Aufgabenblatt 2

Algorithmen und Datenstrukturen

Lesen sie alle Aufgaben genau durch, bevor sie mit der Bearbeitung beginnen.

Legen sie für die Bearbeitung ein Textdokument an. In diesem werden ihre Ergebnisse gesammelt. Ihren Code brauchen sie nicht abgeben (siehe letzte Aufgabe), erstellen sollen sie ihn aber trotzdem. Für die Bearbeitung benötigen sie eine (beliebige) Chat-KI ihrer Wahl. Falls sie kein Account o.ä. haben, so können sie einfach den Chat auf https://duck.ai verwenden.

Aufgabe 1: Maximalwerte in sortierten Arrays

- Erstellen sie ein neuen IntelliJ-Projekt namens “Aufgabenblatt2”, erstellen sie darin ein Package “aufgabenblatt2” für ihre eigenen Klassen und kopieren sie das util-Package mit Util-Klasse aus dem vorigen Projekt (oder erstellen sie es wie im vorigen Aufgabenblatt beschrieben). Kopieren sie auch die “Sorter”-Klasse aus dem vorigen Aufgabenblatt in das aufgabenblatt2 Package.
- Erstellen sie eine Methode in der Klasse “Util” mit der folgenden Signatur:

```
public int findMaxBySorting(int[] array)
```

- Implementieren sie die Methode: Sortieren sie das übergebene Array mit der insertionSort()-Methode und überlegen sie, wie sie danach den Maximalwert finden können. Geben sie den Maximalwert zurück.
- Wie hoch ist die Laufzeitkomplexität dieser Methode, wenn man den Aufwand für das Sortieren außer Acht lässt? Schreiben sie die Antwort in das Textdokument unter Aufgabe 1.
- Lassen sie sich von einer KI ihrer Wahl die Aufgabe generieren. Geben sie der KI dabei genug Informationen, um die Aufgabe bearbeiten zu können. Vergleichen sie ihre Lösung mit der KI Lösung. Wo liegen die Unterschiede? Was ist gleich? Schreiben sie die Antwort in ihr Textdokument unter Aufgabe 1 auf.
- Lassen sie die KI die Laufzeitkomplexität ihrer eigenen Lösung abschätzen. Liegt die KI richtig? Kopieren sie die KI-Antwort in das Textdokument und auch ihre Bewertung der Abschätzung.

Aufgabe 2: Werte in sortierten Arrays finden

- Erstellen sie eine Methode in der Klasse “Util” mit der folgenden Signatur:

```
public Integer findValueBySorting(int[] array, int value)
```

- Implementieren sie die Methode: Sortieren sie das übergebene Array mit der insertionSort()-Methode. Überlegen sie sich nun eine geschickte Strategie, um den übergebenen Wert “value” mit möglichst wenig Aufwand im Array zu finden (Tipp: Denken sie an das Zahlenraten aus dem letzten Aufgabenblatt).
- Geben sie die Position des Wertes zurück, wenn er gefunden wurde, ansonsten geben sie “null” zurück.
- Wie hoch ist die Laufzeitkomplexität bei einem geschickten Vorgehen in dieser Methode, wenn man den Aufwand für das Sortieren außer Acht lässt? Schreiben sie die Antwort in ihr Textdokument unter Aufgabe 2 auf.
- Lassen sie sich von einer KI ihrer Wahl die Aufgabe, insbesondere die Methode autoGuess() generieren. Geben sie der KI dabei genug Informationen, um die Aufgabe bearbeiten zu können. Vergleichen sie ihre

Lösung mit der KI Lösung. Wo liegen die Unterschiede? Was ist gleich? Schreiben sie die Antwort in ihr Textdokument unter Aufgabe 3 auf.

Aufgabe 3: Merge-Sort

- Erweitern sie die Klasse “Sorter”. Fügen sie eine Methode mit der Signatur “public void mergeSort(int[] array)” hinzu.
- Implementieren sie die Methode mit dem Algorithmus “Sortieren durch Mischen”.
- Testen Sie Ihre Implementation, indem sie die folgende Methode in der Klasse Sorter anlegen, eine Instanz erstellen und die Methode testMergeSort() auf der Instanz ausführen.

```
public void testMergeSort()
{
      int[] array = new int[100];
      Util util = new Util();
      util.fillArrayRandom(array, 100);
      Sorter mySorter = new Sorter();
      mySorter.mergeSort(array);
      util.printArray(array);
}
```

- Lassen sie sich von einer KI ihrer Wahl die Aufgabe lösen. Geben sie der KI dabei genug Informationen, um die Aufgabe bearbeiten zu können. Vergleichen sie ihre Lösung mit der KI Lösung. Wo liegen die Unterschiede? Was ist gleich? Bei dieser Aufgabe ist die Wahrscheinlichkeit für Unterschiede besonders hoch, da es viele Varianten des Merge-Sorts gibt. Schreiben sie die Antwort in ihr Textdokument unter Aufgabe 3 auf.

