Mündliche Abiturprüfung
Fach Informatik - Grundkurs

Prüfender Fachlehrer (Autor der Aufgabe): Anton Weigel

Vorbereitungszeit: 20 min, Prüfungszeit 30 min

Sortieralgorithmen

1. Einordnung der Aufgabe in den Lehrplan, Taxonomie:

Sortieralgorithmen sind im Lehrplan für die Oberstufe des Gymnasiums in den Lernbereich 2, Grundkurs einzuordnen. In diesem Lernbereich ist eines der Ziele, dass die SuS „die Implementierung eigener Lösungen zu ausgewählten komplexen Problemstellungen beherrschen“, wobei in den Bemerkungen zu diesem Thema auch explizit die Sortieralgorithmen erwähnt werden. Auch das Kennen der Komplexität und Effizienz von Algorithmen ist Teil dieses Lernbereichs, wozu auch die Sortieralgorithmen und Speicherkomplexität gehören.

Lernbereich 2: Algorithmierung und Programmierung	**20 Ustd.**
Kennen des erweiterten Algorithmusbegriffes	Eigenschaften von Algorithmen → Kl. 8, LB 1
Beherrschen der Implementierung strukturierter Datentypen in einer Programmierumgebung	Zeichenkette, Feld → Kl. 10, LB 1
Beherrschen der Arbeit mit Unterprogrammen	
- Struktur von Unterprogrammen	
- Verwendung von Parametern	
Kennen von Konzepten der objektorientierten Programmierung	Vererbung, Kapselung, Polymorphie → Kl. 7, LB 1 → Kl. 9, LB 1
- Klasse und Objekt	
- Attribut und Attributwert	
- Methode	
Übertragen des objektorientierten Paradigmas auf einfache Problemstellungen	
Kennen von Rekursion und Iteration	Fraktale
Beherrschen der Implementierung eigener Lösungen zu ausgewählten komplexen Problemstellungen	Auswahl des Programmierparadigmas Spiel, Simulation, Sortieralgorithmen, Suchverfahren ⇒ Problemlösestrategien
Kennen von Aspekten der Effizienz und Komplexität von Algorithmen	Komplexitätsklassen, Zeitkomplexität, Speicherkomplexität Sortieralgorithmen, Problem des Handlungsreisenden, Vierfarbenproblem, Brückenproblem, Primfaktorzerlegung
Einblick gewinnen in die Grenzen der Berechenbarkeit	algorithmische Unlösbarkeit, Rucksackproblem, Hamiltonkreis, Halteproblem

Anton Weigel,
Didaktik der Informatik, Universität Leipzig

2. Aufgabenstellung (so wie sie dem Prüfling vorgelegt wird):

Sortieralgorithmen

Aufgabe 1 (8 BE):

a) Nennen Sie zu den Sortieralgorithmen Bubblesort und Quicksort ihre jeweilige Speicherkomplexität, also den zusätzlich nötigen Speicherplatz, im WORST CASE.

b) Erläutern Sie kurz die Bedeutung der Begriffe in-place/out-of-place und stabil/instabil im Bezug auf Sortieralgorithmen. Ordnen Sie dann den Algorithmen aus a) jeweils in-place/out-of-place und stabil/instabil zu.

Aufgabe 2 (14 BE):

```
FUNCTION Bubblesort

        n = [          ]
        FOR i = 0 TO n - 2 DO
                FOR j = 0 TO n - i - 2 DO
                        IF [    ] > A[j + 1]
                        THEN TAUSCHE A[j] UND [          ]
                END FOR
        END FOR

END FUNCTION
```

Abbildung 2.1

a) In Abbildung 2.1 sehen Sie einen Pseudocode für einen einfachen Bubblesort-Algorithmus. Dieser enthält 3 Lücken (Kästchen). Füllen Sie die Lücken so aus, dass der Algorithmus korrekt wie ein Bubblesort-Algorithmus funktioniert.

b) Beschreiben Sie die Funktionsweise des korrigierten Algorithmus aus der Abbildung. Zeigen Sie dabei die wesentlichen Schritte des Verfahrens anhand folgender Liste von Zahlen, die von klein nach groß sortiert werden sollen: [3,1,5,9,2,7,4] Notieren Sie dabei das Ergebnis nach jedem Durchlauf der äußeren FOR-Schleife.

Aufgabe 3 (3 BE):

a) Beurteilen Sie die Eignung von Quicksort und Mergesort für die Sortierung sehr großer Datenmengen in Systemen mit begrenztem Speicher. Bedenken Sie dabei den Speicherbedarf beider Algorithmen.

