DHBW Lörrach
Duale Hochschule
Baden-Württemberg

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

Übungsblatt 9: Vermischtes

Übungsblatt 9 –Aufgabe 1

a) Gegeben sei die folgende Zahlenfolge

41	62	13	84	35	96	57	28	79

Sortieren Sie diese Folge aufsteigend mit Quicksort. Verwenden Sie das linke Elementals Pivotelement. Notieren Sie die Zwischenschritte des Algorithmus.

1. qs(0,8)

41	~~62~~	13	~~84~~	~~35~~	96	57	~~28~~	79
	28		35	84			62	
35	28	13	41	84	96	57	62	79

2. qs(0,2)

35	28	13
13	28	35

3. qs(0,1)

13	28
13	28

13

4. qs(1,1): 28

35

41

5. qs(4,8)

84	79 ~~96~~	57	62	96 ~~79~~
62	79	57	84	96

6. qs(4,6)

62	57 ~~79~~	79 ~~57~~		96
57	62	79		96

7. qs(5,5): 57

62

8.qs(6,6): 79

84

9. qs(8,8) 96

Übungsblatt 9 –Aufgabe 1

b) Für welche Folgen ist Quicksort besonders ungünstig? Begründen Sie Ihre Antwort.

Wenn Quicksort schon (fast) sortiert oder umgekehrt sortiert ist $O(N^2)$

c) Was kann man dagegen tun?

1. Folge Durchschütteln (randomisieren)
2. Pivotelement als mittleres Element einer Menge z.B. links, Mitte, rechts wählen

Übungsblatt 9 –Aufgabe 2

Gegeben sei eine lineare Liste mit einer geraden Anzahl an Elementen (> 0). Jedes Element (*node*) besitzt einen Zeiger auf den Nachfolger (*node.next*) und einen Schlüssel (*node.key*). Der Zeiger *first* zeigt auf den Listenanfang.

a) Erstellen Sie Sie eine Methode, die jedes ungerade Element mit seinem Nachfolger vertauscht, d.h. aus *first*-> 1->2->3->4->5->6 wird *first*-> 2->1->4->3->6->5

