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

Algorithmen und Datenstrukturen
Sommersemester 2020
Musterlösung 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)

Musterlösung

(a) Seien T und T' zwei minimale Spannbäume mit Kanten $e_1, \dots, e_{n-1}$ und $e'_1, \dots, e'_{n-1}$, jeweils aufsteigend nach Gewicht sortiert. Nehme an es gelte $T \neq T'$. Sei j der größte Index, für den $e_j \neq e'_j$ gilt. Da die Gewichte paarweise verschieden sind, muss auch $w(e_j) \neq w(e'_j)$ gelten. O.b.d.A. sei $w(e_j) < w(e'_j)$. Der Graph $T' \setminus \{e'_j\}$ hat zwei Zusammenhangskomponenten mit Knoten S und $V \setminus S$. Sei e_k die Kante aus T, welche S und $V \setminus S$ verbindet. Da T' nur eine Kante zwischen S und $V \setminus S$ haben kann, muss $k \leq j$ sein (da $e_k = e'_k$ für $k > j$). Nun ist aber $(T' \setminus \{e'_j\}) \cup \{e_k\}$ wieder ein Spannbaum und da $w(e'_j) > w(e_j) \geq w(e_k)$ hat dieser ein kleineres Gewicht als T' im Widerspruch dazu, dass T' minimal ist.

(b) Sei T der MST von G und T' die Menge der leichtesten Schnittkanten (wegen der Eindeutigkeit der Gewichte ist T' eindeutig bestimmt).

$T' \subseteq T$: Sei $s = \{x, y\} \in T'$, d.h. s ist die leichteste Schnittkante eines Schnitts $(S, V \setminus S)$ in G. In T gibt es einen (eindeutigen) Pfad von x nach y. Seien $e_1, \dots, e_k$ diejenigen Kanten in diesem Pfad, welche zwischen S und $V \setminus S$ verlaufen ($k \geq 1$). Wäre $s \notin \{e_1, \dots, e_k\}$, so wäre $w(s) < w(e_i)$ (für ein beliebiges $i \in \{1, \dots, k\}$) und der Spannbaum $(T \setminus \{e_i\}) \cup \{s\}$ hätte kleineres Gewicht als T im Widerspruch dazu, dass T ein MST ist. Also muss $s \in \{e_1, \dots, e_k\} \subseteq T$.

$T \subseteq T'$: Sei $e \in T$. Der Graph $T \setminus \{e\}$ hat zwei Zusammenhangskomponenten, die einen Schnitt in G definieren. Mit einem Austauschargument wie oben zeigt man, dass e die (eindeutig) leichteste Kante in diesem Schnitt sein muss, d.h. es ist $e \in T'$.

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)*

Musterlösung

(a) Sei $R = (i_1, \ldots, i_n)$ eine minimale Rundreise und $w(R) := w(p_{i_n}, p_{i_1}) + \sum_{j=1}^{n-1} w(p_{i_j}, p_{i_{j+1}})$. Sei T ein minimaler Spannbaum, sei $w(T) := \sum_{e \in T} w(e)$ dessen Gewicht und $\mathcal{P}_T$ die Folge der Knoten in pre-order Reihenfolge. Die Folge $\mathcal{P}_T$ bildet ebenso eine Rundreise da der Graph vollständig ist.

Wir ergänzen die Folge $\mathcal{P}_T$ wie folgt: Sind zwei aufeinanderfolgende Knoten u und v nicht durch eine Baumkante in T verbunden, so fügen wir zwischen u und v die Knoten auf dem kürzesten Pfad von u nach v in T ein (dies sind alle Knoten von u zum ersten gemeinsamen Vorfahren w und von dort zu v). Die Folge die wir erhalten nennen wir $\mathcal{P}'_T$ (dies ist formal keine Rundreise, da Punkte mehrfach besucht werden).

In der Folge $\mathcal{P}'_T$ sind je zwei aufeinanderfolgende Knoten Nachbarn in T, d.h. man kann dies auch als Folge von Kanten in T auffassen. Jede Kante von T kommt in $\mathcal{P}'_T$ genau zwei mal vor (wenn man vom letzten Punkt wieder zurück zur Wurzel geht). Wir haben daher $w(\mathcal{P}'_T) = 2 \sum_{e \in T} w(e)$. Aus der Dreiecksungleichung folgt $w(\mathcal{P}_T) \leq w(\mathcal{P}'_T)$ und daher $w(\mathcal{P}_T) \leq 2 \sum_{e \in T} w(e)$.

