Übung: Algorithmen und Datenstrukturen
SS 2007

Prof. Lengauer
Sven Apel, Michael Claßen, Christoph Zengler, Christof König

Blatt 10

– Votierung in der Woche vom 09.07.07–13.07.07 –

Aufgabe 29 Hashing I

Es sei folgende Hashfunktion gegeben:

$$h(k) = (g(\text{»1. Buchstabe von } k\text{«}) + g(\text{»2. Buchstabe von } k\text{«})) \bmod 17$$

wobei g eine Funktion ist, welche einen Buchstaben auf die zugehörige Position (von 1 bis 26) im Alphabet abbildet. Die Hashfunktion h bildet also auf den Adressraum 0 bis 16 (mittels der Operation: mod 17) ab.

(a) Berechnen Sie die Werte der Hashfunktion für die 12 Monate (Januar – Dezember).

(b) Tragen Sie die Funktionswerte in eine Hashtabelle ein, wobei Sie das (unterschiedliche) Ergebnis der Überlaufmechanismen

   i. Open Hashing

   ii. Closed Hashing with Linear Probing

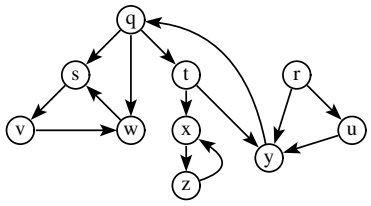
   graphisch darstellen.

Aufgabe 30 Tiefensuche auf gerichteten Graphen

Gegeben sei ein einfacher gerichteter Graph $G = (V, E)$. Eine Tiefensuche auf G partitioniert die Kantenmenge in die Baum-, Vorwärts-, Rückwärts- und Querkanten. Zusätzlich wird jedem Knoten seine Tiefensuchnummer zugewiesen. Hierzu ist die Methode `int dfs(Node<T>, int)` aus dem Skript leicht abzuändern:

```
private int dfs(Node<T> v, int dfsNum) {
    v.setInProgress(true);  // gibt an, ob der aktuelle Knoten in
                            // Bearbeitung ist, ist überall mit
                            // false initialisiert.
    for (Edge<T> e : v.getOutEdges()) {
        Node<T> w = e.getOppositeVertex(v);
        if (w.dfsNum == 0) {
            e.setMark("tree-edge");
            w.dfsNum = dfsNum++;
            dfsNum = dfs(w, dfsNum);
        } else {
            if (w.getInProgress()) {
                e.setMark("backwards-edge");
            } else {
                if (w.dfsNum > v.dfsNum) {
                    e.setMark("forward-edge");
                } else {
                    e.setMark("cross-edge");
                }
            }
        }
    }
    v.setInProgress(false); // Knoten komplett abgearbeitet
    return dfsNum;
}
```

(a) Führen Sie den DFS auf folgendem Graphen durch. Gehen Sie dabei davon aus, daß der Algorithmus bei q startet und die Knoten in alphabetischer Reihenfolge berücksichtigt, sowie die Kanten in alphabetischer Reihenfolge ihrer Zielknoten durchläuft.



Geben Sie dazu für jeden Knoten die DFS-Nummer an, sowie die vier Kantenmengen der Baum-, Vorwärts-, Rückwärts- und Querkanten.

(b) Beweisen oder widerlegen Sie folgende Aussage: Sei $G = (V, E)$ ein gerichteter Graph und $u, v \in V$ so, dass $\text{dfsNum}(u) < \text{dfsNum}(v)$ und es existiert ein Pfad von u nach v. Dann ist v ein Nachfolger von u im DFS-Baum.

Aufgabe 31 Kürzeste Wege in Graphen

Bestimmen Sie mit Hilfe des Dijkstra-Algorithmus den kürzesten Weg vom Knoten a aus zum Knoten e im angegebenen Graphen. Geben Sie dazu nach der Abarbeitung jedes Knotens die Menge der grauen Knoten und die bisher gefundenen Entfernungen an.