Algorithmen & Datenstrukturen 1

Sommersemester 2022

Prof. Dr. Martin Hoefer
Tim Koglin, Marco Schmalhofer

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

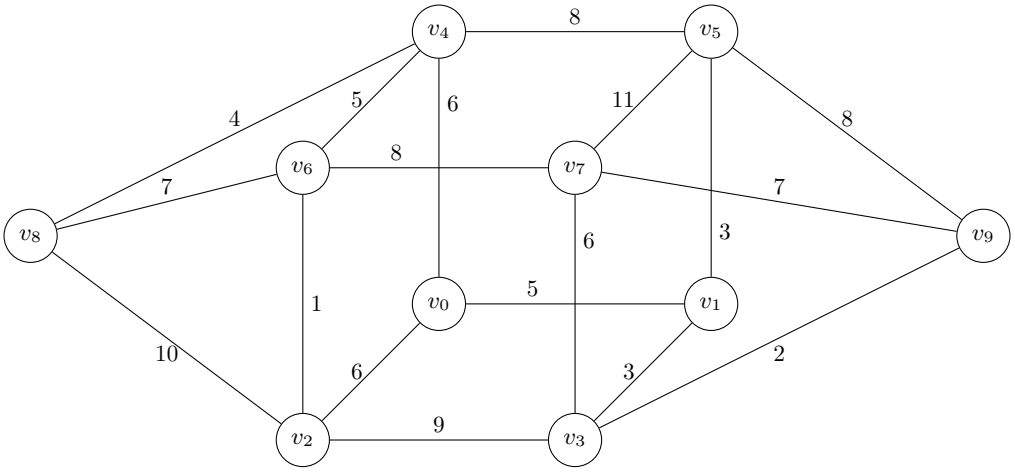
Institut für Informatik
Algorithmen und Komplexität

Übungsblatt 10

Ausgabe: 21.06.2022
Abgabe: 28.06.2022, **08:00**

Aufgabe 10.1 *Kruskal und DJP* (8 + 6 Punkte)

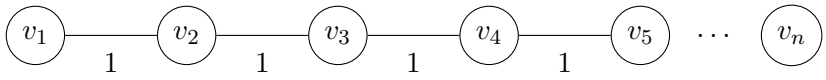
Gegeben sei der folgende ungerichtete Graph mit Kantengewichten:



a) Bestimmen Sie mit Hilfe von Kruskals Algorithmus den minimalen Spannbaum. Bei Kanten mit gleichem Gewicht erfolgt die Sortierung gemäß aufsteigender Knotenindizes (vergleiche zuerst die jeweils kleineren, bei Gleichheit dann die jeweils größeren). Geben Sie in jedem Schritt den Zustand der *Union-Find*-Datenstruktur als Elternarray an. Die Größe eines Baums in der Union-Find-Datenstruktur sei in diesem Beispiel über die Tiefe definiert. Bei gleicher Tiefe sei der Baum mit größerem Index der Wurzel der größere Baum.

b) Betrachten Sie für $n \in \mathbb{N}$ den gewichteten Graphen $G_n = (V_n, E_n)$ mit

- Knotenmenge $V_n = \{v_1, v_2, \ldots, v_n\}$,
- Kantenmenge $E_n = \{\{v_i, v_{i+1}\} : 1 \leq i < n\}$,
- Gewichten $w_n(e) = 1$ für alle $e \in E_n$.

G_n : 

Ergänzen Sie den Graphen G_n um eine Menge E'_n von gewichteten Kanten, so dass beim Aufruf des DJP-Algorithmus auf dem Graphen $G'_n = (V_n, E_n \cup E'_n)$ mit Startknoten v_1 folgendes gilt: Beim Hinzufügen eines jeden Knotens in die Menge S verringert sich der Wert aller Knoten in $V \setminus S$. Geben Sie die Menge E'_n der hinzugefügten Kanten und deren Gewichte an.

