SCHOOL-SCOUT.DE

Unterrichtsmaterialien in digitaler und in gedruckter Form

Auszug aus:

Klausur Informatik zum Thema Algorithmen und Sortierverfahren

Das komplette Material finden Sie hier:

School-Scout.de

Klausur zur Informatik in der Einführungsphase

Kurzvorstellung des Materials	Beim vorliegenden Material handelt es sich um eine Klausur zur Informatik in der Einführungsphase der gymnasialen Oberstufe NRW im zweiten Halbjahr. Sie kam in der Praxis bereits zum Einsatz. Beim Erstellen der Klausur wurde sehr darauf geachtet, alle Aufgabenstellungen in einen Sachzusammenhang zu stellen, wie dies auch im Zentralabitur NRW vorgesehen ist. Obgleich im Unterricht Java eingesetzt wurde, werden in dieser Klausur keine Implementierungen vorgegeben oder verlangt. Die Klausur schließt inhaltlich an die Unterrichtssequenz zur Bildung des Begriffs Algorithmus an.
Inhaltliche Schwerpunkte	Algorithmen und Sortierverfahren
Aufgabenarten	Interpretation eines Pseudocodes, Anwendung der Definition des Begriffs „Algorithmus“, Anwendung und Bewertung von Sortieralgorithmen
Inhaltliche Voraussetzungen	• Definierende Eigenschaften von Algorithmen • Beschreibung mittels Pseudocode und Flussdiagramm • Sortieralgorithmen Bubblesort, Minsort und Insertionsort
Dauer	2 Unterrichtsstunden
Allgemeine Hinweise	Zur Erstellung der Flussdiagramme wurde das kostenlose Werkzeug *PapDesigner* verwendet, das man unter `http://www.friedrich-folkmann.de` erhalten kann.
Schlüsselwörter	Klausur, Informatik, Einführungsphase, Java, Flussdiagramme, Pseudocode, Algorithmus, Sortierverfahren, Bubblesort, Minsort, Insertionsort

Aufgabe 2 (46 Punkte)**.** Gegeben sei die folgende Reihe von Spielkarten:

8	2	5	1	3

(a) Als erstes soll der Bubblesort-Algorithmus betrachtet werden. Dabei gehen wir von dieser sehr einfachen Version aus:
Eingabe: Ein Reihe von n Spielkarten.
Ausgabe: Umsortierung der Reihe, die in aufsteigender Folge sortiert ist.

Wiederhole $(n-1)$-mal:

(1) Setze $k = 1$.

(2) Solange $k \leq n-1$ gilt, wiederhole:

Falls die k-te Karte größer ist als die $(k+1)$-ste Karte, vertausche die beiden.

Erhöhe k um eins.

STOPP.

(i) *Sortiere mittels des oben dargestellten Bubblesort-Algorithmus die gegebene Reihe. Halte den Ablauf tabellarisch fest. Dabei soll jede einzelne Vertauschung in der Tabelle nachvollziehbar sein.* (8 Punkte)

(ii) *Erläutere mit eigenen Worten eine mögliche Verbesserung des Bubblesort-Algorithmus. Erstelle dann einen Pseudocode, der den Bubblesort-Algorithmus inklusive Deiner Verbesserung darstellt.* (16 Punkte)

(b) Wir wollen nun den Minsort-Algorithmus betrachten. Wir verwenden hier die Version, die mit nur einer Reihe von Spielkarten arbeitet.

(i) *Gib die Grundidee des Minsort-Algorithmus wieder.* (4 Punkte)

(ii) *Sortiere mittels des Minsort-Algorithmus die gegebene Reihe. Halte den Ablauf tabellarisch fest. Dabei soll wieder jede einzelne Vertauschung nachvollziehbar sein.* (4 Punkte)

Modelllösungen

Aufgabe 1. (a) Das Wort *endlich* hat hier dreierlei Bedeutung:

1. Das Verfahren muss nach endlich vielen Schritten zu einem Ende kommen.
2. Das Verfahren darf nur endlich viel Speicherplatz benötigen.
3. Das Verfahren muss sich mit einem endlichen Text (sei es Programmtext, Pseudocode etc.) beschreiben lassen.