Aufgabe 4: Optimierter Merge-Sort

- In der Vorlesung haben wir gelernt, dass bei kleinen oder schon sortierten Arrays der Insertion-Sort (Sortieren durch Einfügen) sehr schnell sein kann, bei größeren Arrays aber schnell sehr langsam wird ($O(n^2)$).
- Implementieren sie eine Methode insertionSort(int[] array), welche das Sortieren durch Einfügen implementiert (falls Sie das Sortierverfahren im vorigen Aufgabenblatt implementiert haben, dann dürfen sie den Algorithmus hierfür kopieren).
- Optimieren Sie daher Ihren Merge-Sort: Kopieren Sie den Methodeninhalt in die Methode “optimizedMergeSort” mit ansonsten identischer Signatur und ändern Sie diese Methode dann so ab, dass Sie bei Arrays der Größe 10 oder kleiner den Insertion-Sort benutzt.
- Testen Sie Ihre Implementation, indem sie die folgende Methode in der Klasse Sorter anlegen, eine Instanz erstellen und die Methode testMergeSort() auf der Instanz ausführen.

```
public void testOptimizedMergeSort()
{
      int[] array = new int[100];
      Util util = new Util();
      util.fillArrayRandom(array, 100);
      Sorter mySorter = new Sorter();
      mySorter.optimizedMergeSort(array);
      util.printArray(array);
}
```

- Kopieren Sie Ihre Lösung in eine KI ihrer Wahl und fragen sie diese, wie man dieses Sortierverfahren nennt und ob die KI noch weitere Tipps hat. Kopieren sie die KI-Antwort in ihr Textdokument unter Aufgabe 4.

Aufgabe 5: Umgekehrte Sortierreihenfolge

- Ändern sie die Methoden insertionSort, mergeSort und optimizedMergeSort so ab, dass sich die partielle Ordnung der Sortierung umdreht (d.h. größter Wert vorne, kleinster Wert hinten, statt umgekehrt). Schreiben sie unter Aufgabe 5 im Textdokument, was sie genau geändert haben.

Aufgabe 6: Array-Listen

Erstellen sie eine neue Klasse “IntArrayList”.

- Implementieren sie eine Array-Liste für int-Werte. Implementieren sie dabei insbesondere die Methoden mit den folgenden Signaturen:
- `public void add(int value)` – Fügt einen neuen Wert am Ende der Liste ein.
- `public int get(int pos)` – Gibt den Wert an der Stelle “pos” in der Liste als Rückgabewert zurück, oder -1 falls “pos” außerhalb der Liste liegt (z.B. >= Listengröße oder negativ).
- `public int remove(int pos)` – Entfernt den Wert an der Stelle “pos” in der Liste und gibt ihn als Rückgabewert zurück. Hierbei soll kein “Loch” in der Liste entstehen. Falls “pos” außerhalb der Liste liegt (z.B. >= Listengröße oder negativ), soll -1 zurückgegeben werden und die Liste bleibt unverändert.
- `public int size()` – Gibt die aktuelle Anzahl der Elemente in der Liste zurück.
- `public void sort()` – Sortiert die Liste (Sie können hier Insertion- oder Merge-Sort verwenden).
- Überlegen Sie einmal, welche Laufzeitkomplexität die einzelnen Methoden haben. Schreiben sie die Antwort in ihr Textdokument unter Aufgabe 6.
- Lassen sie sich von einer KI ihrer Wahl die Aufgabe lösen. Geben sie der KI dabei genug Informationen, um die Aufgabe bearbeiten zu können. Vergleichen sie ihre Lösung mit der KI Lösung. Wo liegen die Unterschiede? Was ist gleich? Schreiben sie die Antwort in ihr Textdokument unter Aufgabe 6 auf.

Aufgabe 7: Abgabe

- Erstellen sie eine .pdf-Datei aus ihrem Textdokument und laden sie es in Moodle als Gruppenabgabe hoch (nur 1x pro Kleingruppe).