Die minimale Rundreise R definiert selbst einen Spannbaum T_R indem wir die Kanten zwischen zwei aufeinanderfolgenden Knoten in $R = (i_1, \ldots, i_n)$ zu T_R hinzufügen, denn T_R besteht aus den Kanten der Rundreise exklusive die Kante (p_{i_n}, p_{i_1}) und ist dann zusammenhängend, zyklenfrei und enthält alle Knoten. Da T der minimale Spannbaum ist gilt $w(T) \leq w(T_R) \leq w(T_R) + w(p_{i_n}, p_{i_1}) = w(R)$ und deshalb $w(\mathcal{P}_T) \leq 2 \cdot w(R)$.

Bemerkung: Die obige Argumentation funktioniert grundsätzlich auch für die Post-Order Traversierung. Legt man allerdings einen Startpunkt der Tour fest, so ist es am einfachsten, eine Pre-Order Traversierung von diesem Knoten ausgehend zu machen.

---

[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).

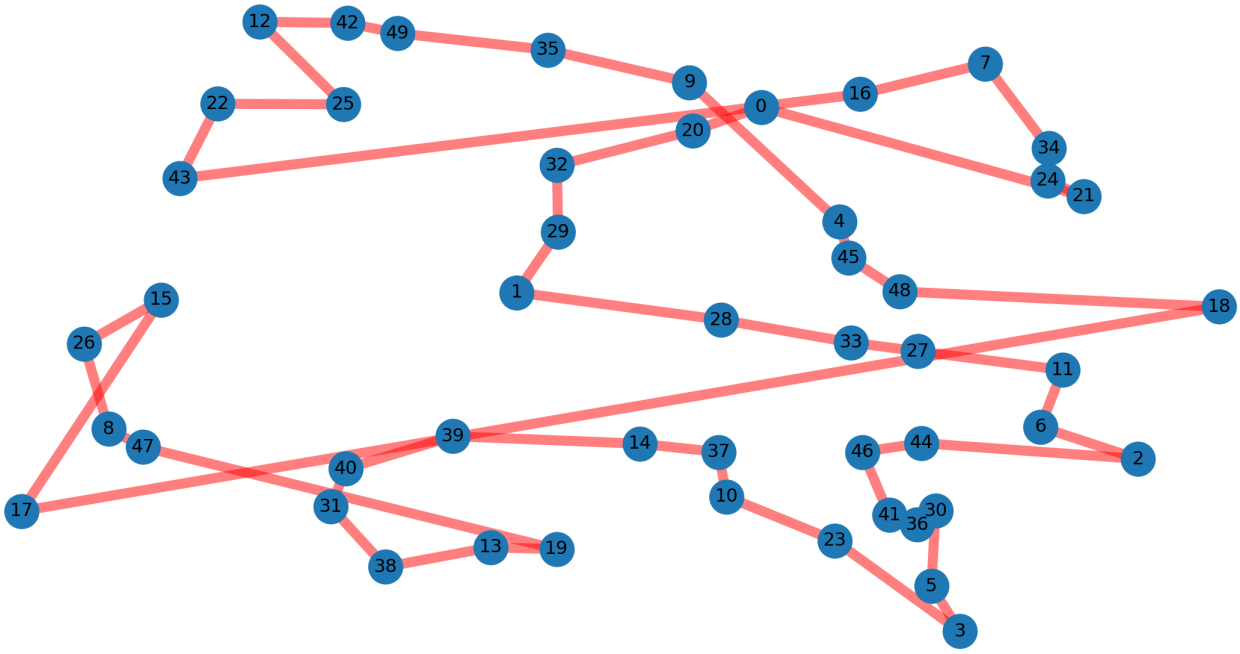

Figure 1: Abbildung der approximativen Rundreise.

(b) Siehe `TSP.py` für unsere Implementierung. Die berechnete Rundreise unserer Lösung hat eine Länge von 400,86 (die Pre-order-Reihenfolge und damit dessen Länge sind allerdings nicht eindeutig). Da der eindeutige MST ein Gewicht von 253,42 hat sollte die eigene Abschätzung mindestens 253,42 und höchstens 506,84 betragen (siehe Aufgabenteil (a)). Eine Illustration wie die Rundreise zwischen den Städten verläuft gibt Abbildung 1.