(b) (i) Man betrachtet der Reihe nach alle Paare in dem Regal. Die Variable k gibt dabei an, welches Paar man aktuell betrachtet (und mit dem bisher schönsten vergleicht). Die Variable s gibt an, welches Paar der bereits betrachteten Paare das schönste ist.

(ii) Das Verfahren würde so ablaufen:

- Setze $s = 1$.
- Setze $k = 2$.
- Das zweite Paar ist schöner als das erste, also setze $s = 2$.
- Das dritte Paar ist nicht schöner als das zweite (also bleibt $s = 2$).
- Das zweite wird als Ergebnis bekannt gegeben.

(iii) Die Eindeutigkeit ist verletzt, da es sicherlich Geschmackssache ist, ob man ein Paar Schuhe schöner findet als ein anderes.

Aufgabe 2. (a) (i) Es ergibt sich folgende Tabelle:

8	2	5	1	3
2	8	5	1	3
2	5	8	1	3
2	5	1	8	3
2	5	1	3	8
2	1	5	3	8
2	1	3	5	8
1	2	3	5	8

(ii) Man könnte bei jedem Durchlauf der Reihe sich merken, ob noch eine Vertauschung nötig war. Wenn dies nicht der Fall ist, dann ist man fertig und kann nach möglicherweise weniger als $n - 1$ Wiederholungen den Algorithmus beenden. Ein möglicher Pseudocode:

(1) Setze $t = 0$.
(2) Setze $k = 1$.
(3) Falls die k-te Karte größer ist als die $(k + 1)$-ste Karte, vertausche die beiden und setze $t = 1$.
(4) Erhöhe k um eins.
(5) Falls $k \leq n - 1$ gilt, gehe zurück zu (3).
(6) Falls $t = 1$ gilt, gehe zurück zu (1). Sonst STOPP.

(b) (i) Das Minsort-Verfahren sucht aus dem noch unsortierten Teil einer Reihe die jeweils kleinste Zahl heraus und setzt diese an den Anfang dieses Bereichs. Dabei wandert der Beginn des unsortierten Bereichs in jeder Runde eine Position weiter nach rechts.

(ii) Es ergibt sich folgende Tabelle:

8	2	5	1	3
1	2	5	8	3
1	2	3	8	5
1	2	3	5	8

(iii) Zu A: Ist die k-Karte kleiner als die *min*-te Karte?
Zu B: Vertausche die *runde*-te Karte und die *min*-te Karte.
Zu C: Setze $runde = runde + 1$.

(c) Dieser Algorithmus arbeitet genau entgegengesetzt dem Minsort-Algorithmus. Es wird in dem jeweils noch unsortierten Teil der Reihe die größte Zahl gesucht. Diese wird dann an das Ende des unsortierten Bereichs gesetzt. Dadurch entsteht der sortierte Bereich am rechten Ende der Reihe.

Aufgabe 3. (a) Der Minsort-Algorithmus liegt hier näher, da ein Mensch mit einem Blick recht schnell erkennen kann, welcher der noch unsortierten Spieler der kleinste ist. Bubblesort wäre für einen Menschen hingegen ein recht unnatürlicher Prozess.

Hier sind alternative Argumentationen denkbar. Auch solche, die nachvollziehbar zur Wahl von Bubblesort führen, sollen als richtig gewertet werden.

(b) Man könnte sich vorstellen, dass der Trainer die Spieler bereits beim Betreten der Halle sortiert. Jeder Spieler, der die Halle betritt, wird dann angewiesen, sich in die Reihe an passender Stelle einzufügen. In dieser Situation bietet sich Insertionsort an, weil noch nicht alle Spieler von Beginn an in der Halle sind. Bei den anderen Verfahren müsste man erst warten, bis alle da sind.

Natürlich sind viele andere Beispiele denkbar.