ETH Zürich
Institute of Theoretical Computer Science
Prof. Angelika Steger, Prof. Emo Welzl
Dr. Anders Martinsson, Charlotte Knierim

FS 2022

Algorithmen und Wahrscheinlichkeit
Peer-Aufgaben 1 — Lösung

Aufgabe 1 – *Touren*

(a) Wir führen den Algorithmus von Kruskal (siehe Kapitel 1.2.2 im Skript) aus um einen minimalen Spannbaum zu finden. Wir wollen die Kanten die zum minimalen Spannbaum gehören in die Menge T einfügen. Zu Beginn sei $T = \emptyset$. Nun sortieren wir die Kanten aufsteigend nach ihrem Gewicht. In dieser Reihenfolge gehen wir durch die Kanten durch und führen wie folgt für die Kante $e = \{u, v\}$ einen von zwei möglichen Schritten aus.

- Falls es einen u-v-Pfad mit Kanten aus T gibt, verwerfen wir e.
- Ansonsten fügen wir e zu T dazu.

Nun bilden die Kanten in T einen minimalen Spannbaum. (Siehe Skript.)

Angewandt auf den gegeben Graphen G bedeutet dies, dass wir zum Beispiel die Kanten $\{A, D\}, \{A, B\}, \{C, G\}, \{B, C\}, \{E, F\}, \{F, G\}$ (in dieser Reihenfolge) zum Spannbaum hinzufügen.

Der Algorithmus von Kruskal ist sicherlich korrekt (siehe Skript). Da der Graph G konstante Grösse hat, interessiert uns hier die asymptotische Laufzeit nicht wirklich. Sogar ein Brute-Force-Algorithmus, der alle möglichen Kantenmengen der Kardinalität 6 auf Zusammenhang und Kreisfreiheit testet und schliesslich die Menge mit dem kleinsten Gewicht ausgibt, hat für diese Eingabe konstante Laufzeit.

(b) *Variante 1 (Einfacher Existenzbeweis):* Wir betrachten den (Multi-)Graph T_2 der jede Kante von T zweimal enthält. Der Grad jedes Knoten in T_2 ist zweimal der Grad dieses Knotens in T, insbesondere ist jeder Grad in T_2 gerade. Ausserdem ist G_2 zusammenhängend. Laut einem Satz aus der Vorlesung enthält T_2 eine Eulertour – also einen Zyklus der jede Kante aus T_2 enthält. Dieser Zyklus besucht offensichtlich jeden Knoten und hat Gewicht $2C$.

Da der Spannbaum aus (a) ein Pfad ist, ist es besonders leicht, T_2 in einen Kreis umzuwandeln: Wir können zum Beispiel folgenden Zyklus verwenden: (A,B,C,G,F,E,F,G,C,B,A,D,A)

Variante 2: Ein solcher Weg Z ist gegeben durch den Weg, den die Tiefensuche benutzt um den Spannbaum T abzusuchen. Immer wenn wir einen Knoten v betrachten, laufen wir zu einem unbesuchten Nachbarn u von v, falls es keinen solchen Nachbarn gibt, laufen wir zum Vorgänger von v. Um genau zu sein, benutzen wir folgende Implementation der Tiefensuche mit Startknoten $s = A$ in T. Wir speichern den Weg der Tiefensuche als $x_1, x_2, \ldots$ ab.

```
t ← 1
v ← s
while v ≠ s oder v hat einen unbesuchten Nachbarn do
    x_t ← v
    t ← t + 1
    if v hat unbesuchten Nachbarn u then
        v ← u
        pred(u) ← v
    else
        v ← pred(v)
```

Der gesuchte Weg Z ist durch $Z = (x_1, x_2, \ldots)$ gegeben. Wir zeigen die Korrektheit des Algorithmus. Z ist sicherlich ein Weg, da x_{t+1} jeweils ein Nachbar von x_t ist (auch der Vorgänger von v ist immer mit v benachbart). Aufgrund der while-Bedingung endet der Algorithmus

in s, somit ist Z ein Zyklus. Da T zusammenhängend ist, besucht die Tiefensuche und somit Z jeden Knoten mindestens einmal. Desweiteren zeigen wir, dass jede Kante $\{u, v\}$ aus T genau zweimal in Z vorkommt, woraus folgt, dass das Gewicht von Z gleich $2C$ ist. O.B.d.A. wird u vor v besucht, dann wird e einmal beim Vorwärtsgehen benutzt (v ist ein unbesuchter Nachbar von u), einmal beim Rückwärtsgehen benutzt (sobald v keine unbesuchten Nachbarn mehr hat) und danach nicht mehr benutzt, da v bereits besucht wurde.

Für das Beispiel von G könnte man zum Beispiel folgenden Zyklus finden:
(D,A,B,C,G,F,E,F,G,C,B,A,D). Beachte, dass dieser Spannbaum vom Startknoten der Tiefensuche abhängt und dass die Reihenfolge in der die Tiefensuche die Nachbarn besucht davon abhängen kann, wie der Graph abgespeichert ist (bzw. in welcher Reihenfolge die Knoten in der Adjazenzliste vorkommen).

(c) Sei V die Knotenmenge von G und sei Z ein geschlossener Weg, der jeden Knoten mindestens einmal besucht. Sei E die Menge der Kanten, die in Z benutzt werden (falls eine Kante mehrmals benutzt wird, kommt sie nur einmal in der Menge E vor). Solange es einen Kreis mit Kanten aus E gibt, entfernen wir eine Kante des Kreises aus E. Die resultierende Kantenmenge bezeichnen wir mit E'. Der Graph (V, E') ist ein Spannbaum. Er ist zusammenhängend, weil der Graph (V, E) zusammenhängend ist und der Graph durch löschen einer Kante eines Kreises jeweils zusammenhängend bleibt, und er ist kreisfrei, da aus jedem Kreis ein Kante gelöscht wurde. Deshalb ist das Gewicht C des minimalen Spannbaums kleiner als das Gewicht der Kanten in E'. Es folgt $C \leq \sum_{e \in E'} \ell(e) \leq \sum_{e \in E} \ell(e) \leq \ell(Z)$.