3. Tabellarisches Erwartungsbild mit Angaben der jeweils erreichbaren BE und der Zuordnung zu den Anforderungsbereichen:

Aufgabe Nr.	Sachverhalt	AB1	AB2	AB3
1a)	Nennen der Speicherkomplexitäten	2	0	0
1b)	Zuordnen der Begriffe	6	0	0
2a)	Richtige Bezeichnung und Fehler finden	0	3	0
2b)	Bubblesort Beschreibung	0	10	0
3a)	Beurteilung	0	0	3
	Summe BE	8	13	3
	Gesamt	24		

4. Musterlösung mit Angabe der Zuordnung der einzelnen BE:

Sortieralgorithmen – Musterlösung

Aufgabe 1 (8 BE):

a) Je 1 BE für:
Bubblesort: 1
Quicksort: O(log n)

b) Bubblesort: stabil und in-place (je 1 BE)
Quicksort: instabil und in-place (je 1 BE)

Ein Sortieralgorithmus ist in-place, wenn er nur eine konstante Menge an zusätzlichem Speicherplatz benötigt, also O(1) zusätzlichen Speicher. Die Sortierung wird direkt im ursprünglichen Array oder in der ursprünglichen Datenstruktur durchgeführt. (1 BE)

Ein Sortieralgorithmus ist stabil, wenn die relative Reihenfolge von gleichwertigen Elementen (Elemente mit gleichem Schlüssel) im sortierten Array die gleiche ist wie im ursprünglichen Array. Mit anderen Worten, wenn zwei Elemente gleich sind, bleiben ihre Reihenfolge beibehalten. (1 BE)

Aufgabe 2 (14 BE):

a) Lücken:
   1. Dort müsste length(A) oder etwas vergleichbares stehen. (1 BE)
   2. Vor dem Vergleichsoperator muss A[j] stehen. (1 BE)
   3. An dieser Stelle müsste A[j + 1] oder etwas vergleichbares stehen. (1 BE)

b) Folgende Punkte (8 BE):
   - Länge der zu sortierenden Liste wir in n gespeichert (1 BE)
   - Die innere Schleife läuft von 0 aufsteigend, bzw. sie startet beim kleinsten Element mit den Vergleichen (1 BE)
   - Es wird jedes Element mit dem nächsten benachbarten verglichen (1 BE)
   - Falls das „linke“ Element größer ist als das „rechte“ wird getauscht (1 BE)
   - Nach jedem Durchlauf der äußeren Schleife ist das größtmögliche Element an der richtigen Stelle (also das größte ganz rechts, das zweitgrößte links neben dem größten …) (1 BE)
   - Auflistung der korrekten Ergebnisse nach den Durchläufen der äußeren Schleife: [3,1,5,2,7,4,9](Start) -> [1,3,5,2,7,4,9](i=0) -> [1,3,2,5,4,7,9](i=1) -> [1,2,3,4,5,7,9](i=2) -> für i=3 bis 5 die gleiche Liste
     - ✓ 1 BE auf die Liste nach dem i=0 Durchlauf
     - ✓ 1 BE auf die Liste nach dem i=1 Durchlauf
     - ✓ 1 BE auf die Liste nach dem i=2 Durchlauf
     - ✓ 2 BE auf die Erkenntnis, dass die Liste nach dem i=2 Durchlauf sortiert ist, der Algorithmus aber weiterläuft

Aufgabe 3 (3 BE):

a) Nennung des korrekten Speicherbedarfs beider Algorithmen; Quicksort O(log n); Mergesort O(n) (1 BE)
Quicksort ist besser geeignet für Systeme mit begrenztem Speicher (1 BE)
Begründung: O(log n) ist für große n besser bzw. kleiner als O(n) (1 BE)

5. Hinweise zur Umsetzung (benötigte Arbeitsmittel, ggf. Software auf dem Prüfungsrechner, …):

Bei der Auswahl dieser Aufgabe ist zu beachten:

- Papier für Notizen und Schreibutensilien
- Keine technischen Hilfsmittel

6. Anhang: Abbildungen:

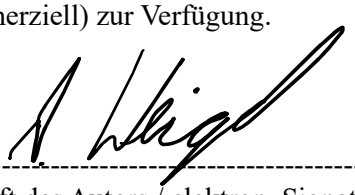
7. Quellenangabe, Abbildungsnachweise, …:

- Lehrplan Informatik Gymnasium

8. Erklärung der Freigabe zur Nachnutzung der Aufgabe:

Hiermit erkläre ich, Anton Weigel, diese Aufgabe unter Wahrung des Urheberrechts erstellt zu haben.

Ich stelle diese Aufgabe zur Nachnutzung nach Lizenz CC BY-NC (Namensnennung, Bearbeitung, nicht kommerziell) zur Verfügung.



(Unterschrift des Autors / elektron. Signatur)