Algorithmen I
Übungsklausur
11.07.2012

Name:

Aufgabe 1: Wissensfragen (6 Punkte)

(1) Nennen Sie zwei in der Vorlesung vorgestellte Algorithmen, die nach dem Greedy-Prinzip arbeiten. Begründen Sie kurz, warum. (1P)

(2) Nennen Sie einen Vorteil und einen Nachteil von Mergesort gegenüber Quicksort. (1P)

(3) Was bedeutet Stabilität bei Sortieralgorithmen? Nennen Sie eine Situation, in der man die Stabilität ausnutzen kann. (2P)

(4) Zeigen oder widerlegen Sie: Binäre Suche hat auf jeder Datenstruktur eine bessere Laufzeit als Lineare Suche. (2P)

Aufgabe 2: Heaps (12 Punkte)

(1) Gegeben sei das folgende Feld:

12	21	19	26	99	30	24	32	18	101	128

(a) Erläutern Sie, warum das Feld keinen gültigen Min-Heap darstellt. (1P)

(b) Reparieren Sie den Min-Heap mit dem aus der Vorlesung bekannten Verfahren. Geben Sie dabei den Heap nach jeder Änderung an (als Feld oder als Baum). (3P)

(2) (a) Warum lässt sich in einem Heap, egal wie gut er implementiert ist, das Minimum nicht in $\Theta(1)$ entfernen, sodass die entstehende Struktur immer noch ein Heap ist? (4P)
Hinweis: Betrachten Sie die Laufzeit einer Heapsort-Implementierung, wenn dies möglich wäre. Welcher Widerspruch ergibt sich daraus?

(b) Geben Sie eine Datenstruktur an, die die entsprechende Eigenschaft aus (a) hat. (3P)

(c) Warum ist dies kein Widerspruch zur Überlegung aus (a)? (1P)

Aufgabe 3: Sortieren (13 Punkte)

(1) Gegebenen sei der Sortieralgorithmus *Cocktail-Sort*. Um ein Array zu sortieren, durchschreitet der Algorithmus das Array zunächst von vorne nach hinten und vertauscht dabei zwei benachbarte Werte, wenn sie in der falschen Reihenfolge stehen. Ist der Algorithmus am Ende des Arrays angekommen, durchschreitet er das Array jetzt in umgekehrter Reihenfolge von hinten nach vorne (bewegt sich also „wieder zurück"). Dies wird so lange fortgesetzt, bis in einem Durchlauf keine Elemente mehr vertauscht werden.

(a) Wenden Sie Cocktail-Sort auf folgendes Array an, um die Elemente **aufsteigend** zu sortieren. Geben Sie dabei das Array nach jedem Schritt an. (3P)

4	2	1	3	7	3

(b) Geben Sie eine naive Implementierung von *Cocktail-Sort* in Pseudocode an. (7P)

(c) Geben Sie eine *Familie* von Zahlenarrays an, die von Bubblesort in $O(n^2)$ sortiert werden, aber von Cocktail-Sort in $O(n)$. (3P)

Aufgabe 4: Dynamische Programmierung (5 Punkte)

(1) Was ist die grundlegende Idee hinter der Methode der dynamischen Programmierung? (1P)

(2) Die Fakultätsfunktion $n!$ lässt sich folgendermaßen rekursiv definieren:
$0! = 1$
$n! = n \cdot (n-1)!$ für $n \geq 1$

(a) Geben Sie ein Programm in Pseudocode an, welches $n!$ mittels dynamischer Programmierung und Bottom-Up-Ansatz berechnet. (3P)

(b) Das Programm soll nun so modifiziert werden, dass es nach einmaligem Berechnen von $n!$ jeden Aufruf $k!$ mit $k \leq n$ in $O(1)$ bearbeiten kann. Beschreiben Sie eine Möglichkeit dafür. (1P)

Aufgabe 5: Datenstrukturen (12 Punkte)

