Vorlesungsprüfung

703010 VO Algorithmen und Daten Strukturen 2019

23. September 2019

Vorname: _____ Nachname: _____

Matr. Nr.: _____ Note: _____

Diese Klausur besteht aus **??** Aufgaben und Sie können maximal 40 Punkte erreichen. Es sind keine elektrotechnischen Geräte oder andere Hilfsmittel zulässig. Sie haben 90 Minuten Zeit und müssen *alle* Zettel nach Abschluss der Klausur abgeben. Wenn nötig, benennen Sie Ihre Annahmen. Geben Sie präzise und knappe Antworten.

This exam consists of **??** questions with a total of 40 points. The use of electronic devices is not allowed. The exam is closed book and closed notes. The total duration of this exam is 90 minutes. Participants have to return this copy and *all* additional sheets at the end of the exam. Specify your own assumptions if needed, and provide precise and concise answers.

1 Big-O (5 Punkte)

Analysieren Sie die Laufzeit des folgenden Code-Ausschnitts und geben Sie diese in der Big-O-Notation an. Begründen Sie, wie sich die Laufzeit zusammensetzt, indem Sie die Laufzeit der einzelnen Code-Zeilen angeben.

Analyze the running-time complexity of the following code snippet and specify it in Big-O-notation. Explain how the running time is composed by specifying the running-time complexity of each line of code.

```
public static int[][] multiply(int[][] a, int[][] b) {
  int rowsInA = a.length;
  int columnsInA = a[0].length;
  int columnsInB = b[0].length;
  int[][] c = new int[rowsInA][columnsInB];
  for (int i = 0; i < rowsInA; i++)
    for (int j = 0; j < columnsInB; j++)
      for (int k = 0; k < columnsInA; k++)
        c[i][j] = c[i][j] + a[i][k] * b[k][j];
  return c;
}
```

2 Hash-Tabelle (10 Punkte)

Befüllen Sie eine Hash-Table unter Verwendung der Hash-Funktion:
Given the following hash function:

$$h(i) = (3i + 1) \bmod 13$$

mit folgenden Werten:
Construct a hash table and fill it with the following values:

$$24, 21, 8, 18, 10, 17, 1, 13, 14, 11, 2, 20$$

1. [**5 Punkte**] Lösen Sie Kollisionen durch quadratisches Sondieren auf und beschreiben Sie die Zwischenschritte.

   Resolve collisions by quadratic probing and show the intermediate steps.

2. [**5 Punkte**] Lösen Sie Kollisionen durch externe Verkettung auf und beschreiben Sie die Zwischenschritte.

   Resolve collisions by separate chaining and show the intermediate steps.

3 AVL-Tree (10 Punkte)

1. **[2 Punkte]** Welche Eigenschaften hat der AVL-Baum? Which properties does an AVL-Tree have?

2. **[8 Punkte]** Bauen Sie einen AVL-Baum auf indem sie folgende Werte, beginnend mit 33, einfügen: [33,10,15,9,3,45,4,36,12,5] Jede durchgeführte Operation ist zu beschreiben.

   Build an AVL-Tree by inserting the following numbers starting with 33: [33,10,15,9,3,45,4,36,12,5] Explain every step done when inserting a new node.

4 Huffman coding und Graphen (10 Punkte)

1. **[2 + 2 = 4 Punkte]**

   a) Angenommen, ein Textabschnitt besteht aus den folgenden Satz mit den jeweiligen Häufigkeiten, basierend auf den ersten 8 Fibonacci-Zahlen: a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21. Was ist ein optimaler Huffman-Code für die oben genannten Häufigkeiten? Zeichne den zugehörigen Huffman-Baum.

   b) Generalisieren Sie Ihre Lösung, um den optimalen Code zu finden, wenn die Häufigkeiten der n Zeichen die ersten n Fibonacci-Zahlen sind!

   a) Assume that the unique characters from a piece of text occur with the following frequencies based on the first 8 Fibonacci numbers: a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21. What is an optimal Huffman code for the above frequencies? Draw the corresponding Huffman tree.

   b) Generalize your answer to find the optimal code when the frequencies of the n characters are the first n Fibonacci numbers!

