Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
P. Bamberger
P. Schneider

Algorithmen und Datenstrukturen
Sommersemester 2020
Übungsblatt 9

Abgabe: Mittwoch, 15.07.2020, 16:00 Uhr.

Aufgabe 1: Eindeutige Minimale Spannbäume *(10 Punkte)*

Sei $G = (V, E, w)$ ein *ungerichteter, zusammenhängender, gewichteter* Graph mit paarweise veschiedenen Kantengewichten.

(a) Zeigen Sie, dass G einen *eindeutigen* minimalen Spannbaum hat. *(5 Punkte)*

(b) Zeigen Sie, dass man durch die folgende Konstruktion den minimalen Spannbaum T' von G erhält:

Starte mit $T' = \emptyset$. Für jeden Schnitt in G, füge die leichteste Schnittkante zu T' hinzu.

(5 Punkte)

Aufgabe 2: Problem des Handlungsreisenden *(10 + 5* Punkte)*

Seien $p_1, \ldots, p_n \in \mathbb{R}^2$ Punkte in der euklidischen Ebene. Der Punkt p_i gibt die Position von Stadt i an. Die Distanz zwischen zwei Städten i und j entspricht der euklidischen Distanz der entsprechenden Punkte p_i, p_j. Eine *Rundreise* ist eine Abfolge von Städten $(i_1, \ldots, i_n)$ die jede Stadt genau ein mal besucht (formal: eine Permutation der Menge $\{1, \ldots, n\}$). Gesucht ist die Rundreise welche die zurückgelegte Distanz minimiert. Dieses Problem ist im Allgemeinen sehr schwer.[1] Wir geben uns deshalb mit einer Rundreise zufrieden die höchstens doppelt so lang ist wie die minimale Rundreise. Wir können das auch als Graphenproblem mit $G = (V, E, w)$ auffassen, wobei $V = \{p_1, \ldots, p_n\}$ und $w(p_i, p_j) := \|p_i - p_j\|_2$. Damit ist G *ungerichtet und vollständig* und es gilt die *Dreiecksungleichung*.[2] Gesucht ist eine Rundreise $(i_1, \ldots, i_n)$ mit kleiner Gewichtssumme $w(p_{i_n}, p_{i_1}) + \sum_{j=1}^{n-1} w(p_{i_j}, p_{i_{j+1}})$.

(a) Sei G ein Graph wie oben beschrieben. Zeigen Sie, dass die Knoten eines minimalen Spannbaumes in Pre-order Reihenfolge (ausgehend von einer beliebigen Wurzel) eine Rundreise in G ergibt die höchstens doppelt so lang ist wie die minimale Rundreise. *(5 Bonus Punkte)*

(b) Implementieren Sie einen Algorithmus der eine Pre-order Reihenfolge eines minimalen Spannbaumes von G berechnet. Sie dürfen dazu die Vorlagen `TSP.py` und `AdjacencyMatrix.py` sowie Module für Heap und Union-Find Datenstrukturen benutzen[3]. Lesen Sie den Beispielgraph aus `cities.txt` als Adjazenzmatrix ein und wenden Sie ihren Algorithmus darauf an. Berechnen und notieren Sie die Gewichtssumme der resultierenden Rundreise (s.o.) in Ihren `erfahrungen.txt`. *(10 Punkte)*

---

[1] Das (exakte) Problem des Handlungsreisenden ist aus der sogenannten Klasse der $\mathcal{NP}$-vollständigen Probleme, für deren Lösung wahrscheinlich kein Polynomialzeitalgorithmus existiert (nur falls $\mathcal{P} = \mathcal{NP}$ was noch niemand zeigen oder widerlegen konnte). Mehr dazu in der Vorlesung zu theoretischer Informatik.

[2] Die Dreiecksungleichung impliziert, dass die direkte Kante zwischen zwei Knoten $a, b \in V$ höchstens so lang ist wie ein Umweg über einen dritten Knoten $c \in V$, d.h. $w(a, b) \leq w(a, c) + w(c, b)$.

[3] Zum Beispiel `heapq` und `networkx.utils.union_find`. Bei `heapq` entspricht `heappush` der `insert` und `heappop` der `delete-min` Operation aus der Vorlesung. Dabei können `heappush` und `heappop` auf Python-Listen angewendet werden (mehr Details hier). Wenn Sie ein Objekt `uf` der Klasse `UnionFind` instantiiert haben, legt `uf[i]` eine neue Menge $\{i\}$ an falls i noch nicht in `uf` existiert und liefert sonst einen Repräsentanten der Menge zurück in der i enthalten ist (dies kombiniert also die Funktionen von `make-set` und `find`, mehr Details hier).