Hausaufgabe 3

Aaron Sastry

May 2, 2022

Aufgabe 3.1 Radix-Sort

a)

Sortieren Sie die Folge a = (271, 842, 172, 828, 904, 023, 991, 800, 601, 889) mit LSD– RadixSort

	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
1st 10 buckets	800	271 991 601	842 172	023	904				828	889

output 1: (800, 271, 991, 601, 842, 172, 023, 904, 828, 889)

	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
2nd 10 buckets	800 601 904		023 828		842			271 172	889	991

output 2: (800, 601, 904, 023, 828, 842, 271, 172, 889, 991)

	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
3rd 10 buckets	023	172	271				601		800 828 842 889	904 991

output 3: (023, 172, 271, 601, 800, 828, 842, 889, 904, 991)
Nach 3 Iterationen ist die List final sortiert mit der Reihenfolge:
023, 172, 271, 601, 800, 828, 842, 889, 904, 991

b)

Sortieren Sie die Folge b = (0.271, 0.842, 0.172, 0.828, 0.904) mit MSD–RadixSort.

1st set of buckets

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
	0.172	0.271						0.842 0.828	0.904

output: 1 = ([0.172], [0.271], [0.842, 0.828], [0.904])

Jetzt werden nur noch die Buckets sortiert die mehr als 1 Element haben.

2nd set of buckets, (the 0.8 bucket)

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
		0.828		0.842					

output: ([0.172], [0.271], [[0.828], [0.842]], [0.904])

Zu guter Letzt werden jetzt noch die Buckets wieder zu einer fertigen Liste zusammen gesetzt:

$\Longrightarrow$ ***0.172, 0.271, 0.828, 0.842, 0.904***

c)

Angenommen, Sie wollen m-stellige Binärzahlen mit MSD-RadixSort sortieren, wobei die Behälter rekursiv wieder mit MSD-RadixSort sortiert werden sollen. Was ist die Laufzeit dieses Sortieralgorithmus für n Binärzahlen, abhängig von n und m? Begründen Sie Ihre Behauptung kurz.

Antwort:
Aus jeder liste entstehen immmer 2 buckets auf denen wieder sortiert werden muss. Somit sind in der iteration i etwa : 2^i befüllte Buckets. Da alle Buckets vereinigt wieder genau n Elemente haben ist jede Iteration in

$$T = n$$

Nun gibt es natürlich etwa 2^m Buckets und exakt m operationen da dies die Anzahl der stellen sind in die wir einteilen. Somit erfolgt die Einteilung in die gesammten Buckets in

$$T = m * n$$

Zu guter Letzt fehlt noch die vereinigung der Buckets wieder in eine vollsändige List, was wieder genau $T = n$ braucht. Somit ist die finale Laufzeit bei:

$$T = (m + 1) * n \Longleftrightarrow T = m * n + n$$

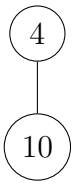
$$\Longrightarrow \mathcal{O}(m * n)$$

Aufgabe 3.2 Heaps

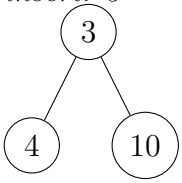
a) Fügen Sie die Werte 10, 4, 3, 15, 21, 2, 8, 11 und 1 in einen anfangs leeren Heap ein. Stellen Sie nach jeder Einfüge-Operation den Heap als Baum dar und geben Sie das Array an, welches dem fertigen Heap entspricht.

insert: 10

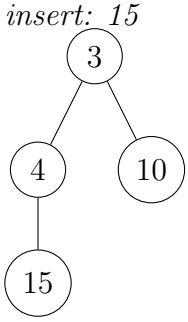
insert: 4



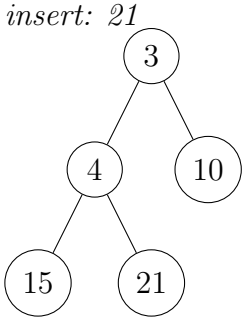
insert: 3



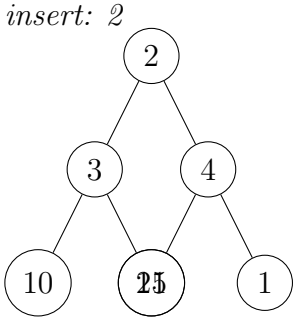
insert: 15



insert: 21



insert: 2



b) Führen Sie auf dem soeben gebauten Heap zwei deleteMin-Operationen durch und geben Sie jeweils den resultierenden Heap in Baumdarstellung und als Array an.

c) Beschreiben Sie einen Algorithmus, der k sortierte Listen mit Gesamtlänge n in O(n log k) Zeit zu einer sortierten Liste zusammenfügt. Benutzen Sie dabei einen Heap. Begründen Sie kurz, dass Ihr Algorithmus die Laufzeitschranke einhält.

d) Gegeben sei die folgende alternative Prozedur zum Erstellen eines binären Heaps für ein unsortiertes Array A[1..n]: buildHeapInsert(A : Array): 1 for i ← 1 to n do 2 insert(A[i]) Geben Sie ein Beispiel für eine Eingabe an, sodass buildHeapInsert eine schlechtere Laufzeit für das Aufbauen des Heaps hat als O(n). Was ist die worst–case Laufzeit von buildHeapInsert? Begründen Sie!