill.txt
Exekverings tiden blir: 100 ms

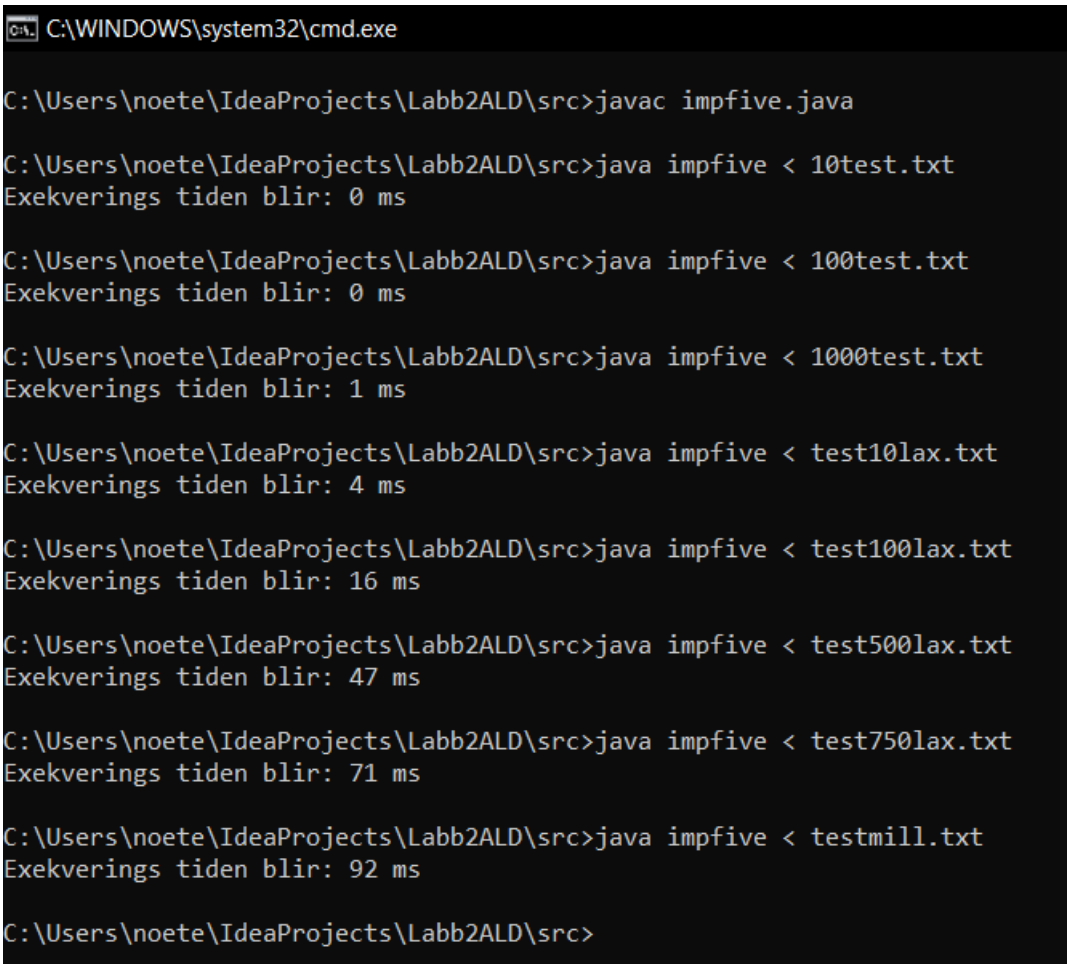
Cutoff = 14


C:\WINDOWS\system32\cmd.exe
C:\Users\noete\IdeaProjects\Labb2ALD\src>javac impfive.java
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 10test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 100test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 1000test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test10lax.txt
Exekverings tiden blir: 3 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test100lax.txt
Exekverings tiden blir: 13 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test500lax.txt
Exekverings tiden blir: 50 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test750lax.txt
Exekverings tiden blir: 69 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < testmill.txt
Exekverings tiden blir: 95 ms

Cutoff = 21

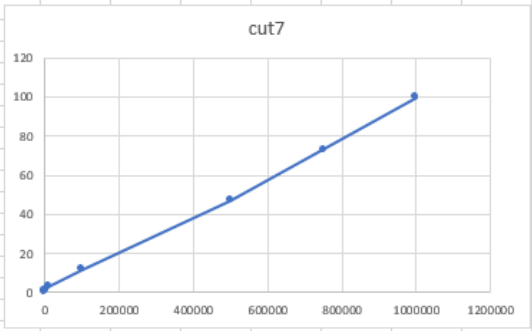

C:\WINDOWS\system32\cmd.exe
C:\Users\noete\IdeaProjects\Labb2ALD\src>javac impfive.java
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 10test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 100test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 1000test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test10lax.txt
Exekverings tiden blir: 3 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test100lax.txt
Exekverings tiden blir: 12 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test500lax.txt
Exekverings tiden blir: 49 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test750lax.txt
Exekverings tiden blir: 72 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < testmill.txt
Exekverings tiden blir: 96 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>

