universität wien

Algorithmen und Datenstrukturen 1

Theoretische Prüfung

1. Nachtermin

16.01.2023

Name:	
Matrikelnummer:	

Die Angaben sind beidseitig bedruckt!

	29	**34**	**22**	**18**	**32**	**19**	**39**
	+	**+**	**+**	**+**	**+**	**+**	**+**
33							
z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8

Aufgabe 1 [2]

Fügen Sie in obiger Tabelle in den leeren Kästchen, vor denen das Pluszeichen steht, die Ziffern Ihrer Matrikelnummer ein. Führen Sie die Additionen durch und ermitteln Sie die Zahlen z_2 bis z_8. (z_1 ist bereits mit dem fixen Wert 33 belegt.)

Aufgabe 2 [18]

Schreiben Sie zwei Funktionen in C++ ähnlicher Notation mit jeweils einem Parameter n (vom Typ int), deren Laufzeitkomplexität jeweils die Ordnung $\Theta(n^4 \log n)$ hat. Eine Funktion muss rekursiv sein, die andere nicht.
Zeigen Sie mithilfe des Mastertheorems, dass die Laufzeitkomplexität Ihrer rekursiven Funktion die gewünschte Ordnung hat.

Aufgabe 3 [20]

Die Werte z_1 bis z_8. (aus Aufgabe 1) seien in dieser Reihenfolge von links nach rechts in einem Array gespeichert. Sortieren Sie die Werte aufsteigend mit

a) [10] Quicksort (Verwenden Sie als Pivotelement immer den letzten - ganz rechten - Wert).
b) [6] Selectionsort
c) [4] Mergesort

Geben Sie alle notwendigen Schritte so genau an, dass die Arbeitsweise des Algorithmus klar ersichtlich wird.

Aufgabe 4 [20]

a) [9] Fügen Sie die Werte z_2 bis z_8 aus Aufgabe 1 (in dieser Reihenfolge) in eine zu Beginn leere Hashtabelle der Länge 7 ein. Verwenden Sie als Hashfunktion $h(k) = k\%7$ und double hashing zur Kollisionsbehandlung.
Die zweite Hashfunktion ist $g(k) = k\%5 + 1$.
Skizzieren Sie den Zustand der Hashtabelle nach jedem Einfügeschritt.
(Anmerkung: Werte können mehrfach in der Tabelle gespeichert werden. Die Tabelle ist statisch, wird also nicht vergrößert!)

b) [1] Löschen Sie den Wert z_3 aus der Tabelle und skizzieren Sie den Zustand der Hashtabelle.

c) [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert z_8 an.

d) [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert 50 an.

Aufgabe 5 [20]

a) [8] Fügen Sie die Werte z_1 bis z_8 aus Aufgabe 1 (in dieser Reihenfolge) in einen zu Beginn leeren Heap ein. (Werte können im Heap eventuell mehrfach gespeichert sein.) Skizzieren Sie den Zustand des Heaps nach jedem Einfügeschritt.

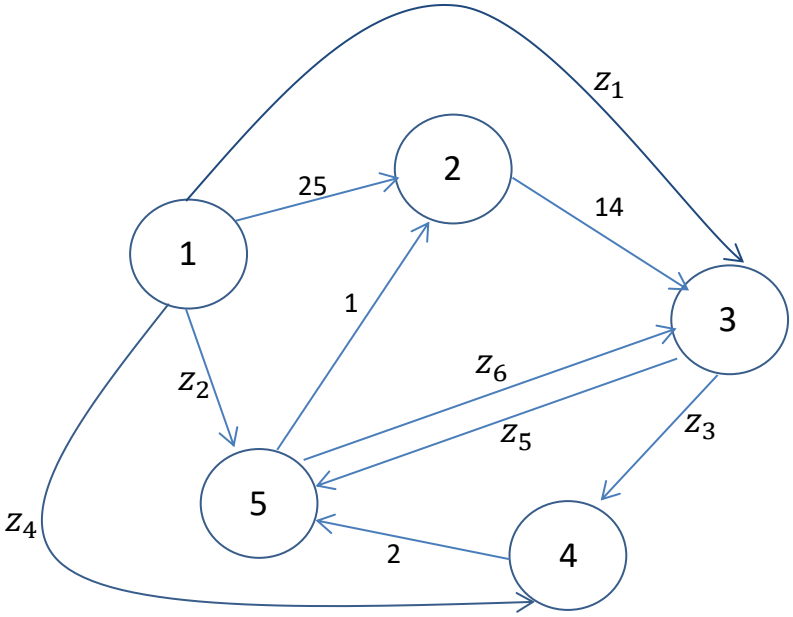
b) [4] Geben Sie in C++ ähnlicher Notation die Definition einer möglichst effizienten Datenstruktur für einen Heap an.

c) [4] Geben Sie in C++ ähnlicher Notation eine Definition einer möglichst effizienten Funktion an, die die Tiefe des Heaps in der Baumdarstellung ermittelt und retourniert.

d) [4] Geben Sie in C++ ähnlicher Notation eine Definition einer effizienten Funktion an, die ermittelt, ob es sich um einen Min-Heap oder einen Max-Heap handelt. Die Funktion soll true retournieren, wenn es ein Min-Heap ist, false sonst.

Aufgabe 6 [20]

Gegeben ist der folgende gerichtete Graph
(die Werte z_1 bis z_6. sind aus Aufgabe 1 zu übernehmen):



a) [3] Skizzieren Sie die Adjazenzliste des Graphen.
b) [10] Bestimmen Sie mit dem Algorithmus von Dijkstra die jeweils kürzesten Wege vom Knoten 1 zu allen anderen Knoten des Graphen.
c) [3] Ist der oben dargestellte Graph topologisch sortierbar? Falls ja, geben Sie eine topologische Sortierung an, andernfalls begründen Sie, warum eine solche nicht gefunden werden kann.
d) [4] Welche der folgenden Voraussetzungen ist hinreichend, damit der Dijkstra-Algorithmus das korrekte Resultat liefert? Zutreffendes bitte ankreuzen.

(1) Alle Kantengewichte des Eingabegraphen sind nicht-negativ.

(2) Der Eingabegraph enthält keinen negativen Kreis.

(3) Der Eingabegraph enthält einen negativen Kreis.

(4) Der Eingabegraph ist ein DAG.