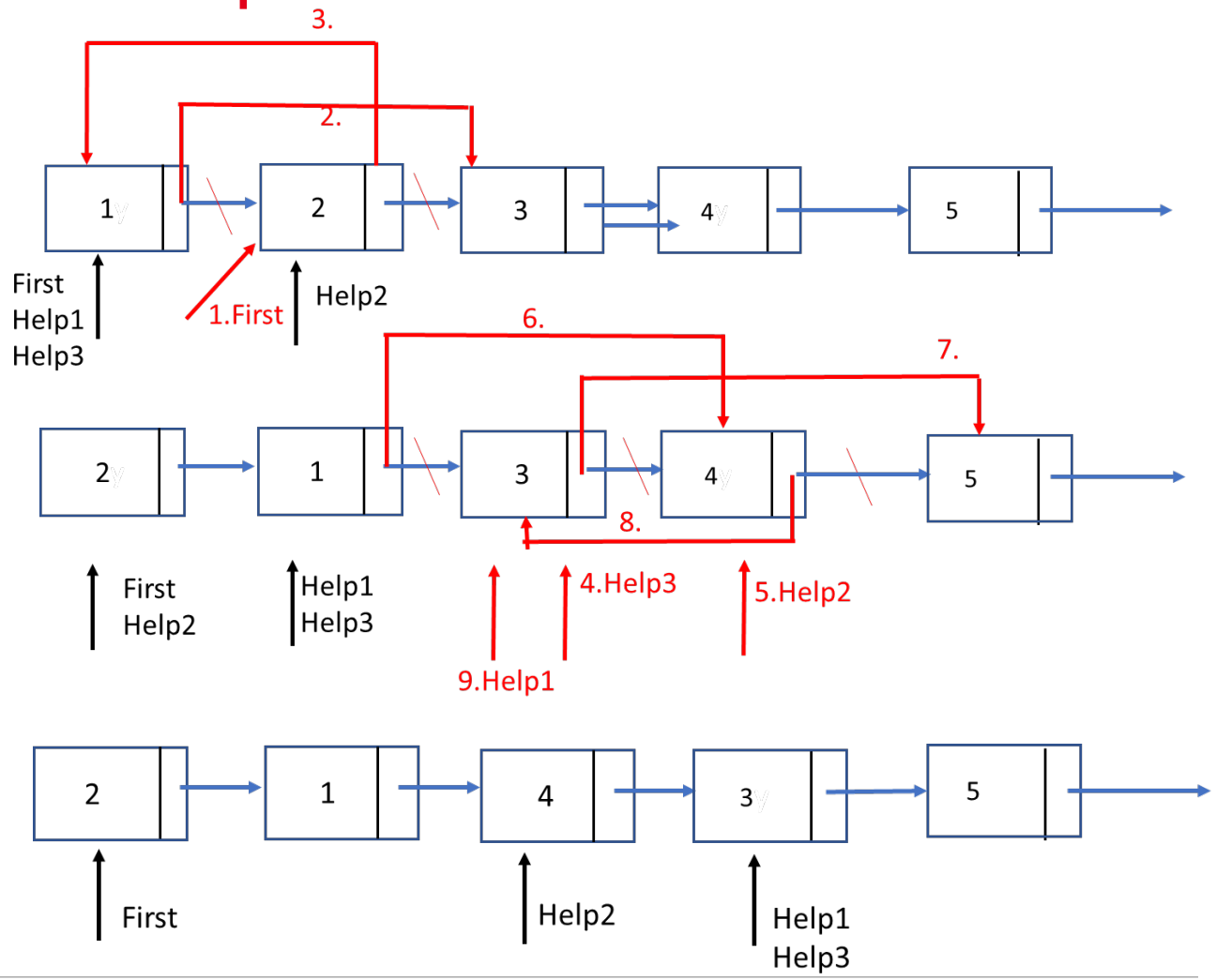
b) Welchen Aufwand benötigt die Methode in Abhängigkeit der Anzahl der Listenelemente (in O-Notation)?

```
public void umdrehen (first node);
{ private node help1 = first;
  private node help2 =help1.next;
  private node help3 = first;
1.  first = help2;  // Listenanfang auf 2. Element
//1. Paar vertauschen
2.   help1.next = help2.next;
3.    help2.next = help1;
  while (help1.next != Null)
 { // 2.tes bis n/2-tes Paar vertauschen, help1 steht vor dem zu vertauschenden Paar
  4. help3 = help1.next;  //help 3 zeigt auf 1.Element des zu vertauschenden Paares
  5 .help2 = help3.next;  //help 2 zeigt auf 2.Element des zu vertauschenden Paares
   6.help1.next = help2;
   7.help3.next = help2.next;
    8.help2.next = help3;
    9. help1 = help3; // help1.next zeigt auf 1. El. des nächsten zu vertauschenden Paares oder  NULL
  }
}
```

b) O(N)