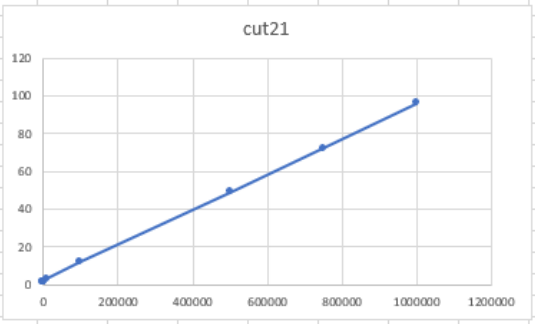
Cutoff = 30

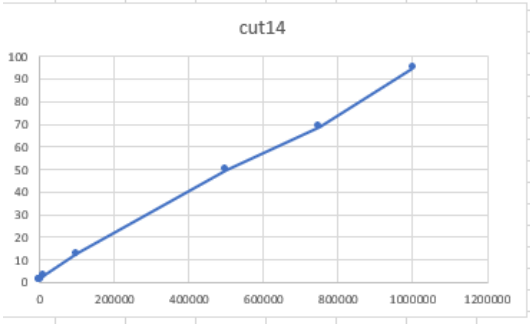

C:\WINDOWS\system32\cmd.exe
C:\Users\noete\IdeaProjects\Labb2ALD\src>javac impfive.java
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 10test.txt
Exekverings tiden blir: 0 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 100test.txt
Exekverings tiden blir: 0 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < 1000test.txt
Exekverings tiden blir: 1 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test10lax.txt
Exekverings tiden blir: 4 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test100lax.txt
Exekverings tiden blir: 16 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test500lax.txt
Exekverings tiden blir: 47 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < test750lax.txt
Exekverings tiden blir: 71 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java impfive < testmill.txt
Exekverings tiden blir: 92 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>

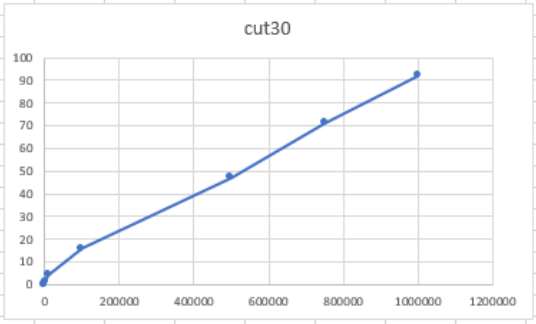
Grafer

cutof:	7		14		21		30	
	N:	tid(ms):	N:	tid(ms):	N:	tid(ms):	N:	tid(ms):
	10	1	10	1	10	1	10	0
	100	0	100	1	100	1	100	0
	1000	1	1000	1	1000	1	1000	1
	10000	3	10000	3	10000	3	10000	4
	100000	12	100000	13	100000	12	100000	16
	500000	47	500000	50	500000	49	500000	47
	750000	73	750000	69	750000	72	750000	71
	1000000	100	1000000	95	1000000	96	1000000	92



