DHBW Lörrach
Duale Hochschule
Baden-Württemberg

Algorithmen und Komplexität
TIF 21A/B
Dr. Bruno Becker

Übungsblatt 4: Heaps & Hashing

Übungsblatt 4 –Aufgabe 1

Thema Heap-Datenstruktur: Erstellen Sie für Algorithmen zum

a) Einfügen und
b) Maximum entfernen in einer Heap-Datenstruktur
c) Was ist der erwartete und worst-case-Aufwand für diese Operationen?

a) und b) -> Literatur, z.B. Ottmann/Widmayer Kap 2.3

Einfügen: („*Hochschwimmen*“)

1. Element an n+1.Stelle einfügen. i=n+1
2. Falls Element i nicht Wurzel Schlüssel des Elements i mit Schlüssel Vater (d.h. a[i/div 2] vergleichen
3. Falls Heap-Bedingung verletzt, mit Element i mit Vater vertauschen und i= IndexVater; Goto 2;
4. Sonst fertig

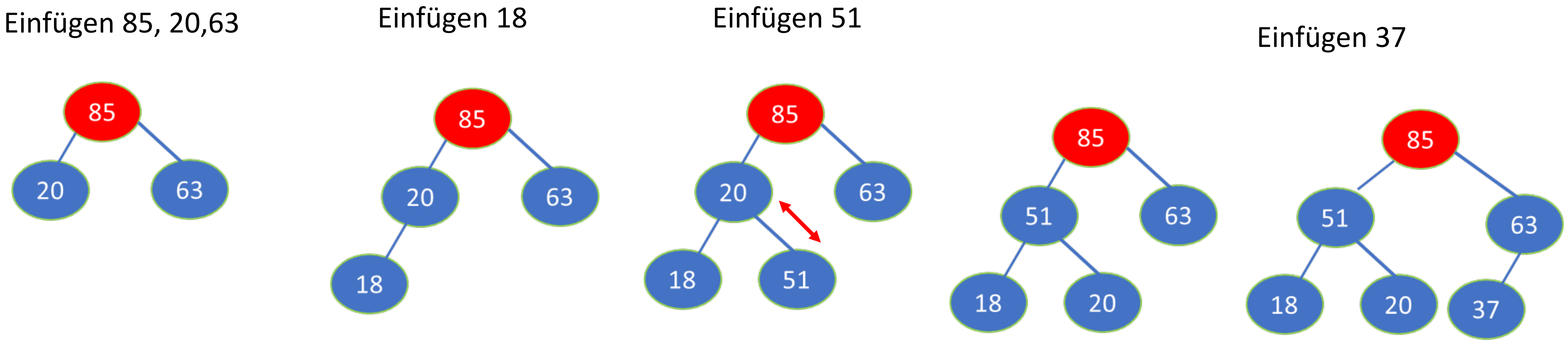
Löschen Maximum: *(„Versickern“)*

1. Element von n-ter-Stelle an die Wurzel (d.h. a[1]) setzen. i=1;
2. Falls Element i nicht Blatt Vergleiche Element i mit beiden Söhnen (d.h. a[i] mit a[2*i] oder a[2*i+1])
3. Falls Heap-Bedingung verletzt, mit dem größeren Sohn vertauschen und i= Index größerer Sohn; Goto 2;
4. Sonst fertig

c) Worst Case und Average case für Einfügen und Maximum entfernen jeweils O(log n))

Übungsblatt 4 –Aufgabe 2a

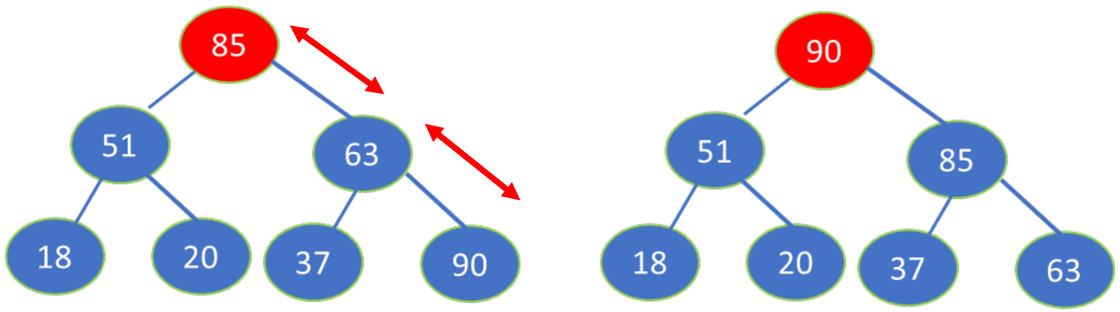
a) Gegeben sei die Folge der Schlüssel
85, 20, 63, 18, 51, 37, 90, 33.
Erzeugen Sie für diese Folge einen Heap und stellen Sie Ihn als Binärbaum dar.