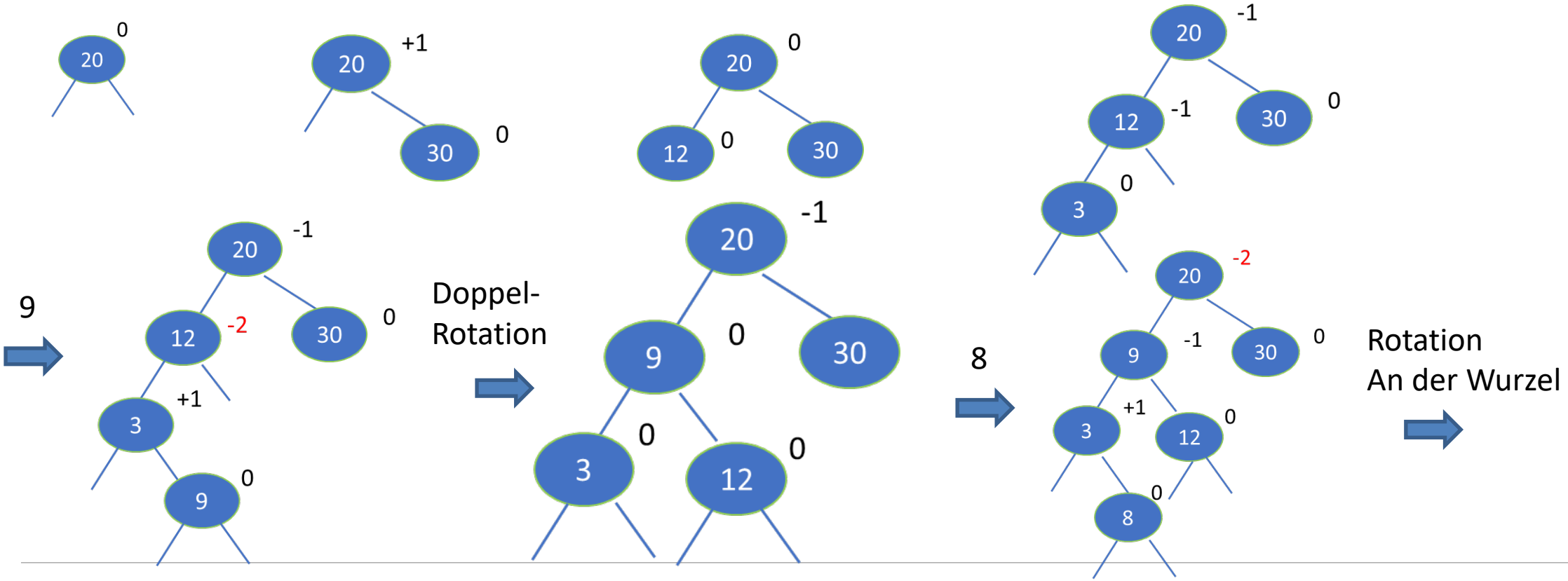
Übungsblatt 9 –Aufgabe 2 - Beispiel

```
public void umdrehen (first node);
{ private node help1 = first;
  private node help2 =help1.next;
  private node help3 = first;
1.  first = help2;  // Listenanfang auf 2. Element
//1. Paar vertauschen
 2.   help1.next = help2.next;
 3.    help2.next = help1;
  while (help1.next != Null)
  { // 2.tes bis n/2-tes Paar vertauschen, help1 steht vor dem zu
vertauschenden Paar
  4. help3 = help1.next;  //help 3 zeigt auf 1.Element des zu
vertauschenden Paares
  5 .help2 = help3.next;  //help 2 zeigt auf 2.Element des zu
vertauschenden Paares
   6.help1.next = help2;
   7.help3.next = help2.next;
    8.help2.next = help3;
   9. help1 = help3; // help1.next zeigt auf 1. El. des nächsten zu
vertauschenden Paares oder  NULL
  }
}
```



