Institut für Theoretische Informatik

Leibniz Universität Hannover

Vorlesung: PD Dr. Arne Meier
Übung: Nicolas Fröhlich

Bearbeitungszeitraum 23.11.23 - 07.12.23

Datenstrukturen und Algorithmen
Hausübung 2 (DJP, Graphen und Sortieren)

Relevant aus dem Skript: Kapitel 5-9

Für alle Lösungen sind Begründungen anzugeben!

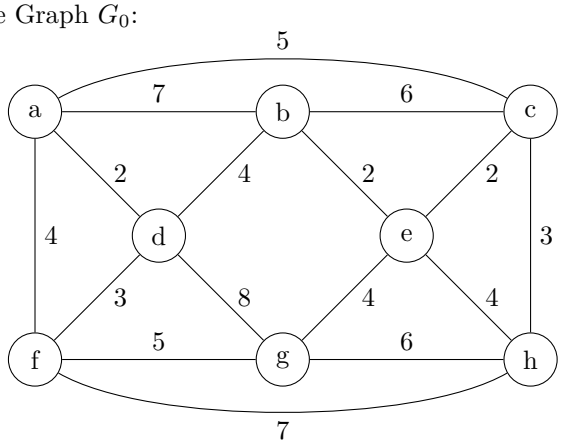
Organisatorisches und Hinweise zur Bearbeitung:

1. Bearbeiten Sie dieses Übungsblatt in Gruppen von 2-4 Leuten. (Das bedeutet insbesondere, dass Sie **keine Einzelabgaben** abgeben sollen!)
2. Schreiben Sie die Namen und Matrikelnummern **aller Gruppenmitglieder** sowohl in die Python-Datei als auch auf die PDF.
3. Verwenden Sie für die Programmieraufgaben die Datei *hausuebung02.py* und verändern Sie die bereits implementierten Datentypen und Funktionen nicht signifikant.
4. Erfolgreiche Testfälle garantieren **nicht**, dass ein Algorithmus korrekt implementiert wurde. Die Testfälle dienen nur dazu, grobe Fehler und falsche Ausgaben zu erkennen.
5. Verwenden Sie keine Datenstrukturen oder Funktionen von Python außer: `list`, `dict`, `deque`, `PriorityQueue`, `enumerate`, `range` und `len`.
6. Geben Sie Ihre Lösung **bis spätestens** am 07.12.2023 23:59 im ILIAS Kurs unter folgendem Link ab: `https://ilias.uni-hannover.de/goto.php?target=crs_175670_rcodeYPKATwaXYQ`
7. Wenn Sie Ihre Lösung in LaTeX erstellt haben und als PDF- und Python-Datei abgeben, bekommen Sie einen Extrapunkt.

Hinweis zum Einsatz von KI: Für die Bearbeitung der Aufgaben ist der Einsatz von KI erlaubt, sofern er vollständig dokumentiert und der Abgabe beigefügt wird (Chat-GPT hat z.B. eine Share-Funktion. Alternativ auch als extra PDF).

Aufgabe 1 (5 + 5 Punkte)

Gegeben sei der folgende Graph G_0:



a) Bestimmen Sie mit Hilfe des DJP-Algorithmus den minimalen Spannbaum von G_0 und geben Sie diesen an. Nutzen Sie dabei die in der Vorlesung verwendete Tabellendarstellung, um den Ablauf des Algorithmus darzustellen.

b) Implementieren Sie DJP in Python. Der Algorithmus bekommt als Eingabe einen gewichteten, ungerichteten, zusammenhängenden Graphen und soll den minimalen Spannbaum ausgeben.

Erklären Sie in eigenen Worten, wie Ihre Implementierung funktioniert.

Aufgabe 2 (2 + 6 + 2 Punkte)

Ein **Eulerkreis** in einem Graphen $G = (V, E)$ ist ein Pfad $v_1, \ldots, v_m$ mit $v_1 = v_m$ und $(v_i, v_{i+1}) \in E$ für $1 \leq i < m$, der jede Kante von G *genau einmal* besucht.

Eine **kürzeste Rundreise** in einem gewichteten Graphen $G = (V, E)$ mit Gewichtsfunktion w ist ein Pfad $v_1, \ldots, v_m$ mit $v_1 = v_m$ und $(v_i, v_{i+1}) \in E$ für $1 \leq i < m$, der jeden Knoten *genau einmal* besucht und dessen Gewicht *minimal* ist, d.h. es gibt keine andere Rundreise mit kleinerem Gewicht.

a) Bestimmen Sie einen Eulerkreis und die kürzeste Rundreise im Graphen G_0.

b) Implementieren Sie einen *beliebigen* Polynomialzeit-Algorithmus zur Berechnung von Eulerkreisen in gerichteten Graphen. Beschreiben Sie wie ihr Algorithmus arbeitet und erläutern Sie die Korrektheit und die Laufzeit!

c) Verwenden Sie den DJP-Algorithmus und Ihren Algorithmus für Eulerkreise um Rundreisen in *metrischen* Graphen in Polynomialzeit zu berechnen. Die berechnete Rundreise muss *nicht* die kürzeste Rundreise im Graphen sein! Erklären Sie Ihren Algorithmus!

Ein Graph heißt **metrisch**, wenn er vollständig ist und $w(a, c) \leq w(a, b) + w(b, c)$ für alle Knoten a, b, c gilt. Das bedeutet also, dass das Gewicht eines Umweges über Knoten b nicht kleiner als das Gewicht der direkten Verbindung von a nach c ist.

Aufgabe 3 (3 + 5 + 2 Punkte)

a) Gegeben ist folgende Liste: $[58, 3, 217, 13, 365, 524, 34, 134]$. Sortieren Sie sie mit Hilfe des Mergesort-Algorithmus. Dokumentieren Sie Ihre Zwischenschritte sinnvoll!

b) Implementieren Sie die Funktion `run_decomposition`, die eine Liste von Zahlen in Runs der Länge $\geq$ `minrun` zerlegt. Ein **Run** ist eine Teilliste, deren Elemente aufsteigend sortiert sind. Solange ein Run kleiner als `minrun` ist, werden weitere Elemente mit Hilfe von *Insertionsort* hinzugefügt. Erst danach fängt ein neuer Run an. Erklären Sie Ihre Implementierung!

c) Implementieren Sie mit Hilfe von `run_decomposition` den in der Vorlesung vorgestellten Sortieralgorithmus **Timsort**.