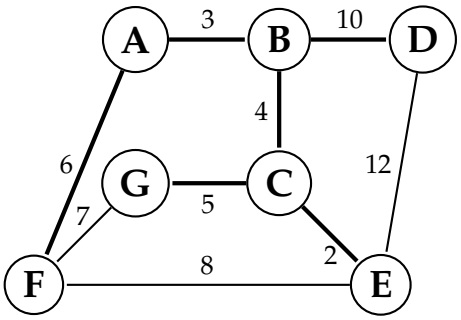
ETH Zürich
Institut für Theoretische Informatik
Prof. Dr. Angelika Steger, Dr. Johannes Lengler
Übungsleitung: Florian Meier

HS 2019

Algorithmen und Komplexität
Lösungsvorschlag zu Übungsblatt 6

Lösungsvorschlag zu Aufgabe 1

Der Graph G mit fett markiertem (eindeutigem) minimalem Spannbaum:



(a) Reihenfolge: $\{C, E\}, \{A, B\}, \{B, C\}, \{C, G\}, \{A, F\}, \{B, D\}$.

(b) Reihenfolge: $\{A, B\}, \{B, C\}, \{C, E\}, \{C, G\}, \{A, F\}, \{B, D\}$.

(c) Die vermutlich eleganteste Lösung verwendet Teilaufgabe (d) und funktioniert wie folgt: Negiere zuerst alle Kantengewichte im Graphen G, bestimme dann einen minimalen Spannbaum T' in dem so angepassten Graphen G' mittels Kruskal oder Prim, und beobachte dann, dass der soeben bestimmte Spannbaum T' einem maximalen Spannbaum T im ursprünglichen Graphen G entspricht. Die Beobachtung folgt, weil ein Spannbaum in G mit noch grösserem Gewicht als T einem Spannbaum in G' mit noch kleinerem Gewicht als T' entsprechen würde. Für beide Algorithmen (Kruskal und Prim) ist es leicht einzusehen, dass das Negieren der Kantengewichte die asymptotische Laufzeit nicht beeinflusst.

Alternativ kann man die Funktionsweise von Kruskal oder Prim auch direkt anpassen: Bei Kruskal werden einfach die Kanten nach absteigenden (anstatt aufsteigenden) Gewichten sortiert und betrachtet. Bei Prim wird in jedem Schritt die jeweils schwerste (anstatt leichteste) Kante hinzugefügt, die den gegenwärtigen Teilbaum mit einem neuen Knoten verbindet. Für beide Algorithmen bleibt die asymptotische Laufzeit wieder unverändert.

Von der Korrektheit des modifizierten Algorithmus von Prim überzeugt man sich, indem man den Beweis von Satz 2.13 durchgeht und jeweils min bzw. max durch max bzw. min ersetzt und die Ungleichheitszeichen umdreht.

Auf den Korrektheitsbeweis von modifizierten Algorithmus von Kruskal gehen wir hier nicht im Detail ein und verweisen wie im Skript auf Kapitel 3.3 im Skript.

(d) Man kann den Fall mit negativen Kantengewichten leicht auf den Fall mit nicht-negativen Kantengewichten reduzieren: Sei w das Gewicht der leichtesten Kante in G, und sei $\hat{w} = |w|$. Dann konstruiert man einen neuen Graphen G' (mit nicht-negativen Kantengewichten) indem man $\hat{w}$ zu allen Kantengewichten in G hinzuaddiert. Lässt man dann entweder Kruskal oder Prim mit Eingabe G und G' laufen, dann sieht man leicht, dass für G und G' derselbe Spannbaum konstruiert wird. Andererseits wird das Gewicht jedes Spannbaums (minimal oder nicht) durch das Ändern der Kantengewichte um denselben Betrag vergrössert, nämlich um $\hat{w}(|V| - 1)$. Insbesondere ist ein minimaler Spannbaum in G auch ein

minimaler Spannbaum in G' und umgekehrt. Die Ausgabe der Algorithmen von Prim und Kruskal auf G ist daher korrekt.