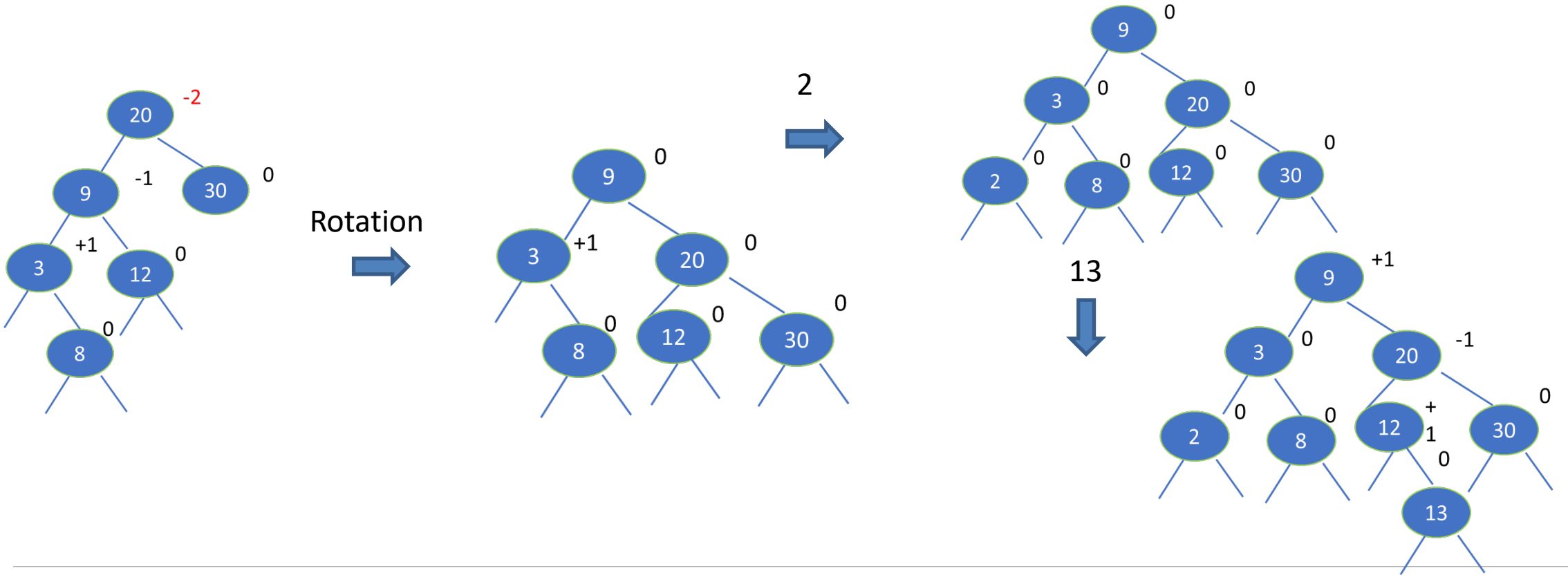
Übungsblatt 9 –Aufgabe 3

a) Geben Sie den AVL-Baum an, der durch Entstehen der Schlüssel 20,30,12,3,9,8,2,13 entsteht. Aktualisieren Sie nach jedem Einfügen den Baum (inkl. Balance-Informationen).



Übungsblatt 9 –Aufgabe 3

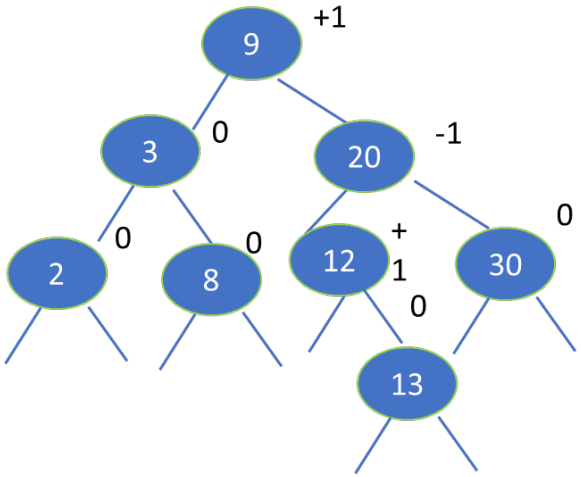
a) Geben Sie den AVL-Baum an, der durch Entstehen der Schlüssel 20,30,12,3,9,8,2,13 entsteht. Aktualisieren Sie nach jedem Einfügen den Baum (inkl. Balance-Informationen).



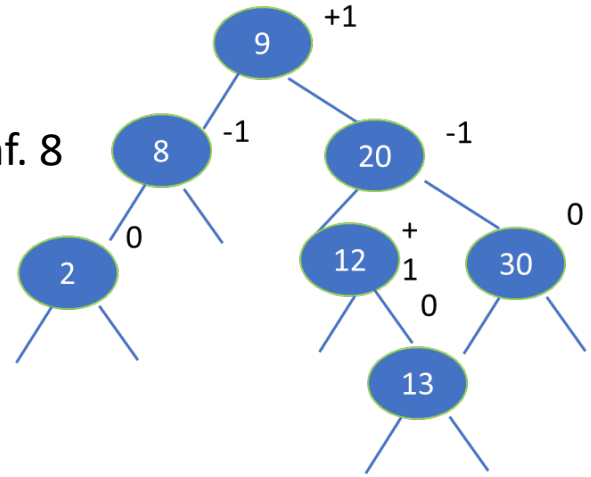
Übungsblatt 9 –Aufgabe 3

b) Löschen Sie nacheinander die Schlüssel 3 und 30 aus dem AVL-Baum

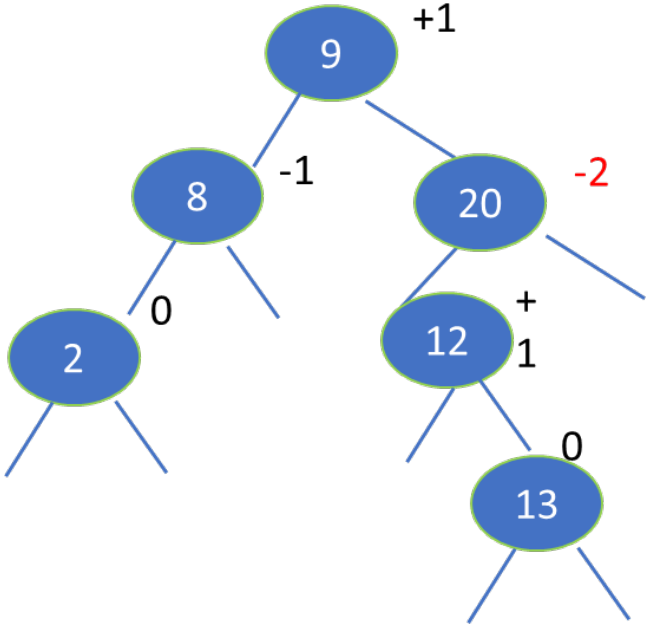
c) Geben Sie den AVL-Baum nach den Löschungen aus b) in Postreihenfolge aus.



