Algorithmen und Berechnungskomplexität I WS 15/16

Universität Bonn, Institut für Informatik, Abteilung I

9. Aufgabenblatt zur Vorlesung **Abgabe: 12.01.** (12^{30})

Aufgabe 33: Weihnachtsbäume (4 Punkte)

Ein Weihnachtsbaum ist ein binärer Baum, in welchem jeder Knoten geschmückt wurde. Schmuck ist entweder ein Weihnachtsstern oder eine Kerze. Aus Gründen der Sicherheit und Ästhetik müssen beim Schmücken folgende Bestimmungen eingehalten werden:

1. Die Spitze (Wurzel) und jedes Blatt wird mit einem Stern geschmückt;
2. Ein Knoten der mit einer Kerze geschmückt wurde hat zwei Söhne, die mit einem Stern geschmückt sind;
3. Für jeden Knoten befinden sich auf dem Pfad zu jedem Blatt in jedem seiner Teilbäume gleich viele Sterne. Wir nennen diese Anzahl *Sternenzahl*.

Beweisen Sie, dass jeder so geschmückte Weihnachtsbaum mit n inneren Knoten höchstens die Höhe $2\lg(n+1)$ besitzt. Gehen Sie dazu wie folgt vor:

- $\star$ Zeigen Sie via Induktion über die Höhe: Ein Weihnachtsbaum der Höhe h und Sternenzahl s in der Wurzel hat mindestens $2^s - 1$ innere Knoten.
- $\star\star$ Überlegen Sie sich, wie groß die Sternenzahl eines Weihnachtsbaums der Höhe h mindestens ist und folgern Sie die Behauptung.

Bitte wenden!

Aufgabe 34: Hashing **(4 Punkte)**

Erstellen Sie durch sukzessives Einfügen eine Hashtabelle mit $m = 7$ Zellen für die Schlüsselwerte

$$10, 22, 31, 4, 15$$

a) Verwenden Sie Kollisionsbehandlung mittels verketteter Listen mit der primären Hashfunktion $h(x) = x \mod 7$.

b) Verwenden Sie lineare Sondierung mit der primären Hashfunktion $h(x) = x \mod 7$.

Beim der linearen Sondierung wird im Falle einer Kollision das nächste freie Feld in der Hashtabelle benutzt. Für ein Element x und eine Hashtabelle mit m Feldern wird somit folgende Probefolge erzeugt:

$$g(x, i) = (h(x) + i) \mod m.$$

Dabei wird solange iteriert, bis ein Wert für i gefunden wurde, der zu einem noch nicht belegten Tabellenfeld führt.

Aufgabe 35: Union-Find **(4 Punkte)**

Betrachten Sie eine Union-Find-Datenstruktur, die durch verkettete Listen repräsentiert wird und zu Anfang leer ist. Wie sieht die Union-Find-Datenstruktur aus, die durch Anwendung der folgenden Folge von Anweisungen entsteht? Wir erlauben dabei, dass eine Menge nicht nur durch ihren Repräsentanten angesprochen wird, sondern alternativ durch einen explizit vergebenen Bezeichner.

1. MAKE-SET(1)
2. MAKE-SET(2)
3. MAKE-SET(3)
4. MAKE-SET(4)
5. MAKE-SET(5)
6. MAKE-SET(6)
7. A := UNION(1,2)
8. B := UNION(3,4)
9. C := UNION(B,6)
10. D := UNION(A,C)

Geben Sie auch die Zwischenergebnisse nach den Schritten 7 bis 9 an!

Bitte wenden!

Aufgabe 36: Breitensuche (4 Punkte)

Gegeben einen zusammenhängenden, ungerichteten Graphen $G = (V, E)$ mit Knotenmenge $V = \{1, \ldots, n\}$ und Kantenmenge E: beschreiben Sie einen Algorithmus, der G mit Breitensuche von Startknoten $v = 1$ aus durchsucht. Unterscheiden Sie dabei die folgenden zwei Fälle:

- Der Graph G ist in Form einer Adjazenzmatrix gegeben, also einer $n \times n$-Matrix $A[.][.]$, wobei

  $$A[i][j] = 1 \Leftrightarrow (i, j) \in V$$

  und

  $$A[i][j] = 0 \Leftrightarrow (i, j) \notin V$$

  gilt.

- G ist in Form von Adjazenzlisten gegeben, d.h. für jeden Knoten $i \in V$ gibt es eine Liste L_i, die für jeden in G zu Knoten i benachbarten Knoten j einen Container mit Inhalt j enthält.

Geben Sie in beiden Fällen die Laufzeit Ihres Algorithmus in Θ-Notation an und begründen Sie Ihre Einschätzung.