Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
M. Fuchs, G. Schmid

Algorithmen und Datenstrukturen
Sommersemester 2026
Musterlösung Übungsblatt 9

Abgabe: Dienstag, 30. Juni, 2026, 10:00 Uhr

Aufgabe 1: Topologische Sortierung *(6 Punkte)*

(a) Sei G ein *gerichteter* Graph definiert durch die folgende Adjazenzmatrix A. Zeichnen Sie den zugehörigen Graph G. *(2 Punkte)*

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

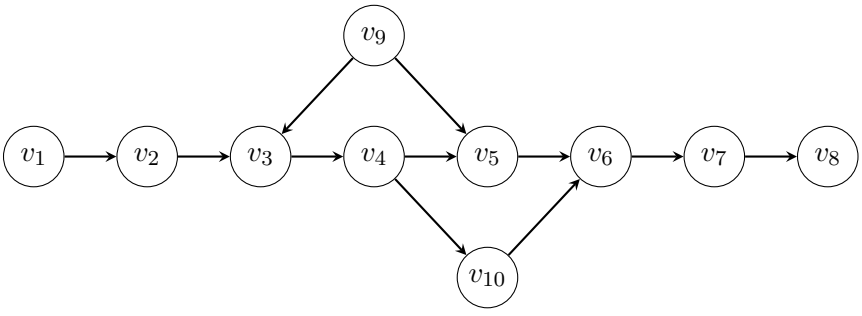
(b) Geben Sie eine topologische Sortierung für G an. *(1 Punkt)*

(c) Wie viele topologische Sortierungen hat G? *(2 Punkte)*

(d) Gegeben zwei zyklenfreie gerichtete Graphen (DAG) $G = (V, E)$ und $G' = (V, E')$ definiert auf derselben Knotenmenge V, so dass folgende Teilmengenbeziehung vorliegt: $E' \subseteq E$. Was können wir über die Anzahl der topologischen Sortierungen von G im Vergleich zu G' sagen? Begründen Sie Ihre Antwort. (Wir suchen hier nach Formulierungen wie "gleich viele", "strikt mehr", "mindestens so viele", "höchstens so viele", usw.) *(1 Punkt)*

Musterlösung

(a) G:

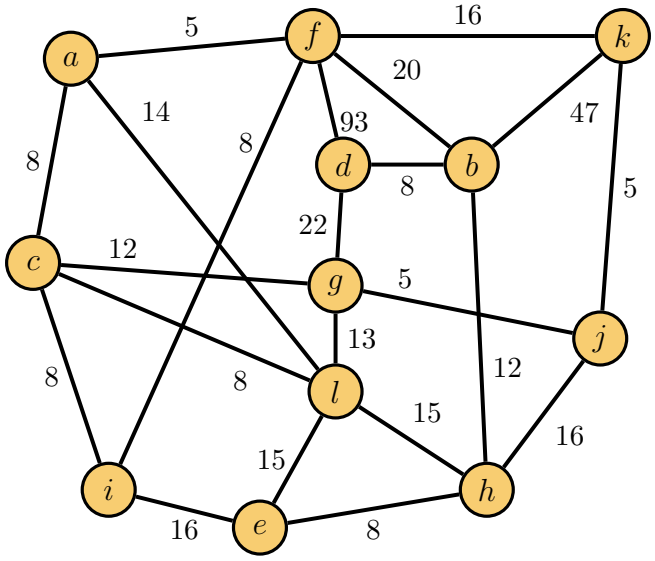


(b) Eine mögliche Sortierung: v_9, v_1, v_2, v_3, v_4, v_5, v_{10}, v_6, v_7, v_8.

(c) Wir wissen, dass v_9 vor v_3 liegen muss und wir wissen dass v_{10} nach v_4 aber vor v_6 sein muss. Entfernt man v_{10} bleiben genau 3 mögliche topologische Sortierungen. Fügen wir nun wieder v_{10} hinzu, können wir diesen entweder direkt nach v_4 oder nach v_5 hinzufügen. In jeder der 3 Sortierungen gibt es also 2 Möglichkeiten für v_{10}, macht insgesamt $3 \cdot 2 = 6$ Sortierungen.

(d) Jede topologische Sortierung von G ist automatisch auch eine topologische Sortierung von G'. Somit hat G' mindestens so viele Sortierungen wie G.