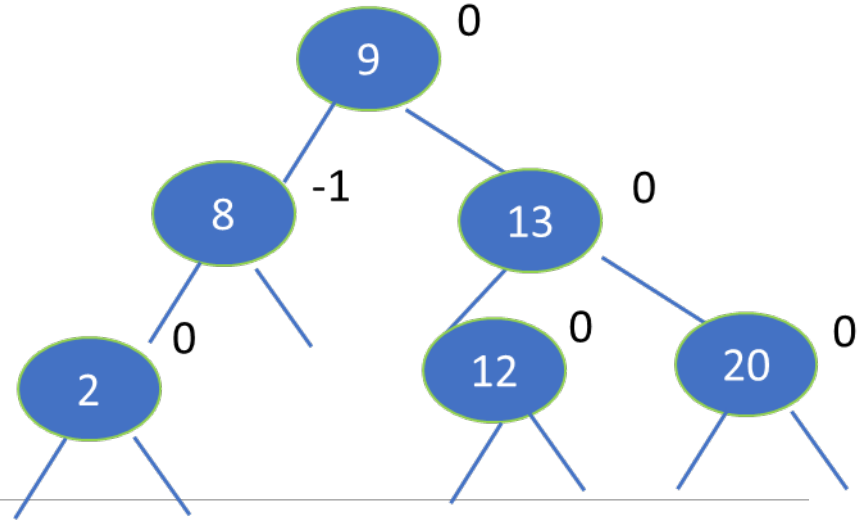
Lösche 3,
Symm Nachf. 8



Lösche 30,



Doppel-
Rotation

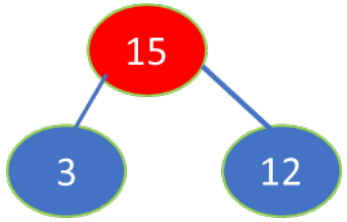


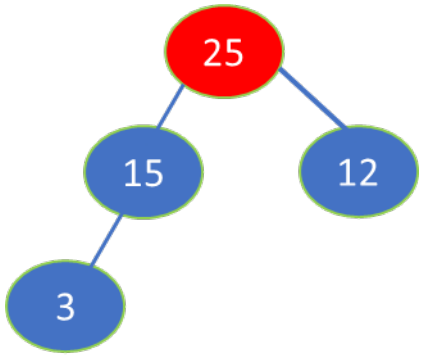
c) Postreihenfolge 2-8-12-20-13-9

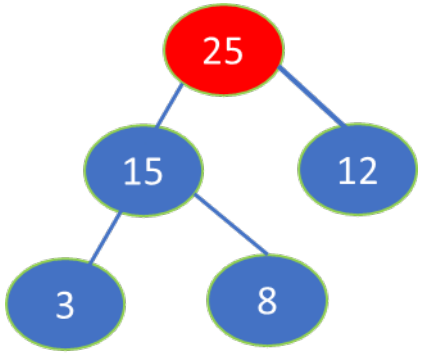
Übungsblatt 9 –Aufgabe 4

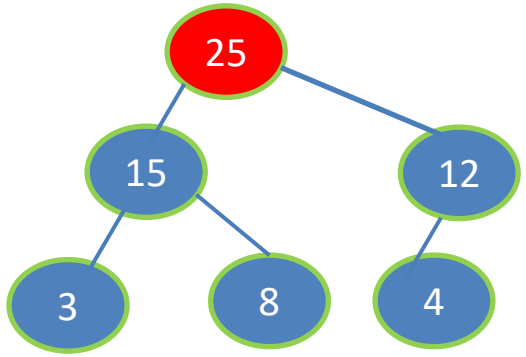
a) Gegeben sie die Folge der Schlüssel 12, 3, 15, 25, 8, 4, 35, 31.
Fügen Sie diese Folge in einen anfangs leeren Max-Heap ein. Notieren Sie dabei die Zwischenschritte.

