TECHNISCHE UNIVERSITÄT BRAUNSCHWEIG

INSTITUT FÜR
BETRIEBSSYSTEME UND RECHNERVERBUND

Prof. Dr. H. Langendörfer · Prof. Dr. M. Zitterbart

Übungen zu Algorithmen und Datenstrukturen II

Blatt 7

17. Aufgabe: Dynamisches Hashen

(a) In eine dynamische Hashtabelle sollen die Hashwerte {48, 251, 20, 118, 128, 16, 22, 132, 169, 69, 190} eingetragen werden. In jeder Seite ist Platz für maximal 4 Einträge. Geben Sie schrittweise den Aufbau der Hashtabelle an.

(b) Falls alle Hashwerte einer Seite im k−ten Bit gleich sind, kann man die Seite auf der Ebene k nicht aufspalten. Was kann man in einem solchen Fall tun?

18. Aufgabe: Adjazenzlisten

Geben Sie eine Methode zum Löschen von Knoten in Adjazenzlisten an. Hierbei ist darauf zu achten, daß auch alle Kanten gelöscht werden, die von/zu diesem Knoten führen. Gehen Sie von den in der Vorlesung verwendeten Basisklassen für Knoten und Kanten aus:

```
class Node
{
  Node   next;
  Edge   edge;
  Object data;
}
```

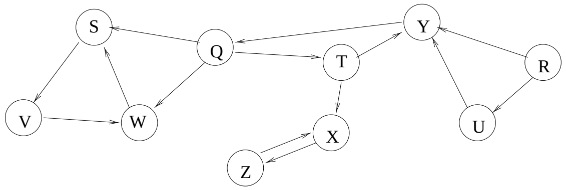
```
class Edge
{
  Edge   next;
  Node   node;
  Object data;
}
```

(a) Geben Sie eine Implementierung in Java an, die in einem gegebenen Graphen einen vorhandenen Knoten löscht. Der Graph wird durch eine Referenz auf den Startknoten in der Adjazenzliste spezifiziert. Die Methode soll dabei folgende Syntax haben:
`Node delNode ( Node G, int nr )` .
`G` ist die Referenz auf der Startknoten in der Adjazenzliste und `nr` ist die Nummer des zu löschenden Knotens. Die Methode liefert eine neue Referenz auf den Graphen zurück.

(b) Welche Sonderfälle können auftreten ?

(c) Was ist die Laufzeit des Algorithmus ?

19. Aufgabe: Tiefen- und Breitensuche



(a) Durchsuchen Sie den obigen Graphen mittels der Tiefensuche. Gehen Sie dabei davon aus, daß die Nachbarknoten in alphabetischer Reihenfolge gespeichert sind. In welcher Reihenfolge wird der Graph durchsucht, wenn von Knoten **Q** und von Knoten **R** mit der Suche begonnen wird.

(b) Durchsuchen Sie den obigen Graphen mittels der Breitensuche. Beginnen Sie bei Knoten **R**.

(c) Ein gerichteter Graph $G = (V, E)$ ist *einfach zusammenhängend*, wenn für jedes Knotenpaar $(v, v') \in V$ einen Weg von v nach v' existiert. Geben Sie einen effizienten Algorithmus in Java-Pseudo-Code an, der entscheidet, ob ein Graph einfach zusammenhängend ist.