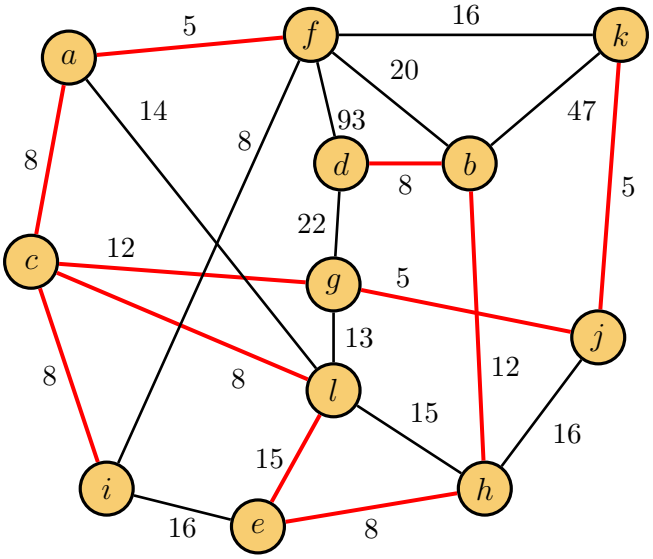
Aufgabe 2: Prims Algorithmus *(4 Punkte)*



Führen Sie Prims Algorithmus zur Berechnung eines MST aus. Beachten Sie dabei folgende Punkte:

- Starten Sie mit Knoten a.
- Wenn der Algorithmus die Wahl zwischen mehreren Kanten hat, soll immer diejenige Kante (u, v) (mit $u \in A$ und $v \notin A$) genommen werden, die lexikographisch kleiner ist (man würde also zum Beispiel (a, c) vor (f, i) wählen oder (c, i) vor (f, i)).
- Markieren Sie die Kanten im finalen Spannbaum.
- Schreiben Sie auf in welcher Reihenfolge die Kanten zum Baum hinzugefügt werden.
- Was ist das Gesamtgewicht des finalen MST?

Musterlösung



Die Kanten werden in folgender Reihenfolge hinzugefügt. Das Gesamtgewicht beträgt 94.

1. $(a, f) : 5$
2. $(a, c) : 8$
3. $(c, i) : 8$
4. $(c, l) : 8$
5. $(c, g) : 12$
6. $(g, j) : 5$
7. $(j, k) : 5$
8. $(l, e) : 15$
9. $(e, h) : 8$
10. $(h, b) : 12$
11. $(b, d) : 8$

Aufgabe 3: Eindeutige Minimale Spannbäume *(10 Punkte)*

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

(a) Beweisen Sie, dass wenn G paarweise veschiedene Kantengewichte hat, es einen *eindeutigen* minimalen Spannbaum gibt. *(5 Punkte)*

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

Starte mit $T' = (V, E)$. Suche einen Kreis C in T' und lösche die schwerste Kante e aus C. Wiederhole bis keine Kreise mehr in T' sind.

(5 Punkte)

Musterlösung

(a) Seien T und T' zwei minimale Spannbäume mit Kanten $e_1, \ldots, e_{n-1}$ und $e'_1, \ldots, e'_{n-1}$, jeweils aufsteigend nach Gewicht sortiert. Angenommen 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 eine 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 $k > j$) und somit auch $w(e'_j) > w(e_j) \geq w(e_k)$. Da $(T' \setminus \{e'_j\}) \cup \{e_k\}$ wieder ein Spannbaum ist und dieser ein kleineres Gewicht als T' hat, haben wir einen Widerspruch zur Annahme dass T' minimal ist.

(b)

- T' ist ein Spannbaum:
  Nach Definition des Algorithmus enthält der resultierende Graph T' keine Kreise mehr. Da das Löschen einer Kante aus einem Kreis die Anzahl der Zusammenhangskomponenten eines Graphen nicht erhöht (einfach aus der Definition eines Kreises, dass man von jedem Knoten zu jedem anderen kommt egal ob man im Uhrzeigersinn oder im Gegenuhrzeigersinn den Kreis abläuft. Das löschen einer Kante "blockiert" nur einen der beiden Richtungen), bleibt der Graph stets zusammenhängend. Ein zusammenhängender, kreisfreier Graph, der alle Knoten umfasst, ist per Definition ein Spannbaum. Somit ist T' ein Spannbaum.
- Wenn e die schwerste Kante auf einem Kreis ist, dann gibt es einen MST der nicht e enthält.
  Sei $e = (u, v)$ eine schwerste Kante auf einem Kreis C in T'. Wir zeigen, dass das Löschen von e die Existenz eines minimalen Spannbaums nicht zerstört. Sei dazu T ein minimaler Spannbaum von G.

  Falls $e \notin T$, ist nichts zu zeigen. Sei also $e \in T$. Dann zerfällt $T \setminus \{e\}$ in zwei Zusammenhangskomponenten mit Knotenmengen S und $V \setminus S$, wobei $u \in S$ und $v \in V \setminus S$ gilt. Da C ein Kreis ist und e auf C liegt, enthält C neben e noch eine weitere Kante e', die ebenfalls den Schnitt $(S, V \setminus S)$ kreuzt. Da e eine schwerste Kante auf C ist, gilt $w(e') \leq w(e)$.