b) Wie hoch ist der Aufwand um einen Max-Heap aus N Schlüsseln aufzubauen und anschließend N/2 Schlüssel zu entfernen im schlechtesten Fall (O-Notation)





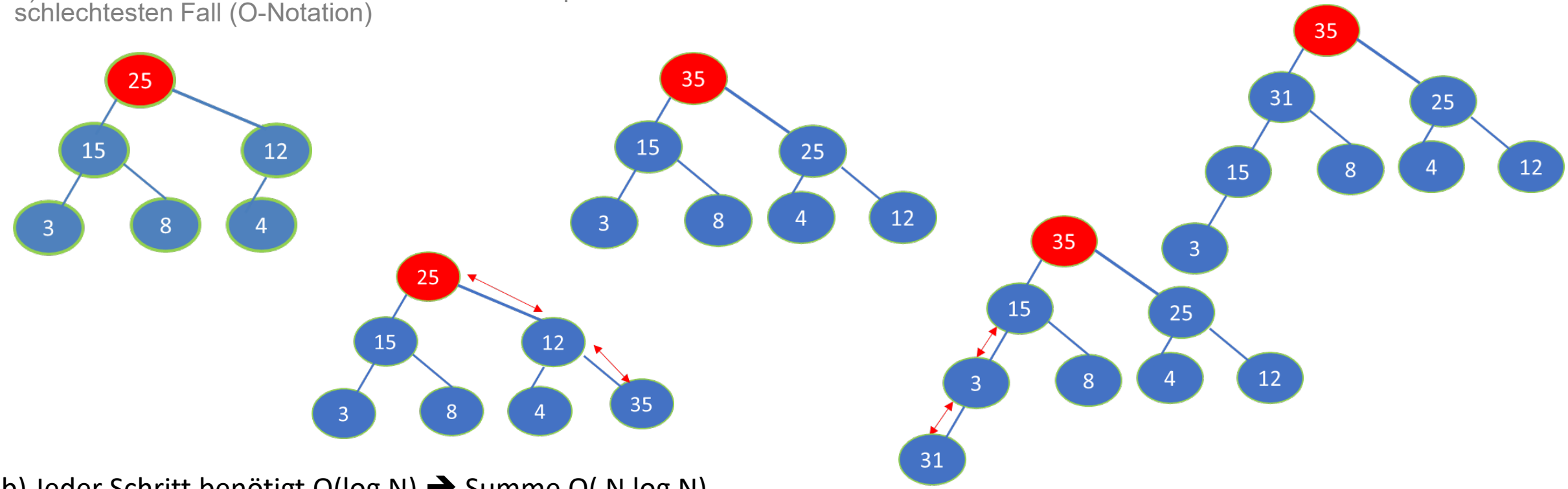




Übungsblatt 9 –Aufgabe 4

a) Gegeben sie die Folge der Schlüssel 12, 3, 15, 25, 8, 4, 35, 31.
Fügen Sie diese Folge in einen anfangs leeren Max-Heap ein. Notieren Sie dabei die Zwischenschritte.

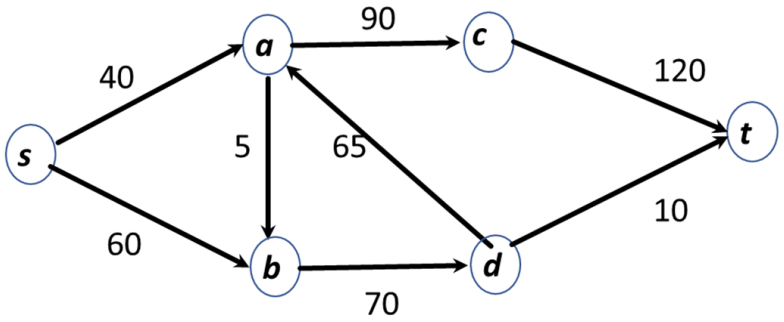
b) Wie hoch ist der Aufwand um einen Max-Heap aus N Schlüsseln aufzubauen und anschließend N/2 Schlüssel zu entfernen im schlechtesten Fall (O-Notation)