(1) Für die folgenden Anwendungsfälle soll jeweils eine geeignete Datenstruktur ausgewählt werden. Geben Sie jeweils eine passende Datenstruktur an und begründen Sie Ihre Wahl:

(a) In einer Musiksammlung sollen häufig neue Musikstücke hinzugefügt, gelöscht und gesucht werden. Über die zukünftige Größe der Sammlung kann beim Anlegen noch keine Aussage gemacht werden. (1P)

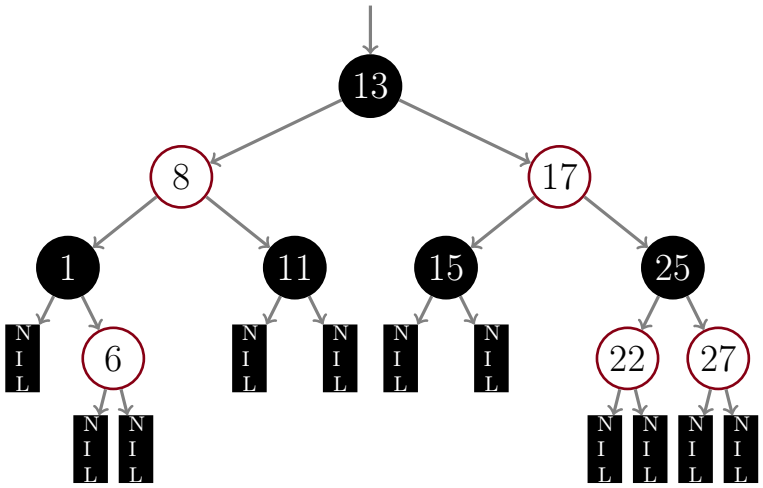
(b) An einen Datenbankserver werden zu manchen Zeitpunkten so viele Anfragen geschickt, dass er sie nicht sofort bearbeiten kann. Daher soll er die Möglichkeit bekommen, die Anfragen zwischenspeichern zu können, bis er wieder genügend freie Ressourcen besitzt. (1P)

(c) Ein Prozess-Scheduler in einem Betriebssystem arbeitet mit unterschiedlich hohen Prioritäten. Bei jedem Aufruf soll jeweils der Prozess mit der höchsten Priorität ausgeführt werden, wobei die Auswahlgeschwindigkeit für die Leistungsfähigkeit des Betriebssystems eine entscheidende Rolle spielt. (1P)

(2) Zeigen Sie, wie eine Warteschlange mit einer **einfach verketteten** Liste sowie zwei Zeigern K (ältestes Element) und E (neuestes Element) implementiert werden kann. Hierbei sollen sowohl ENQUEUE(X) als auch DEQUEUE() die Laufzeit O(1) besitzen. (4P)
Gehen Sie dafür folgendermaßen vor:

- Stellen Sie mit einer Skizze dar, wie die beiden Zeiger in der Liste positioniert werden müssen.
- Geben Sie den Pseudocode für beide Warteschlangenoperationen an. Stellen Sie dabei sicher, dass Ihre Implementierung auch korrekt mit einer leeren Warteschlange umgehen kann.

(3) Löschen Sie in folgendem Rot-Schwarz-Baum das Element mit dem Schlüssel 17 und zeichenen Sie den Baum nach Ablauf der Operation. Sie können dabei die schwarzen Knoten durch einen doppelten Kreis kennzeichnen, die roten durch einen einfachen. (5P)



Aufgabe 6: Graphen (12 Punkte)

(1) Ein Graph G = (V,E) sei durch die folgende Adjazenzfelddarstellung gegeben:

