HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN · FH MÜNCHEN

Algorithmen und Datenstrukturen (IB) Praktikum
Aufgabenblatt 6 – Effizientes Sortieren

Ausgabe: 26. April 2018
Abgabe: 2. Mai 2018, 18:00 Uhr

Allgemeines

Diese Aufgabe baut auf der Vorgabe vom letzten Übungsblatt auf. Die Bearbeitung des fünften Übungsblattes ist aber nicht Voraussetzung für die Bearbeitung dieses Übungsblattes.

1 MergeSort (Pflichtaufgabe)

Anders als die bisher vorgestellten Algorithmen nutzt MergeSort ein temporäres Array beim Mischen zweier vorsortierter Arrays. Damit kommt es eigentlich nicht ohne direkte Zugriffe auf die zu sortierenden Elemente aus. Auf der Moodle-Seite finden Sie trotzdem eine Klasse „MergeSorter“, die dieses Problem dadurch umgeht, dass es ein temporäres Array nur nutzt, um die korrekte Reihenfolge der Indizes beim Mischen zu vermerken und dann entsprechende Swaps durchzuführen. Beachten Sie, dass dabei die Anzahl der Swaps auf Kosten anderer Zuweisungen niedrig gehalten wird.

1.1 Evaluation MergeSort (Pflilchtaufgabe)

Lassen Sie die vorgegebene Implementierung von MergeSort auf den unten vorgegebenen Eingaben für die Größen 10.000, 100.000 und 1.000.000 laufen und notieren Sie die Anzahl der durchgeführten Vergleiche in den folgenden Tabelle. Der Hypothese folgend, dass die Anzahl der Vergleiche in allen drei Fällen von der Form $c \cdot n \cdot \log_2(n)$ hat, geben Sie für alle gemessenen Werte den entsprechenden Wert von c an.

Beispiel: Bei der Ausführung von MergeSort auf einer Eingabe der Größe 10.000 messen Sie für eine sortierte Eingabe 64.608 Vergleiche. Für c ergibt sich in diesem Fall ein Wert von $c = 64608/(10000 \cdot \log_2(10000) = 0,49$.

	Sortierte Eingabe		Fast Sortiert		Unsortiert		Umgekehrt Sortiert	
n	Vergleiche	c	Vergleiche	c	Vergleiche	c	Vergleiche	c
10^4	64608	0,486	71391	0,537	120461	0,907	69008	0,519
10^5	815024	0,491	898091	0,541	1536382	0,925	853904	0,514
10^6	9884992	0,496	10883525	0,546	18673536	0,937	10066432	0,505

Hinweis: Um die effizienten Verfahren zu evaluieren, nutzen Sie Eingabegrößen, die mit den einfachen Algorithmen nicht mehr in sinnvoller Zeit sortiert werden. Kommentieren Sie diese Algorithmen in der Main-Methode aus.

1.2 Verbesserung MERGESORT (Pflichtaufgabe)

MergeSort lässt sich verbessern, indem man den Merge zweier Teilarrays überspringt, wenn der letzte Wert des ersten Arrays kleiner oder gleich dem ersten Wert des zweiten ist. Kopieren Sie die Klasse `MergeSort` in eine Klasse `BetterMergeSort` und implementieren Sie diese Verbesserung. Lassen Sie beide Implementierungen für Eingaben der Größe 1000000 laufen; erstellen Sie ein Diagramm, dass die Anzahl der Vergleiche von Original und Verbesserung für die Eingaben gegenüberstellt.

2 QUICKSORT (tw. Pflichtaufgabe)

2.1 Implementierung QUICKSORT (Pflichtaufgabe)

Implementieren Sie den in der Vorlesung vorgestellten Algorithmus QUICKSORT! Dafür steht Ihnen im Moodle bereits eine Vorlage zur Verfügung, in der nur noch die Partitionierung selbst fehlt. Im Vergleich zur Vorlesung unterscheidet sich diese Variante dadurch, dass Sie den Wert am Ende des ersten Fünftels der Eingabe als Pivot wählt. Dieser Wert ist weder wie die Mitte in einigen Fällen optimal, noch führt er zu linear vielen Rekursionen wie der erste Wert im Array.

2.2 Verbesserung Pivotsuche QUICKSORT (optional)

Erstellen Sie zwei Kopien ihrer QuickSort-Implementierung und wandeln sie so ab, dass

- ein zufälliges Pivot-Element gewählt wird
- Bei Intervallen mit mehr als 30 zu pivotisierenden Elementen drei zufällige Elemente ausgewählt werden und das der Größe nach mittlere Element als Pivot verwendet wird

Lassen Sie die drei Implementierungen für Eingaben der Größe 1000000 laufen; erstellen Sie ein Diagramm, dass die Anzahl der Vergleiche der Varianten für die Eingaben gegenüberstellt.

Hinweis: Um einen zufälligen Index zwischen `a` (inklusive) und `b` (inklusive) zu erzeugen, nutzen Sie bitte den Aufruf

```
ThreadLocalRandom.current().nextInt(a,b+1);
```

Um den der Größe nach mittleren Wert dreier Kandidaten zu finden, müssen Sie die Werte vergleichen, nicht die Indizes!