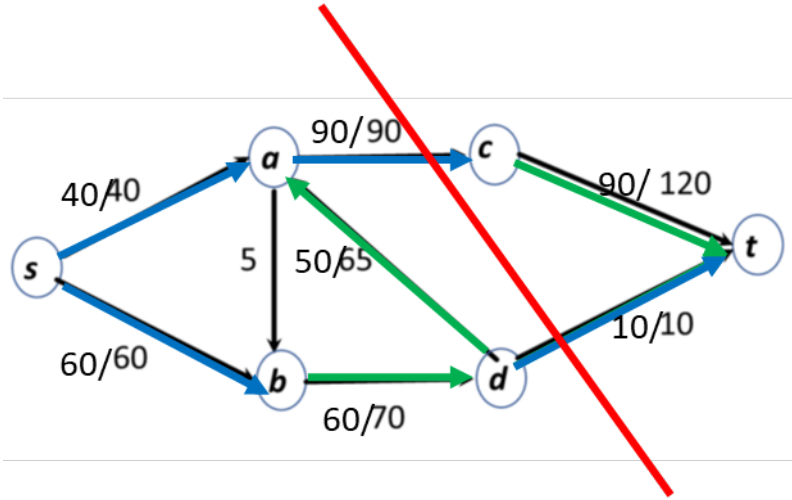


b) Jeder Schritt benötigt O(log N) ➔ Summe O(N log N)

Übungsblatt 9 –Aufgabe 5

Gegeben folgendes Flussnetzwerk mit Quelle s und Senke t





a) Bestimmen Sie den maximalen Fluss zu diesem Flussnetzwerk durch Erweiterungspfade. Aktualisieren Sie das Flussnetzwerk und den Fluss nach jeder Erhöhung durch einen Erweiterungspfad.

b) Geben Sie einen Minimalen Schnitt an.

1. s-b-d-t E=10 F=10
2. s-a-c-t E= 40 F =50
3. s-b-d-a-c-t E=50 F=100

S={s,a,b,d}T={c,t}

Übungsblatt 9 –Aufgabe 6

a) Geben Sie die Belegung einer Hashtabelle der Größe 11 an, wenn die Schlüssel
13 4 28 21 35 24 15 56
in die anfangs leere Tabelle eingefügt werden und offenes Hashing mit Hashfunktion $h(k) = k \bmod 11$ verwendet wird mit Linearem Sondieren (nach links).

0	1	2	3	4	5	6	7	8	9	10
24	35	13	15	4		28			56	21

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

4 Elemente an Original-Adresse: je 1 Zugriff;
2x 2Zugriffe (35,15); 1x 3 Zugriffe (24); 1x 4 Zugriffe (56)

Summe: 15 Zugriffe für 8 Schlüssel ➔ Durchschnittliche Kosten 15/8 (=1,875)

c) Nennen Sie zwei Probleme für offene Hashverfahren.

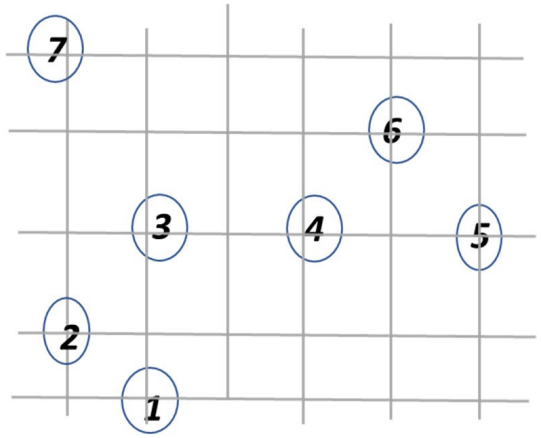
Primäre Cluster → Kollisionen; bei hohem Fullgrad explodieren Vergleiche für Suche
Beim Löschen müssen evtl. Elemente mit anderem Hash-key verschoben werden

