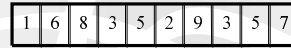


Quicksort — Sort Parts



Quicksort



Quicksort



Quicksort



Quicksort



Quicksort



Quicksort

1	6	5	3	5	2	9	3	8	7
---	---	---	---	---	---	---	---	---	---

Copyright © 2000, Russel Winder 13

Quicksort

1	6	5	3	5	2	9	3	8	7
---	---	---	---	---	---	---	---	---	---

Copyright © 2000, Russel Winder 14

Quicksort

1	6	5	3	5	2	3	9	8	7
---	---	---	---	---	---	---	---	---	---

Copyright © 2000, Russel Winder 15

Quicksort

1	6	5	3	5	2	3	9	8	7
---	---	---	---	---	---	---	---	---	---

Copyright © 2000, Russel Winder 16

Quicksort

1	3	5	3	5	2	6	9	8	7
---	---	---	---	---	---	---	---	---	---

Copyright © 2000, Russel Winder 17

Quicksort

1	3	5	3	5	2	6	9	8	7
---	---	---	---	---	---	---	---	---	---

Copyright © 2000, Russel Winder 18

Quicksort

- The partition operation uses $O(n)$ comparisons.
- There are $O(\log_2(n))$ levels of recursion and hence partitions.
- Thus, $T(n) = O(n \log_2(n))$.

Quicksort

- Although recursive, Quicksort operates with the data *in situ*, there is no need to make copies of the data.

Merge Sort

- Merge Sort partitions and merges streams in order to create the sorted sequence.
- Does not sort the data during partitioning (unlike Quicksort).

Merge Sort

1	6	8	3	5	2	9	3	5	7
---	---	---	---	---	---	---	---	---	---

Merge Sort

1	6	8	3	5	2	9	3	5	7
1	6	8	3	5					
2	9	3	5	7					

Merge Sort

1	6	8	3	5	2	9	3	5	7
1	6	8	3	5					
2	9	3	5	7					
1	6								
8	3	5							
2	9								
3	5	7							

Merge Sort

1	6	8	3	5	2	9	3	5	7
1	6	8	3	5	2	9	3	5	7
1	6	8	3	5	2	9	3	5	7
8	3	5	3	5	7				

Copyright © 2000, Russel Winder 31

Merge Sort

1	6	8	3	5	2	9	3	5	7
1	6	8	3	5	2	9	3	5	7
1	6	3	5	8	2	9	3	5	7

Copyright © 2000, Russel Winder 32

Merge Sort

1	6	8	3	5	2	9	3	5	7
1	3	5	6	8	2	3	5	7	9

Copyright © 2000, Russel Winder 33

Merge Sort

1	2	3	3	5	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

Copyright © 2000, Russel Winder 34

Merge Sort

- ADS has a Merge Sort.

Copyright © 2000, Russel Winder 35

Merge Sort

- Resource usage:
 - $S(n) = O(n \log_2(n))$
 - $T(n) = O(n \log_2(n))$

Copyright © 2000, Russel Winder 36

Merge Sort

- The partition operation uses no comparisons.
- There are $O(\log_2(n))$ levels of recursion and hence partitions.
- There are n comparisons at each level during the merge.
- Thus, $T(n) = O(n \log_2(n))$.

Merge Sort

- The algorithm has no instabilities (or degeneracies), unlike Quicksort.

Merge Sort vs. Quicksort

- Similar in some ways to Quicksort, the partitioning and the fact that a recursive implementation is usually given.

Merge Sort vs. Quicksort

- The two are very different though:
 - Merge Sort is two phase globally whereas Quicksort is not.
 - Merge Sort copies the data Quicksort does not.
 - Quicksort has pathological cases, Merge Sort does not.

Searching and Sorting in Collections

- There is a class `java.util.Arrays` which is a function server class serving searching and sorting functions.
- This deals only in arrays.
- There appears to be no predefined mechanism for searching and sorting other data structures.

Searching in Collections

- The search mechanism offered is binary chop search.
- The data in the array must be pre-sorted.
- For linear search, use an iterator.

Sorting in Collections

- The sort method for primitive types in Collections is a symbiotic sort: for arrays of less than 7 items Insertion Sort is used and otherwise a Quicksort was used.
- For non-primitive types Merge Sort is used.

Searching and Sorting in JGL

- JGL provides generic procedures: the methods can operate on any sequence for which iterators can be defined.
- This can lead to slow algorithms if an inappropriate mix of algorithms and data structure are used.

Searching in JGL

- The class `com.objectspace.jgl.algorithms.Finding` provides a number of searching related functions.
- All appear to be linear searches.

Sorting in JGL

- The class `com.objectspace.jgl.algorithms.Sorting` serves sorting algorithms.
- This provides Quicksort.

JGL and Iterators

- The use of iterators allows generic (polymorphic) methods to be defined.
- The use of iterators and abstract data type approaches leads to code that is reusable at run-time.

End of this Session