V :=

1	3	4	6	9	11

E :=

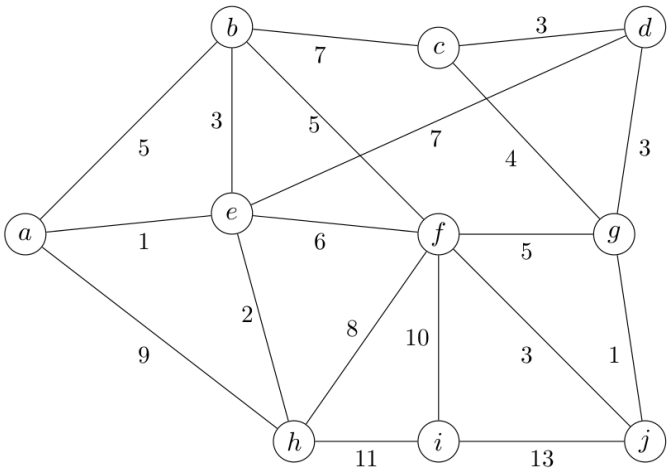
2	3	4	2	5	2	3	6	1	6	5

(a) Zeichnen Sie den Graphen. (2P)

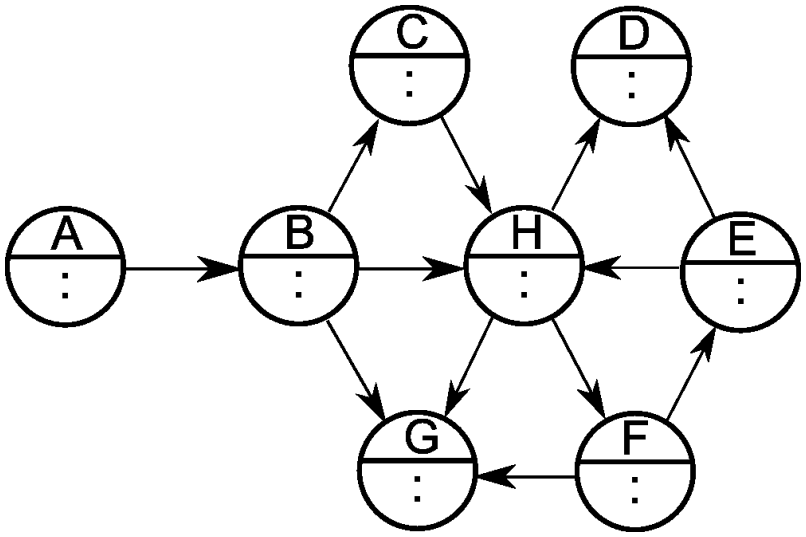
(b) Geben Sie den Graphen in Adjazenzlistendarstellung an. (1P)

(c) Welche dritte Möglichkeit zur Darstellung von Graphen wurde in der Vorlesung vorgestellt? Beschreiben Sie sie kurz und nennen Sie sowohl einen Vor- als auch einen Nachteil dieser Methode. (2P)

(2) Wenden Sie den Algorithmus von Prim auf dem folgenden Graphen an. Der Startknoten sei **a**. Markieren Sie die Kanten des resultierenden MSTs eindeutig (z.B. durch dicke Linien) und geben Sie die Reihenfolge der hinzugefügten Kanten an. (2P)



(3) Gegeben sei folgender Graph:



Führen Sie auf diesem Graphen eine Tiefensuche aus und gehen Sie dabei wie folgt vor:

- Beginnen Sie bei Knoten A.
- Betrachten Sie die ausgehenden Kanten eines Knotens bei der Suche jeweils im Uhrzeigersinn.
- Schreiben Sie die *discovered*-Zeiten der Knoten links neben die Doppelpunkte, die *finalized*-Zeiten jeweils rechts daneben.
- Klassifzieren Sie die Kanten. Kennzeichnen Sie dabei:
  - jede *Baumkante* durch eine dicke Linie
  - jede *Rückwärtskante* mit einem "B"
  - jede *Vorwärtskante* mit einem "F"
  - jede *Querkante* mit einem "C"

(5P)