Aufgabe 10.2 *MinMax-Spannbäume* (7 Punkte)

Sei $G = (V, E)$ ein ungerichteter Graph. Entwerfen Sie einen Algorithmus, der einen Spannbaum T von G bestimmt, wobei das Gewicht der schwersten Kante in T minimiert werden soll. Die Laufzeit soll $\mathcal{O}(|V| + |E| \cdot \log |V|)$ nicht überschreiten.

Beschreiben Sie zuerst Ihre Idee und anschließend den Algorithmus (kein Pseudocode notwendig). Zeigen Sie seine Korrektheit und begründen Sie, warum die Laufzeitschranke eingehalten wird.

Hinweis: Für diese Aufgabe ist der Wert $v(T)$ eines Spannbaums $T = (V, E')$ definiert als $v(T) := \max_{e \in E'} w(e)$. Gesucht ist also ein Spannbaum T von G, so dass $v(T) \leq v(T')$ für alle Spannbäume T' von G gilt.

Aufgabe 10.3 *Alternative Union-Find Implementierungen* (4 + 4 + 3* Punkte)

Betrachten Sie die folgenden zwei verschiedenen Implementierungen einer Union-Find-Datenstruktur auf der Knotenmenge $V = \{1, 2 \ldots, n\}$:

(1) Ein Array `component[1...n]` speichert für jeden Knoten den Index eines Repräsentanten in der aktuellen Zusammenhangskomponente des Knotens. Anfangs wird `component[v]=v` für jeden Knoten $v \in V$ gesetzt. Beim Aufruf von `find(u)` wird lediglich `component[u]` zurückgegeben. `union(i,j)` ist durch folgenden Pseudocode gegeben:

```
union(i,j):
    for (k=1...n) {
        if (component[k] == component[j]) component[k]=component[i];
    }
```

(2) Die Implementierung aus der Vorlesung mit Eltern-Array. Beim Aufruf von `find(u)` wird die Wurzel des Baums von u durch Aufsteigen gesucht und zurückgegeben. Beim Aufruf von `union(i,j)` werden die Bäume vereinigt, indem die Wurzel mit *kleinerem Knotenindex* als Kind unter die andere Wurzel gehängt wird.

Nehmen Sie an, die Kantenmenge liege bereits sortiert vor. Bestimmen Sie die asymptotische worst-case Laufzeit von Kruskals Algorithmus in Abhängigkeit von der *Knotenzahl* n in $\mathcal{O}$-Notation, wenn

a) ... die Union-Find-Implementierung (1) verwendet wird.

b) ... die Union-Find-Implementierung (2) verwendet wird.

c*) Beschreiben Sie für b) einen Eingabegraphen mit n Knoten, der diese Laufzeit erzeugt.

Aufgabe 10.4 *Greedy-Matching* (5 + 3 + 3 Punkte)

Ein *Matching* auf einem ungerichteten Graphen $G = (V, E)$ ist eine Menge $M \subseteq E$ von „unabhängigen“ Kanten, d.h., für alle Kanten $e, f \in M$ gilt, dass $e \cap f = \emptyset$, wenn $e \neq f$. Ein Matching hat *maximale Kardinalität*, wenn $|M|$ größtmöglich ist.

Gegeben sei nun ein ungerichteter Baum mit n Knoten in Adjazenzlistendarstellung. Entwickeln Sie einen Greedy-Algorithmus, der in Laufzeit $\mathcal{O}(n)$ ein Matching mit maximaler Kardinalität berechnet.

a) Geben Sie Ihren Algorithmus in Pseudocode an. Beschreiben Sie zunächst Ihre Idee.

b) Zeigen Sie, dass Ihr Algorithmus die Laufzeitschranke einhält.

c) Zeigen Sie, dass Ihr Algorithmus ein korrektes Ergebnis liefert.

Bei allgemeinen Anmerkungen zu den Übungsaufgaben oder Fragen zum Übungsbetrieb erreichen Sie uns unter der folgenden E-Mail-Adresse: algo122@cs.uni-frankfurt.de.