  Somit ist

  $$T'' := (T \setminus \{e\}) \cup \{e'\}$$

  wieder ein Spannbaum und es gilt

  $$w(T'') = w(T) - w(e) + w(e') \leq w(T).$$

  Da T minimal ist, ist auch T'' ein minimaler Spannbaum. Außerdem enthält T'' die Kante e nicht. Folglich existiert ein minimaler Spannbaum, der e nicht enthält.
- T' ist ein MST:
  Wir haben oben gezeigt dass T' ein Spannbaum ist und wir haben gezeigt dass für jede Kante $e \notin T'$, es einen MST gibt der e nicht enthält. Da T' gerade der Spannbaum ist der alle diese schwersten e nicht enthält muss dieser ein MST sein.

Aufgabe 4: Bonus: Problem des Handlungsreisenden *(7* 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 Reihenfolge von Städten, so dass keine Stadt ausser der ersten mehr als einmal besucht wird. 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 $(i_1, ..., i_n)$ mit kleiner Gewichtssumme $w(p_{i_n}, p_{i_1}) + \sum_{j=1}^{n-1} w(p_{i_j}, p_{i_{j+1}})$.

Wir wollen nun das zeigen, dass die Knoten eines minimalen Spannbaumes in pre-order[3] Reihenfolge (ausgehend von einer beliebigen Wurzel) eine Rundreise in G ergeben, deren gesamtstrecke höchstens doppelt so lang ist wie die minimale Rundreise. Folgende Teilaufgaben sind hierfür hilfreich. Sei $R = (i_1, \ldots, i_n)$ eine optimale Rundreise und sei $w(R) := w(p_{i_n}, p_{i_1}) + \sum_{j=1}^{n-1} w(p_{i_j}, p_{i_{j+1}})$ die Gesamtstrecke (bzw Gewicht) von R. Sei ausserdem T ein minimaler Spannbaum von G und sei $W_T = \sum_{e \in T} w(e)$ das Gewicht von T. Sei ausserdem $P_T = (\ell_1, \ell_2, ..., \ell_n)$ die pre-order traverierung von T. Da wir uns ja auf einem vollständigen Graphen befinden ist P_T ja ebenso eine Rundreise, bei der wir nicht die Baumkanten entlang laufen, sondern eben $\ell_1 \to \ell_2 \to ... \to \ell_{n-1} \to \ell_n \to \ell_1$. Somit können wir uns $W_{PRE} = w(p_{\ell_n}, p_{\ell_1}) + \sum_{j=1}^{n-1} w(p_{\ell_j}, p_{\ell_{j+1}})$ als das Gewicht der Rundreise die entsteht wenn man P_T abläuft definieren.

(a) Zeigen Sie dass $W_{PRE} \leq 2 \cdot W_T$ gilt. *(5 bonus Punkte)*

(b) Zeigen Sie dass $W_T \leq w(R)$ gilt. *(2 bonus Punkte)*

Die Aussage dass die durch die preorder Traversierung erzeugte Rundreise höchstens doppelt so lange wie die optimale Rundreise ist folgt direkt aus (a) und (b) mit $W_{PRE} \leq 2 \cdot W_T \leq 2w(R)$.

Musterlösung

(a) Eine vollständige Tiefensuche auf T, bei der jede Baumkante einmal hinunter und einmal hinauf durchlaufen wird, hat Gewicht $2W_T$. Die Preorder-Tour entsteht daraus durch Abkürzungen bereits besuchter Knoten. Wegen der Dreiecksungleichung können solche Abkürzungen die Länge nicht erhöhen. Daher gilt $W_{PRE} \leq 2W_T$.

(b) Wenn man von der minimalen Rundreise R die letzte Kante weglässt, also die Kante die den Kreis schließt, dann ergibt R ebenso einen Spannbaum (da jeder Knoten genau einmal besucht wird). Wir nennen diesen Baum T_R mit dem Gewicht $w_{T_R} = \sum_{j=1}^{n-1} w(p_{i_j}, p_{i_{j+1}}) = w(R) - w(p_{i_n}, p_{i_1})$. Da T per Definition ein minimaler Spannbaum ist gilt $W_T \leq W_{T_R}$. Die Aussage folgt direkt:

$$W_T \leq W_{T_R} = w(R) - w(p_{i_n}, p_{i_1}) \leq w(R)$$

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] Pre-order wurde in der Vorlesung nur auf binären Bäumen definiert, diese kann aber auf natürliche Weise verallgemeinert werden, sprich, man besucht erst die Wurzel und dann dann rekursiv die Teilbäume von links nach rechts.