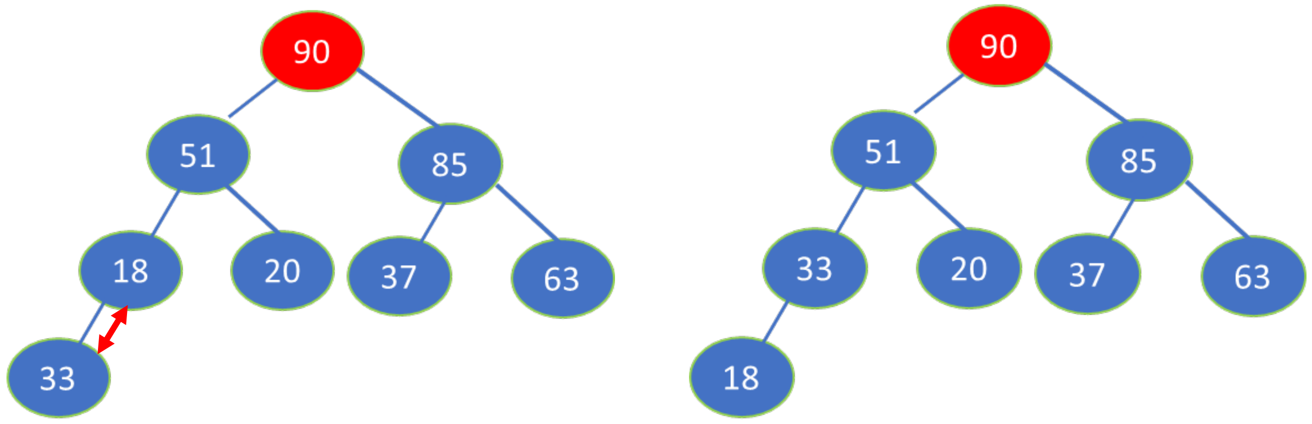
Übungsblatt 4 –Aufgabe 2a

a) Gegeben sei die Folge der Schlüssel
85, 20, 63, 18, 51, 37, 90, 33.
Erzeugen Sie für diese Folge einen Heap und stellen Sie Ihn als Binärbaum dar.

Einfügen 90



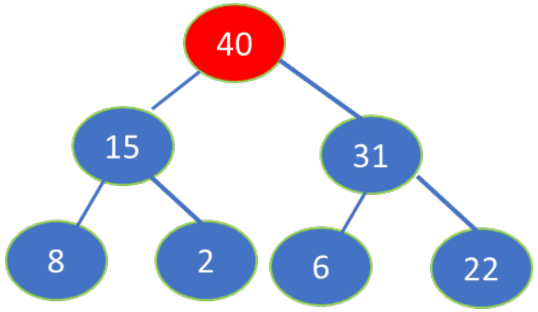
Einfügen 33



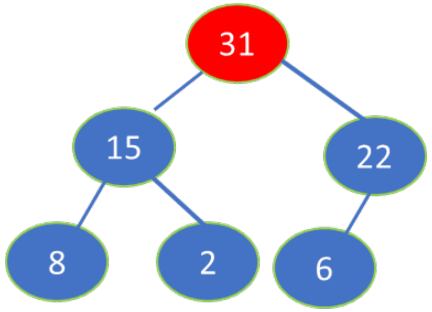
Übungsblatt 4 –Aufgabe 2b

Sortieren Sie die Folge der Schlüssel
40, 15, 31, 8, 2, 6, 22 aufsteigend mit Heapsort.
Stellen Sie zunächst einen Heap her und geben Sie dann jede Belegung nach einem Schlüsseltausch an.

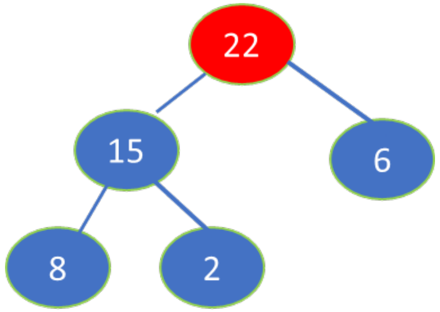
Heap erzeugen
a[1]-a[7]



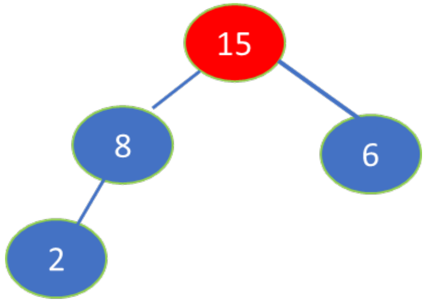
Löschen 40
22 in Wurzel-
versickern
a[7] = 40