Lösungsvorschlag zu Aufgabe 2

a) Sei T ein minimaler Spannbaum. Nehme an, die eindeutige Kante $e_{min} = \{u, v\}$ mit minimalem Gewicht ist nicht in T. Wenn wir e_{min} zu T hinzufügen, entsteht ein Kreis (Da T zusammenhängend ist, gibt es einen u-v-Pfad in T, durch hinzufügen von e_{min} schliesst sich dieser zu einem Kreis). Durch entfernen einer beliebigen Kante f dieses Kreises entsteht wieder ein Spannbaum T', denn $T' = T \cup \{e\} \setminus \{f\}$ ist nach wie vor zusammenhängen und enthält keinen Kreis. T' hat wegen der Minimalität von e_{min} kleineres Gewicht als T, was ein Widerspruch zur Annahme ist.

b) Nehme an T und T' sind zwei unterschiedliche minimale Spannbäume. Wir führen dies zum Widerspruch indem wir einen leichteren Spannbaum konstruieren. Es sei e die leichteste Kannte die in einem der beiden Spannbäume aber nicht im anderen enthalten ist. O.B.d.A. sei $e \in T$ und $e \notin T'$. Durch hinzufügen von e zu T' entsteht ein Kreis. In diesem Kreis gibt es eine Kante f mit $w(f) > w(e)$, da alle Kanten $e' \in T'$ mit $w(e') < w(e)$ auch in T enthalten sind und es keinen Kreis in T gibt. $T \cup \{e\} \setminus \{f\}$ ist ein Spannbaum mit kleinerem Gewicht als T. Widerspruch!

Lösungsvorschlag zu Aufgabe 3

a) Der Algorithmus stoppt immer durch die angegebene Terminierungsbedingung: Nehme an, dass nicht alle Kanten gefärbt sind und die roten Kanten keinen zusammenhängenden Graphen bilden. Aus letzterem, und da der Graph zusammenhängend ist, gibt es einen Schnitt $(W, V \setminus W)$, der keine rote Kante enthält. Er enthält ausserdem immer mindestens eine ungefärbte Kante: es ist nicht möglich alle Kanten über einen Schnitt grau zu färben, da die letzte davon nicht in einem Kreis enthalten sein kann, der keine anderen grauen Kanten enthält.

   Somit kann immer wenigstens die erste Regel angewendet werden. Da in jedem Schritt eine Kante gefärbt wird, stoppt der Algorithmus in endlicher Zeit (spätestens wenn alle Kanten gefärbt sind).

   Zur Korrektheit zeigen wir durch Induktion über die Anzahl k der gefärbten Kanten:

   Sind k Kanten gefärbt, so gibt es einen minimalen Spannbaum T_k, der alle roten und keine graue Kante enthält.

   Induktionsanfang $k = 0$: Da nach Annahme der Graph zu Beginn ungefärbt ist, können wir für T_0 einfach einen beliebigen minimalen Spannbaum wählen.

   Induktionsschritt $k - 1 \Rightarrow k$: Wir unterscheiden zwei Fälle.

   (a) *Im k-ten Schritt wurde die rote Regel angewandt.* Sei W die dazugehörige Menge und $e_0 \in (W, V \setminus W)$ diejenige Kante, die im k-ten Schritt rot gefärbt wurde. Wenn $e_0 \in T_{k-1}$, so können wir einfach $T_k = T_{k-1}$ setzen. Nehmen wir daher an, dass $e_0 \notin T_{k-1}$. Sei $e_0 = \{w, v\}$ mit $w \in W$ und $v \in V \setminus W$. Da T_{k-1} zusammenhängend ist, gibt es in T_{k-1} einen Pfad P von w nach v. Wegen $w \in W$ und $v \notin W$ muss dieser Pfad P eine Kante $e'_0 = \{w', v'\}$ enthalten mit $w' \in W$ und $v' \notin W$. Dann gilt:

   - e'_0 *ist nicht rot gefärbt.* Dies gilt, da $e'_0 \in (W, V \setminus W)$ und der Schnitt nach Definition der roten Regel keine rot gefärbte Kante enthält.
   - $T_k := (T_{k-1} - e'_0) + e_0$ *ist ein minimal spannender Baum, der alle roten Kanten und keine graue Kanten enthält.* Dass T_k ein Spannbaum ist, folgt unmittelbar aus der Tatsache,

dass e'_0 eine Kante aus dem (eindeutigen) Kreis in $T_{k-1} + e_0$ ist. Dass T_k ein *minimaler* Spannbaum ist, folgt aus der Definition der roten Regel: Da e_0 und e'_0 beides ungefärbte Kanten aus dem Schnitt $(W, V \setminus W)$ sind, gilt $\ell(e_0) \leq \ell(e'_0)$. Da T_{k-1} nach Induktionsannahme alle roten Kanten aus den ersten $k-1$ Schritten und keine graue Kante enthält und e'_0 nicht rot gefärbt war (s.o.) enthält T_k daher alle roten Kanten aus den ersten k Schritten und keine graue Kante.

(b) *Im k-ten Schritt wurde die graue Regel angewandt.* Sei C der dazugehörige Kreis und $e_1 \in C$ diejenige Kante, die im k-ten Schritt grau gefärbt wurde. Wenn $e_1 \notin T_{k-1}$, so können wir einfach $T_k = T_{k-1}$ setzen. Nehmen wir daher an, dass $e_1 \in T_{k-1}$. Dann zerfällt $T_{k-1} - e_1$ in zwei Komponenten und, da C ein Kreis ist, gibt es eine Kante $e'_1 \in C$, so dass $(T_{k-1} - e_1) + e'_1$ wieder ein Baum ist. Nach Definition der grauen Regel kann e'_1 nicht grau gefärbt sein. Wegen $e'_1 \notin T_{k-1}$ kann e'_1 auch nicht rot gefärbt sein. Nach Definition der grauen Regel gilt daher $\ell(e'_1) \leq \ell(e_1)$, woraus folgt, dass $T_k := (T_{k-1} - e_1) + e'_1$ ein minimaler Spannbaum ist.

Aus dem Induktionsbeweis folgt: Nach Terminierung des Algorithmus gibt es einen minimalen Spannbaum $\hat{T}$, der alle roten Kanten und keine grauen Kanten enthält. Aus der Terminierungsbedingung folgt, dass entweder die roten Kanten einen zusammenhängenden Graphen induzieren (also den Baum $\hat{T}$) oder alle Kanten gefärbt wurden (und somit $\hat{T}$ keine ungefärbte und also nur rote Kanten enthält). In jedem Falle gilt somit: Nach Terminierung bilden die roten Kanten einen minimalen Spannbaum.

b) Im Pseudo-Code vom Algorithmus von Prim (Algorithmus 2.8) entspricht F der Menge der roten Kanten. Dann führt Prim in jedem Schritt die rote Regel aus. Dabei ist W jeweils die Menge der bereits besuchten Knoten. Prim wählt die Kante mit minimalem Gewicht zwischen W und $V \setminus W$ und fügt sie zum minimalen Spannbaum F hinzu, was soviel wie rot färben bedeutet.

Auch im Pseudo-Code vom Algorithmus von Kruskal (Algorithmus 2.9) sei F die Menge der roten Kanten. Kruskal testet in jedem Schritt ob $(V, F \cup \{e_i\})$ kreisfrei ist. Falls ja so führt er im wesentlichen die rote Regel aus: Da $(V, F \cup \{e_i\})$ kreisfrei ist, gibt es eine Menge W, sodass keine Kante aus F dafür aber e_i im $(W, V \setminus W)$-Schnitt enthalten ist. Da Kruskal alle Kanten mit kleinerem Gewicht bereits abgearbeitet hat, hat e_i zugleich minimales Gewicht unter den ungefärbten Kanten zwischen W und $V \setminus W$. Er fügt e_i zu F hinzu bzw. färbt e_i rot. Falls $(V, F \cup \{e_i\})$ nicht kreisfrei ist, so führt Kruskal die graue Regel aus. Der gefundene Kreis besteht nur aus roten Kanten welche alle kleineres Gewicht haben als e_i, da sie bereits gefärbt sind. Kruskal ignoriert die Kante e_i was soviel bedeutet wie sie wird grau gefärbt.