2. **[6 Punkte]**
   Ein *bipartiter* Graph ist ein Graph, dessen Knoten in zwei disjunkte Mengen U und V unterteilt werden können, so dass jede Kante im Graphen einen Knoten in U mit einem Knoten in V verbindet (z.B. existieren keine Kanten zwischen Knoten in derselben Menge). Ein bipartiter Graph ist unten dargestellt.

   Modifizieren Sie den Algorithmus der Tiefensuche (*DFS*), so dass er die Knoten eines bipartiten Graphen jeweils einer von zwei disjunkten Mengen U und V entsprechend der Definition zuweist.

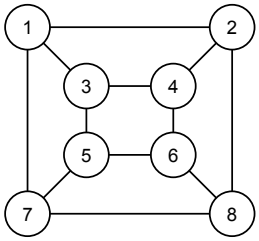
   Beschreiben Sie den Ablauf Ihres Algorithmus anhand des gegebenen Beispielgraphen, wobei im Zweifelsfall Knoten mit kleinerer Nummer vor Knoten mit größerer Nummer besucht werden. Zeigen Sie Schritt für Schritt, wie jeder Knoten entweder U oder V zugewiesen wird, und wie sich der Inhalt des von DFS verwendeten Stapels ändert.

   A *bipartite* graph is a graph whose vertices can be divided into two disjoint sets U and V such that every edge in the graph connects a vertex in U to a vertex in V (that is there are no edges between vertices in the same set). A bipartite graph is shown below.

   Modify the DFS algorithm to assign each vertex of a bipartite graph to either of two disjoint sets U and V according to the definition.

   Describe a run of your algorithm on the example graph below, where choices are resolved by visiting lower-numbered vertices before higher-numbered vertices.

   Describe step by step how each vertex is assigned to either U or V, and how the stack used by DFS is modified at each step.



5 Question Topic (5 Punkte)

Die Bergrettung bittet um Ihre Hilfe. Ein verletzter Alpinist ist zu retten, aber die Situation ist extrem kompliziert. Er ist nicht direkt per Hubschrauber erreichbar. Er kann über ein komplexes Wegenetz erreicht werden, das aus J Gabelungen besteht, von denen $H < J$ zum Herunterlassen oder Hochwinden per Hubschrauber geeignet sind. Falls eine Einmündung j von einer anderen Einmündung i durch einen direkten, gabelungsfreien Routenabschnitt erreichbar ist, beinhaltet seine Traversierung ein Risiko r_{ij} (für ein Team beliebiger Größe); das Risiko r_{ji} der Gegenrichtung kann größer oder kleiner sein.

1. Beschreiben Sie einen effizienten Algorithmus, der bestimmt, wo das Rettungsteam abgesetzt werden sollte, und wohin es den verletzten Alpinisten bringen sollte, damit er vom Hubschrauber hochgewunden werden kann, und zwar bei minimalem Risiko für den gesamten Rettungseinsatz.

   Beziehen Sie sich auf abstrakte Datentypen, Datenstrukturen und/oder Algorithmen, die im Kurs besprochen wurden. Verwenden Sie präzise, deutsche Sprache sowie ggf. Pseudocode. Schreiben Sie keinen C, Java oder ähnlichen Code.

2. Was ist die Laufzeitkomplexität Ihres Algorithmus als Funktion von J, H und der Anzahl S der gabelungsfreien Routenabschnitte?

The mountain rescue service asks you for help. An injured climber must be rescued, but the situation is extremely complicated. He is not directly reachable by helicopter. The climber can be reached via a complex network of routes consisting of J junctions, $H < J$ of which are suitable for drop-off and pick-up by helicopter. If a junction j can be reached from another junction i via a junction-free route segment, traversing it (by a team of any size) is associated with a risk r_{ij}, which may well differ from r_{ji}.

1. Devise an efficient algorithm that determines where the rescue team should be dropped off, and where they should take the injured climber for pick-up, minimizing the total risk of the rescue operation.

   Express your algorithm in terms of abstract data types, data structures and/or algorithms discussed in class. Use precise English language, and pseudocode where appropriate. Do not write C, Java or similar code.

2. What is the running-time complexity of your algorithm in terms of J, H, and the number S of junction-free route segments?