FAU, Informatik 2, AUD-Team
aud@i2.cs.fau.de

11. Übung

Abgabe bis 29.01.2018, 10:00 Uhr

Einzelaufgabe 11.1: Blätter, Wurzeln, Äste, Bäume, Wälder — 29 EP

Betrachten Sie in den folgenden Teilaufgaben die Buchstaben und Sonderzeichen der Zeichenketten „*AuDi$!Fun*“ und „*Noot?;)*“ symbolweise (ohne Anführungszeichen) und in Leserichtung mit Ordnung gemäß ASCII-Tabelle (also z.B. $A < D < a < u$). Sofern nichts anderes gefordert, zeichnen Sie bitte nur den Endzustand nach jeder Teilaufgabe.

a) Fügen Sie „*AuDi$!Fun*“ in einen *linksvollständigen*, ansonsten aber allgemeinen Binärbaum ein. Beachten Sie, dass auch Zwischenstände (z.B. nach „*AuD*“) linksvollständig sein müssen!

b) Welche Höhe haben jeweils die Knoten mit den Buchstaben „*u*“ bzw. „*D*“ und welche Höhe hat der gesamte Baum?

c) Ergänzen Sie den vorangehend erhaltenen und weiterhin allgemeinen aber linksvollständigen Binärbaum um „*Noot?;)*“.

d) Fügen Sie nun „*AuDi$!Fun*“ in einen binären *Suchbaum* ein.

e) Ergänzen Sie den vorangehend erhaltenen binären *Suchbaum* um „*Noot?;)*“.

f) In welcher Reihenfolge müssten Sie die Zeichen des Wortes „*AuDi$!Fun*“ in einen binären *Suchbaum* einfügen, damit der Baum *maximale* Höhe hat? Zeichnen Sie diesen Baum!

g) In welcher Reihenfolge müssten Sie die Zeichen des Wortes „*AuDi$!Fun*“ in einen binären *Suchbaum* einfügen, damit der Baum *minimale* Höhe hat? Zeichnen Sie diesen Baum!

h) Fügen Sie „*AuDi$!Fun*“ in einen *AVL-Baum* ein. Zeichnen Sie den Baum *mit* Balancefaktoren *unmittelbar nach* dem Einfügen eines *jeden einzelnen* Buchstabens. Im Falle einer Rebalancierung zeichnen Sie den Baum auch nochmal nach der Rotation und geben Sie jeweils die Art der Rotationen an (einfach oder doppelt, Links- und/oder Rechts-Rotation[1]...)! Geben Sie auch den finalen *AVL-Baum* an.

i) Fügen Sie „*AuDi$!Fun*“ in eine *Min-Halde* („kleinster“ Buchstabe oben) ein. Bedenken Sie, dass die aus der Vorlesung bekannte Array-Einbettung dafür sorgt, dass der Heap jederzeit linksvollständig und partiell geordnet ist. Zeichnen Sie den Baum einmal nach „*AuDi$!*“ und am Schluss.

j) Fügen Sie „*Noot?;)*“ zeichenweise in eine *Max-Halde* („größter“ Buchstabe oben) ein. Zeichnen Sie den Baum jeweils nach dem vollständigen Wiederherstellen der Heap-Eigenschaft für jedes einzelne Zeichen und schließlich am Schluss. Markieren Sie jeweils die Knoten des Pfades, entlang dessen das neue Zeichen dabei „nach oben gesickert“ ist.

Geben Sie Ihre Bäume als `Baum.pdf` über EST ab.

[1] Je nach AVL-Literatur bezeichnet eine Links/Rechts-Rotation jeweils eine Rotation gegen/im Uhrzeigersinn.

Einzelaufgabe 11.2: Repeated Partial Extraction (RPE-Sort) — 10 EP

In dieser Aufgabe sollen Sie den im folgenden beschriebenen Sortieralgorithmus *RPESort* (Repeated Partial Extraction) implementieren. Das Verfahren eignet sich besonders gut, wenn schon große Teile der zu sortierenden Daten in der korrekten Reihenfolge vorliegen. Es funktioniert so:

