Merge och Quicksort

Datastrukturer och algoritmer DA252A
Grupp: Strict Defiant Wildcat
Gustav Florén, Carl Johan Helgstrand, Robert Lupeskovic

Quick + Mergesort

Standard merge och quicksort, resultat hämtat från godkänt test i codegrade.

Mergesort (Unordered)	Mergesort (Ordered)
T(10000)=0.003	T(10000)=0.0
T(20000)=0.001	T(20000)=0.001
T(40000)=0.002	T(40000)=0.002
T(80000)=0.006	T(80000)=0.004
T(160000)=0.014	T(160000)=0.006
T(320000)=0.017	T(320000)=0.014
T(640000)=0.036	T(640000)=0.031
T(2560000)=0.153 growth per N log N: 0.9626771556232869	T(2560000)=0.134 growth per N log N: 0.979117562455563
MergeSorter.sort tested ok!	MergeSorter.sort tested ok!

Quicksort (Unordered)	Quicksort (Ordered)
T(10000)=0.005	T(10000)=0.002
T(20000)=0.005	T(20000)=0.003
T(40000)=0.007	T(40000)=0.006
T(80000)=0.014	T(80000)=0.01
T(160000)=0.023	T(160000)=0.02
T(320000)=0.043	T(320000)=0.043
T(640000)=0.092	T(640000)=0.093
T(2560000)=0.457 growth per N log N: 1.125175089820568	T(2560000)=0.457 growth per N log N: 1.113076432940777
QuickSorter.sort tested ok!	QuickSorter.sort tested ok!

Optimerad Quick + Mergesort

Quicksort + Insertion (unordered)	Quicksort + Insertion (ordered)
T(10000)=0.002	T(10000)=0.0
T(20000)=0.002	T(20000)=0.001
T(40000)=0.004	T(40000)=0.003
T(80000)=0.008	T(80000)=0.006
T(160000)=0.018	T(160000)=0.012
T(320000)=0.026	T(320000)=0.024
T(640000)=0.055	T(640000)=0.053
T(1280000)=0.114	T(1280000)=0.111
T(5120000)=0.575 growth per N log N: 1.1478115396234232	T(5120000)=0.55 growth per N log N: 1.1275798438603781

Mergesort + insertion (unordered)	Mergesort + insertion (ordered)
T(10000)=0.004	T(10000)=0.0
T(20000)=0.002	T(20000)=0.0
T(40000)=0.001	T(40000)=0.0
T(80000)=0.002	T(80000)=0.001
T(160000)=0.004	T(160000)=0.003
T(320000)=0.009	T(320000)=0.004
T(640000)=0.017	T(640000)=0.01
T(1280000)=0.039	T(1280000)=0.02
T(5120000)=0.155 growth per N log N: 0.9044294272082892	T(5120000)=0.09 growth per N log N: 1.0240475127422888

Gränsvärde för M, Quick och Mergesort

Quicksort

Gränsvärde för M quicksort + insertion:
M = 14
N = 2000000, loopar 10 gånger på samma array för att få fram ett snitt
Tid: optimized quicksort < non-optimized quicksort

```
Ordered [QUICKSORT]
Ordered [QUICKSORT + INSERTION]
Average time taken by Non-Optimized quicksort: 195349360ns
Average time taken by Optimized quicksort: 183922060ns
```

M = 15
N = 2000000, loopar 10 gånger på samma array för att få fram ett snitt
Tid: optimized quicksort > non-optimized quicksort

```
Ordered [QUICKSORT]
Ordered [QUICKSORT + INSERTION]
Average time taken by Non-Optimized quicksort: 195914330ns
Average time taken by Optimized quicksort: 200814170ns
```

Reflektion

I detta scenariot var det betydligt svårare att hitta en tydlig brytpunkt för M i jämförelse med en optimerad mergesort. Förändringarna var mindre och det behövdes en större mängd tester för att kunna dra någon som helst slutsats. Vidare bestämde vi - som grupp - oss för att utföra testerna på samma array för att nå ett mer pricksäkert resultat.

```
int[] arr = new int[2000000];
int[] arrTwo = new int[2000000];
int[] shuffled = ArrayUtil.createShuffeled( N: 2000000);
// perform Quicksort NUM times and take average
for (int i = 0; i < NUM; i++) {
    System.arraycopy(shuffled,  srcPos: 0, arr,  destPos: 0, length: 2000000);
    System.arraycopy(shuffled,  srcPos: 0, arrTwo,  destPos: 0, length: 2000000);
```

Mergesort

Gränsvärde för M mergesort + insertion: M = 29
M = 29
N = 20000000, loopar 10 gånger på samma array för att få fram ett snitt
Tid: optimized mergesort < non-optimized mergesort

```
Ordered [MERGESORT]
Ordered [MERGESORT + INSERTION]
Average time taken by Non-Optimized mergesort: 311332830ns
Average time taken by Optimized mergesort: 291588960ns
```

M = 30
N = 20000000, loopar 10 gånger på samma array för att få fram ett snitt
Tid: optimized mergesort > non-optimized mergesort

```
Average time taken by Non-Optimized mergesort: 59645370ns
Average time taken by Optimized mergesort: 61256970ns
```

Reflektion

Till följd av vår algoritmdesign var brytpunkten för M med mergesort alltid 30, detta gällde både vid olika N och antal tester.

Ingen shuffle i quicksort?

När man inte har shuffle i quicksort för en sorterad array blir rumskomplexiteten o(n) och ger stackoverflow när N > 10000. Fungerar inte på större N, vidare ger en sorterad array värsta utfallet för quicksort med en tidskomplexitet på o(n^2).