Übung 04: Quicksort, Mergesort

Aufgabe 1: Quicksort

Gegeben sei das folgende Array `A`: 4, 62, 13, 84, 35, 96, 57, 28, 79

a) Wenden Sie das Schema von *Hoare* ***einmal*** an, um das Array `A` in 2 Teile zu partitionieren. Als Pivot soll das rechte Element, also die 79, gewählt werden.
Hinweis: Geben Sie die Belegung des Arrays nach ***jeder einzelnen*** Vertauschung an!

b) Führen Sie den weiteren Verlauf von Quicksort manuell aus. Geben Sie die Belegung des Arrays nach ***jeder einzelnen*** Vertauschung aus. Wie viele Vertauschungen sind insgesamt notwendig?
Tipp: Überlegen Sie, in welcher Reihenfolge die rekursiven Aufrufe von Quicksort erfolgen. Halten Sie sich an den Pseudocode der Vorlesung.

c) Nehmen Sie an, dass alle Elemente des Eingabearrays den gleichen Wert haben. Erklären Sie anhand des Beispiel-Arrays *A*= <5, 5, 5>:
- Wie viele Element-Vertauschungen finden statt?
- Wie viele rekursive Aufrufe fallen an, falls alle Elemente des Eingabearrays den gleichen Wert haben?

d) Ein Array enthält *n* Elemente mit $n > 2$. Das Array enthält jedoch viele Duplikate und besteht insgesamt nur aus 2 verschiedenen Elementen. Was können Sie über das Array sagen, nachdem Sie einmal das Schema von Hoare angewendet haben, um das Array zu partitionieren?

Aufgabe 2: Iterativer Mergesort mit Queues

Mergesort kann auch mit Hilfe von Queues implementiert werden. Die Idee: Gegeben seien *n* zu sortierende Elemente. Erzeugen Sie erst *n* Queues, so dass anfangs jede einzelne Queue genau eines der zu sortierenden Elemente enthält. Eine zusätzliche *zentrale Queue* speichert alle diese *n* Queues speichert, also eine Queue von Queues.
Wiederholen Sie nun: Entfernen Sie die ersten beiden Queues aus der *zentralen Queue,* mischen Sie diese beiden Queues („`Merge`"-Operation) und fügen Sie das sortierte Ergebnis ***am Ende (!!!)*** *wieder* in die *zentrale Queue* ein. Wiederholen Sie, bis die *zentrale Queue* nur noch 1 Queue enthält.
Hinweis: Im Gitlab unter **src/de/th_rosenheim/ad/uebung04/angabe** ist ein Codegerüst sowie ein JUnit-Test vorgegeben. Das Verzeichnis `uebungen/uebung04` enthält nur die pdf-Angabe.

a) Implementieren Sie zunächst die folgende Operation:
`Queue<Comparable> merge(Queue<Comparable> a, Queue<Comparable> b)`

Die Methode mischt ***zwei sortierte Queues*** `a` und `b` und liefert eine neue Queue zurück, in der alle Elemente aus `a` und `b` enthalten sind und zwar wieder sortiert.
Hinweis: `Queue` ist in Java nur ein Interface. Verwenden Sie als konkrete Implementierung eine `LinkedList` und die entsprechenden Methoden, siehe Tabelle.

Queue-Operation	Operation bei einer LinkedList	Bedeutung
enqueue	add	Fügt ein Element am „Tail" hinzu.
dequeue	remove	Entfernt ein Element am „Head".
peek	peek	Schaut Element am „Head" an.

b) Implementieren Sie nun die folgende Methode: `sort(Comparable[] a)`
Die Methode sortiert das Array `a`. Dazu wird initial zunächst die „*zentrale Queue*" angelegt, siehe oben. Anschließend wird wiederholt die `merge`-Funktion aus a) aufgerufen. Testen Sie mit JUnit!

c) Welche asymptotische Laufzeit ergibt sich im Worst Case und im Best Case?