Datenstrukturen und Algorithmen: Hausübung 5

Felix Schrader, 3053850
Jens Duffert, 2843110
Eduard Sauter, 3053470

19. November 2015

Aufgabe 1

a)

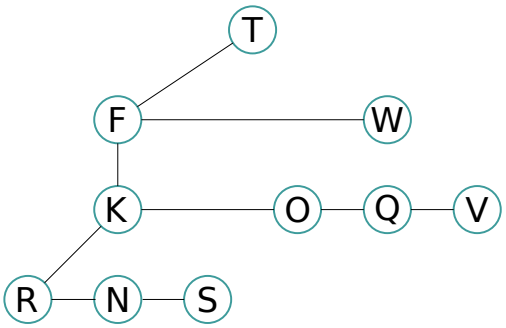

Abbildung 1: Binärgraphenkonstruktion nach Vorlesung

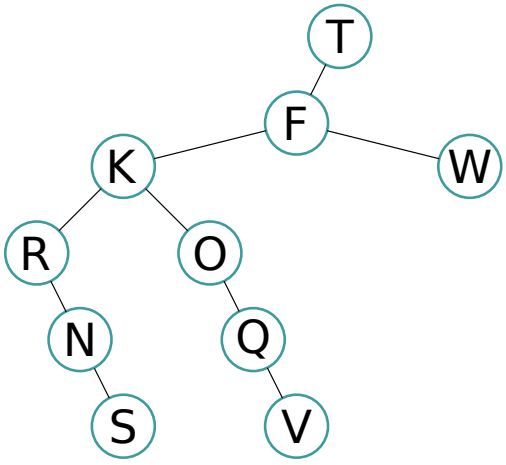

Abbildung 2: Binärgraph in kanonischer Darstellung

b) **Einfacher Baum**

Stufenordnung TFWKOQVRNS
Präordnung TFKRNSWOQV

Postordnung RNSKFOQVWT

Symmetrische Ordnung *Ist nur für binäre Bäume definiert*

Binärer Baum

Stufenordnung TFKWRONQSV

Präordnung TFKRNSOQVW

Postordnung SNRVQOKWFT

Symmetrische Ordnung RNSKOQVFWT

Aufgabe 3

a.) Die Idee ist, das für den iterativen Algorithmus ein ADT-Stack erstellt wird zuerst. In diesem werden immer die Daten von den einzelnen Punkten gespeichert. Nun werden die linken Punkte in diesem Stack schrittweise hinzugefügt. Wenn nun das Ende des Stranges erreicht ist, wird das erste Element aus dem Stack ausgegeben und gelöscht.

```
1  InorderTreeWalk(ADTTree T){
2      S = new ADTStack;
3      root = T.root();
4      node = root;
5      while(true) {
6          if(node != NIL) {
7              S.push(node);
8              node = T.left(node);
9          }
10         else {
11             if(S.empty()) {
12                 break;
13             }
14             node = S.pop();
15             print(T.retrieve(node));
16             node = T.right(node);
17         }
18     }
19 }
```

b.) Die Idee ist, das hierfür ein ADT-Queue als Speicher benutzt wird. Dieser wird zuerst erstellt. Anschließend wird die Wurzel (das erste Element) gespeichert. Anschließen wird dies ausgegeben. Nun werden die Elemente links und rechts von dem ausgegebenen Element eingelesen. Dieser Vorgang wiederholt sich.

```
1  function LevelOrderTreeWalk(ADT Tree T){}
2      Q = new Queue;
3      root = T.root();
4      Q.enqueue(root);
5      LevelOrderTreeWalkRecurse(root, Q);
6  }
7
8  function LevelOrderTreeWalkRecurse(ADTTree T, ADTQueue Q){
9      node = Q.dequeue();
10     left = T.left(node);
11     right = T.right(node);
12
13     print(T.retrieve(node))
14
```

```
15      if (left != NIL) {
16          Q.enqueue(left)
17          LevelOrderTreeWalkRecursive(T, Q);
18      }
19      if (right != NIL) {
20          Q.enqueue(right)
21          LevelOrderTreeWalkRecursive(T, Q);
22      }
23  }
```