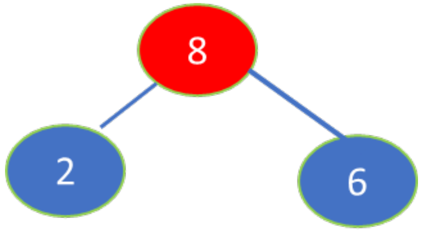
Löschen 31
6 in Wurzel-
versickern
a[6] = 31



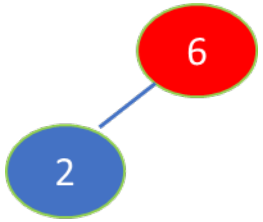
Löschen 22
2 in Wurzel-
versickern
a[5] = 22



Löschen 15
2 in Wurzel-
versickern
a[4] = 15



Löschen 8
6 in Wurzel-
versickern
a[3] = 8



Löschen 6
2 ist Wurzel
a[2] = 6; a[1]=2

Übungsblatt 4 –Aufgabe 2c

Geben Sie größenordnungsmäßig die Komplexität der folgenden Operationen auf einem Heap mit *N* Elementen im schlimmsten Fall an. Am Ende jeder Operation soll stets ein gültiger Heap zurückbleiben.

I. Einfügen eines beliebigen Elementes	O (log N)
II. Suchen des Maximums	O(1) für Maxheap
III. Suchen eines beliebigen Elementes	O(N)
IV. Suchen des Minimums	O(N) für Maxheap
V. Entfernen des Maximums	O(log N) für Maxheap

Übungsblatt 4 –Aufgabe 3

Geben Sie die Belegung einer Hashtabelle der Größe 13 an, wenn die Schlüssel
5, 1, 19, 23, 14, 17, 32, 30, 2
in die anfangs leere Tabelle eingefügt werden und offenes Hashing mit Hashfunktion *h(k) = k mod 13* verwendet wird mit

a) Linearem Sondieren

b) Quadratisches Sondieren.

a) Lineares Sondieren nach links

0	1	2	3	4	5	6	7	8	9	10	11	12
14	1	30	32	17	5	19				23		2

b) Quadratisches Sondieren: 1 nach links, 1 nach rechts, 2^2 nach links, 2^2 nach rechts, 3^2 nach links, 3^2 nach rechts,….

0	1	2	3	4	5	6	7	8	9	10	11	12
14	1	2	30	17	5	19	32			23		

Übungsblatt 4- Aufgabe 3c

Welche Kosten sind für eine erfolgreiche Suche bei beiden Sondierungsverfahren zu erwarten, wenn nach jedem Schlüssel mit gleicher Wahrscheinlichkeit gesucht wird?

Erfolgreiche Suche Schlüsselvergleiche – Schwarze Zahlen stehen an richtiger Stelle => 1 Vergleich. Rote Zahlen:

0	1	2	3	4	5	6	7	8	9	10	11	12
14	1	30	32	17	5	19				23		2

Lineares Sondieren: 14: 2 Vergleiche, 30 3 Vergleiche, 32 4 Vergleiche, 2 4 Vergleiche
Summe: 13 Vergleiche für rote Zahlen + 5 schwarze Zahlen = 18.
=> Durchschnitt pro Suche 18/9 = 2 Vergleiche

0	1	2	3	4	5	6	7	8	9	10	11	12
14	1	2	30	17	5	19	32			23		

Quadratrisches Sondieren: 14: 2 Vergleiche, 30 2 Vergleiche, 32 3 Vergleiche

- Summe: 7 Vergleiche für rote Zahlen + 6 schwarze Zahlen = 13.
  => Durchschnitt pro Suche 13/9 ca. 1,44 Vergleiche

Übungsblatt 4 – Aufgabe 4

Wieviele Schritte werden im schlechtesten Fall benötigt, um in eine anfangs leere Hashtabelle *n* Schlüssel einzufügen, wenn zur Überlaufbehandlung die Methode der separaten Verkettung verwendet wird und die Überläufer

a) In unsortierten Listen abgespeichert werden?

b) In sortierten Listen abgespeichert werden?

c) Wie viele Schritte benötigt man in beiden Fällen, um nach jedem der n eingefügten Schlüssel einmal zu suchen?

Schlechtester Fall: Alle Schlüssel haben gleichen Hash-Wert

a) Lineare unsortierte Liste: Jedes Einfügen vorne in Liste $O(1)$, n mal Einfügen $O(n)$

b) Lineare sortierte Liste: Jedes Einfügen $O(n)$, n mal Einfügen $O(n^2)$!

c) 1 Element an 1ster Stelle in Liste, 1 an 2ter, 1 an 3ter,…, 1 an n-ter Stelle
=> Suche $1+2+\ldots+n-1 = n*(n-1)/2 = O(n^2)$.
Gleiche Begründung gilt für sortierte Liste