- Seien $\mathcal{I}$ die Eingabe- und $\mathcal{O}$ die Ausgabeliste (Ergebnis). $\mathcal{O}$ ist zu Beginn leer.
- $\mathcal{I}$ wird iterativ bearbeitet. In jedem Durchgang wird zunächst das erste Element e_0 aus $\mathcal{I}$ entnommen und einer temporären Liste $\mathcal{T}$ hinzugefügt. Anschließend werden nach und nach diejenigen $e_i, e_j, e_k, \ldots$ aus $\mathcal{I}$ entfernt und an $\mathcal{T}$ angehängt, die jeweils größer als das zuletzt entfernte Element sind, so dass $e_0 \preceq e_i \preceq e_j \preceq e_k \preceq \ldots$ für eine Ordnungsrelation $\preceq$ gilt.
- Nach jedem Durchgang werden $\mathcal{T}$ und $\mathcal{O}$ so verschmolzen, dass am Ende alle Elemente beider Listen in $\mathcal{O}$ aufsteigend sortiert sind, während $\mathcal{T}$ leer ist. Da beide Listen prinzipbedingt stets sortiert sind, ist diese Operation in linearer Zeit $\mathcal{O}(t + o)$ möglich ($t = |\mathcal{T}|$, $o = |\mathcal{O}|$).
- Ist $\mathcal{I}$ noch nicht leer, erfolgt ein weiterer Durchgang. Sobald $\mathcal{I}$ leer ist, befinden sich alle Elemente in der korrekten Reihenfolge sortiert in der Ergebnisliste $\mathcal{O}$.

ACHTUNG: Es sind **sämtliche** Aufrufe in die Java-API **untersagt**, abgesehen von der Verwendung von `Comparator`, `LinkedList` und `ListIterator`/`Iterator`! Programmieren Sie grundsätzlich defensiv, aber werfen Sie *keine* Ausnahmen in Ihrem Code (ggf. sind stattdessen leere Listen zurückzugeben)! Ihre Klasse darf **genau ein** Attribut und **keine** zusätzlichen Methoden deklarieren! Geben Sie Ihre Lösung als `RPESorter.java` über EST ab.

Gruppenaufgabe 11.3: Sortierverfahren – Theorie — 21 GP

Betrachten Sie in den folgenden Teilaufgaben die Buchstaben und Sonderzeichen der Zeichenkette „*AuDi$!Fun!?*“ symbolweise (ohne Anführungszeichen) und in Leserichtung mit Ordnung gemäß ASCII-Tabelle (also z.B. $\$ < A < D < a < u$). Verwenden Sie jeweils die in den Beispielen gezeigte Darstellung! ***Achtung:*** Verwenden Sie ***exakt*** die in der Vorlesung beschriebenen Varianten der Sortierverfahren!

a) Sortieren Sie die Zeichen zuerst *absteigend* und *stabil* mittels *Sortieren durch Auswählen*. Löschen Sie dazu jeweils das Maximum aus der noch zu sortierenden Buchstabenliste l und fügen Sie es hinten an die anfangs leere Ergebnisliste l_s an. *Beispiel:*

l_s	l		
	B	a	r
r		B	a
r	a		B
r	a	B	

b) Sortieren Sie die Zeichen jetzt *aufsteigend* mittels *Sortieren durch Einfügen*. Stellen Sie den Ablauf wie vorhin tabellarisch dar.

c) Sortieren Sie die Zeichen nun wieder *absteigend* aber *in-situ* mittels *Blasensortierung*. Stellen Sie den Ablauf ebenfalls in einer Tabelle (diesmal *ohne* Spalte l_s, da *in-situ*!) dar.

d) Sortieren Sie die Zeichen jetzt wieder *aufsteigend*, diesmal aber mittels *Haldensortierung*. Verwenden Sie dazu die Array-Einbettung einer *Min-Halde* (d.h. *kleinstes* Zeichen „oben“, also bei $l[0]$). Stellen Sie den Ablauf in *zwei Phasen* dar: Die erste zeigt den initialen Aufbau

der *Min-Halde* in l (dabei bleibt l_s leer) und die zweite den eigentlichen Sortiervorgang, bei dem das Halden-Minimum jeweils ans Ende von l_s verschoben und anschließend die Min-Halden-Eigenschaft wiederhergestellt wird. Pro Zeile der ersten Phase wird jeweils ein Element „versickert“. *Beispiel:*

l_s	l				
	F	A	I	L	
	A	F	I	L	
A		F	L	I	
A	F		I	L	
A	F	I		L	
A	F	I	L		

e) *Sortieren* Sie die Zeichen *absteigend durch Verschmelzen*. Führen Sie den rekursiven Abstieg zuerst in der linken und danach in der rechten Hälfte des jeweils betrachteten Intervalls. Geben Sie jeweils an, ob Sie ein Intervall gerade zerlegen (S) oder verschmelzen (M). *Beispiel:*

Aktion	l				
	F	a	i	l	
S	F	a		i	l
S	F		a	i	l
M	a	F		i	l
S	a	F	i		l
M	a	F	l	i	
M	l	i	a	F	

Geben Sie Ihre Lösung als `Sortieren.pdf` über EST ab.

39 EP + 21 GP = 60 Punkte