Technische Universität München
Institut für Informatik I-16
Dr. Tobias Lasser
Jakob Vogel

WiSe 2012/13
Übungsblatt 10
9. Januar 2013

Algorithmen und Datenstrukturen

Aufgabe 1 (P) Heap-Sort

Man kann den Heap, wie zuvor in Vorlesung und Übung gezeigt, auf zweierlei Arten einsetzen, ein Array zu sortieren. Die bessere Methode, weil *in-place*, konvertiert das unsortierte Array in einen Heap, tauscht dann wiederholt das aktuelle Minimum ans Ende und ignoriert diesen letzten Eintrag in Folge, bis der Heap „abgebaut“ wurde. Im Falle eines *Min-Heap* erzeugt dies eine absteigend sortierte Folge.

Implementieren Sie den Heap-Sort!

Aufgabe 2 KMP-Präfix-Tabelle

Ein wichtiger Teil der *Knuth-Morris-Pratt*-Stringsuche ist die Präfix-Tabelle, über die die Durchsuchung der Zeichenkette beschleunigt werden kann. Im Wesentlichen enthält sie pro Zeichen des Suchwortes, wie viele vorhergehende Zeichen ein Präfix des Suchwortes selbst bilden.

Bestimmen Sie die Präfix-Tabellen der folgenden Suchworte:

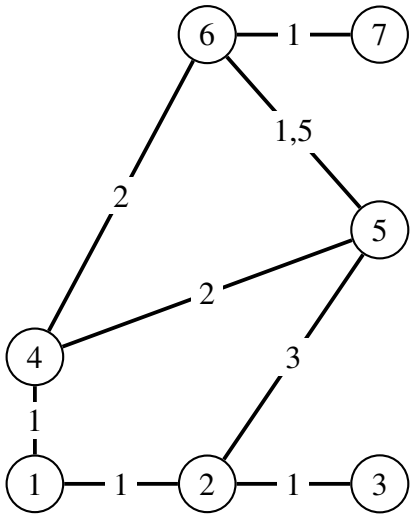
a) `std::string`

b) `int sort(int* data, int n)`

c) `TU München`

Aufgabe 3 Dijkstra

Gegeben sei der nachfolgende gewichtete Graph, in dem mittels des Dijkstra-Algorithmus der kürzeste Pfad vom Knoten 1 zum Knoten 7 gesucht werden soll.



a) Führen Sie den Dijkstra-Algorithmus durch! Geben Sie in jedem Schritt die Distanzen und Vorgänger zu jedem Knoten an.

b) Wäre der Endknoten 6 statt 7, wie würde sich der Ablauf des Algorithmus unterscheiden? Was schließen Sie hinsichtlich der *SSSP*- und *SPSP*-Probleme?

Das *SSSP*-Problem (*single source shortest path*) beschreibt die Aufstellung der Menge der kürzesten Pfade zu allen Knoten von einem fest definierten Startknoten aus. Das *SPSP*-Problem (*single pair shortest path*) ist dagegen die Suche nach dem kürzesten Pfad zwischen zwei wohldefinierten Knoten.

Aufgabe 4 (P) Dijkstra und A*

a) Nun soll der Dijkstra-Algorithmus implementiert werden. Dazu muss zunächst die *Min-Priority-Queue* erweitert werden um eine Funktion *decreaseKey*, um einem bereits zuvor erfassten Objekt einen kleineren Wert zuweisen zu können. Implementieren Sie diese Methode!

b) Nutzen Sie diese Warteschlange nun, um den Dijkstra-Algorithmus selbst zu implementieren. Vergegenwärtigen Sie sich den Verlauf der Knoten-Besuche!

c) Um die Suche zu beschleunigen, kann man zielgerichteter in Richtung des Zielknotens suchen. Die Idee ist dabei, die Kosten des bisherigen Pfades zu ergänzen um eine Heuristik $h[v]$, also eine Abschätzung, wie weit das Ziel vom Knoten v entfernt ist.

Im Fall einer Suche auf einer Landkarte könnte man etwa die direkte Distanz (die Luftlinie) zum Ziel benutzen, und dann den Pfad optimieren, der in der Summe aus Pfadlänge und Abschätzung optimal ist.

Diese Variante nennt man A*-Suche. An welcher Stelle müsste man die Heuristik einfügen?