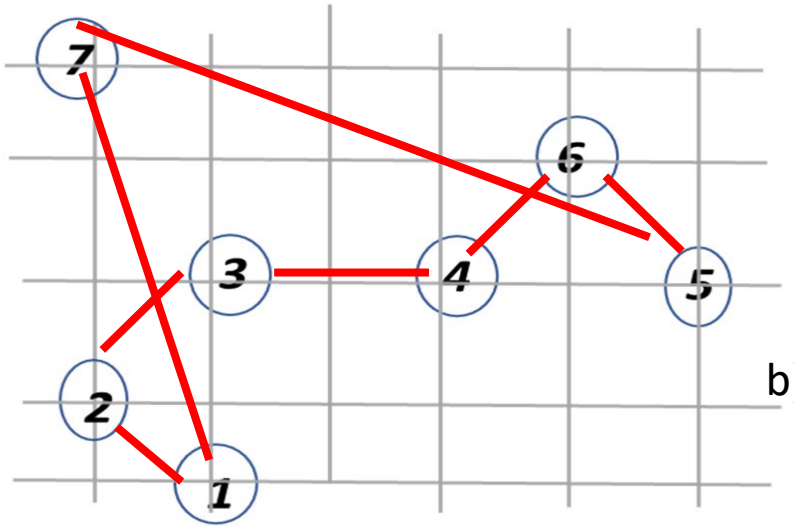
Übungsblatt 9 –Aufgabe 7

Gegeben sei der Graph G mit der Knotenmenge als Punkte im 2-dimensionalen Raum.
Die Kantengewichte sind gegeben durch den euklidischen Abstand zwischen den Knoten.

a) Bestimmen Sie ausgehend vom Startknoten 1 mit der Heuristik *Nearest Neighbor* (aus der Vorlesung) eine zulässige Lösung für das TSP-Problem.

b) Was kann man i.a. von der Güte der mit dieser Heuristik gefundenen Lösungen aussagen?





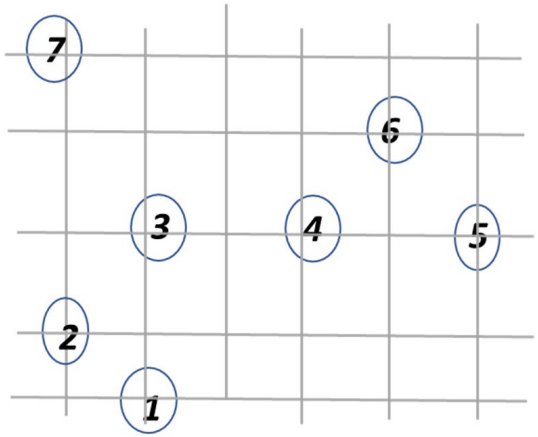
b) Heuristik kann beliebig schlecht sein, d.h. keine endliche Güte

Übungsblatt 9 –Aufgabe 7

Gegeben sei der Graph G mit der Knotenmenge als Punkte im 2-dimensionalen Raum.
Die Kantengewichte sind gegeben durch den euklidischen Abstand zwischen den Knoten.

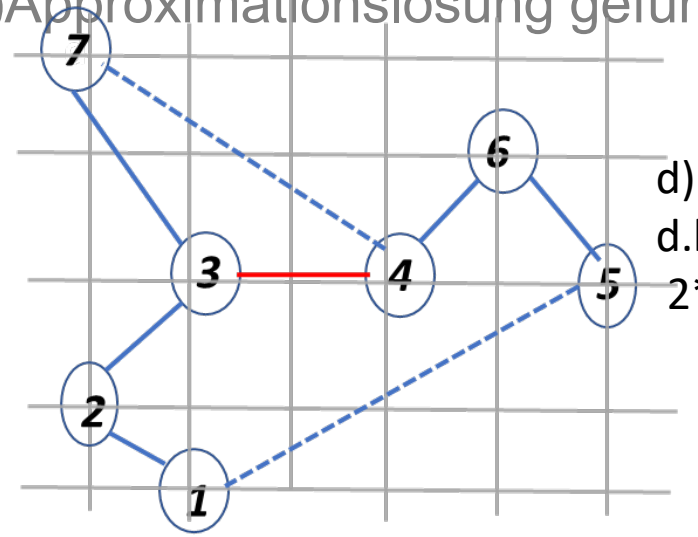
c) Ermitteln Sie ausgehend vom Startknoten 1 einen Minimal Spannenden Baum und hieraus eine (MST-)Approximationslösung.

d) Was kann man i.a. zur Güte der mit dieser (MST-)Approximationslösung gefundenen Lösungen aussagen?



MST: 1-2-3(-7)-4-6-5
MST preorder
1-2-3-7-3-4-6-5-6-4-3-2-1

Durch Überspringen der grünen schon besuchten Knoten
1-2-3-7-4-6-5-1



d) Güte 2
d.h. MST-App ≤ 2* TSP