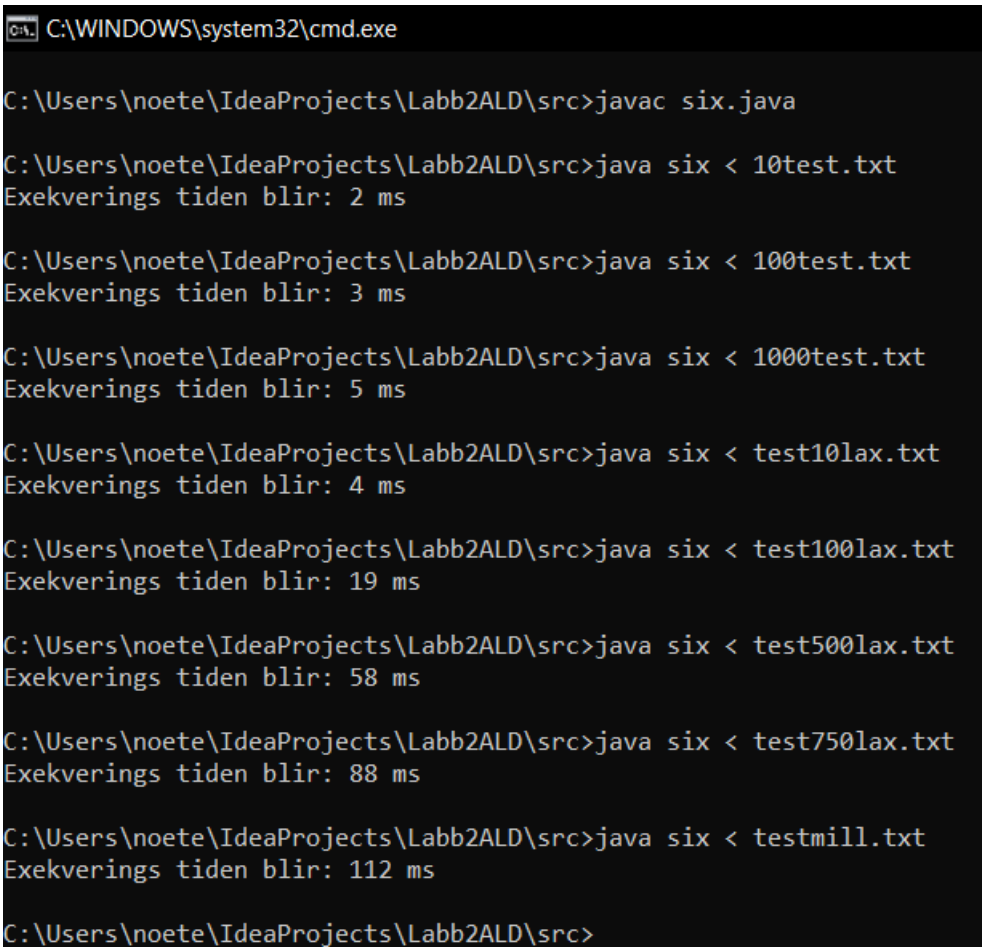






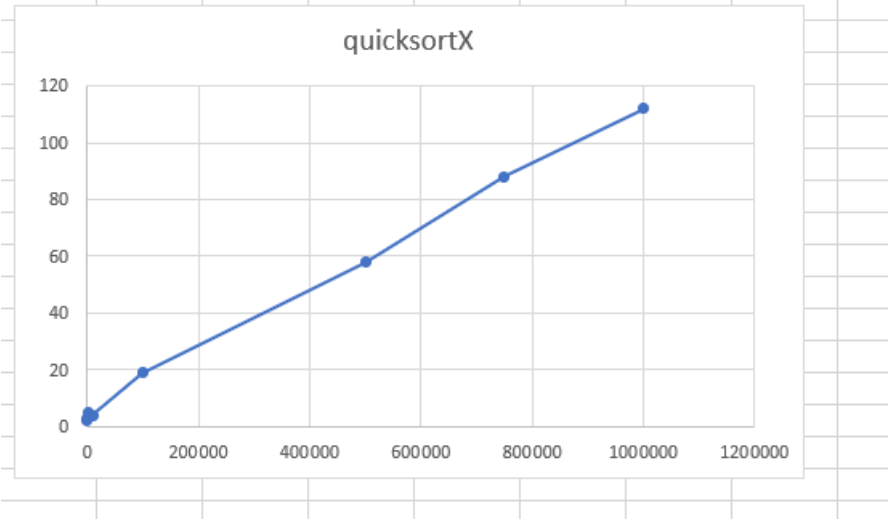
Optimal cutof fick jag melan 21 och 30. Denna förbätring kan man se bäst när större element ska sorteras. Det blev ca 10% snabbare sortering jämfört med vanlig mergesort.

2.6


C:\WINDOWS\system32\cmd.exe
C:\Users\noete\IdeaProjects\Labb2ALD\src>javac six.java
C:\Users\noete\IdeaProjects\Labb2ALD\src>java six < 10test.txt
Exekverings tiden blir: 2 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java six < 100test.txt
Exekverings tiden blir: 3 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java six < 1000test.txt
Exekverings tiden blir: 5 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java six < test10lax.txt
Exekverings tiden blir: 4 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java six < test100lax.txt
Exekverings tiden blir: 19 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java six < test500lax.txt
Exekverings tiden blir: 58 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java six < test750lax.txt
Exekverings tiden blir: 88 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>java six < testmill.txt
Exekverings tiden blir: 112 ms
C:\Users\noete\IdeaProjects\Labb2ALD\src>

Graf

N:	tid(ms):
10	2
100	3
1000	5
10000	4
100000	19
500000	58
750000	88
1000000	112



Mina tester visar att 3way partitioning är lite snabbare än vanlig quicksort men eftersom att man shufflar och bara kollar på 3 element så handlar det lite om tur också, att kontrollen efter median elementet går snabbare ön valen första elementet som median. Men såg ca 9% förbättring på en million element.

2.7

innan sorteringen så multiplicerades alla elementen med -1 så att det största elementet blir det minsta, sedan sorteras alla elementen och multipliceras en gång till och då har men arrayen i minskande ordning.