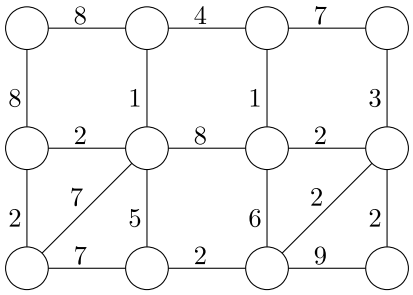
Datenstrukturen und Algorithmen, SS 2019
Prof. Dr. P. Rossmanith
S. Dollase, J. Dreier, H. Lotze

Datum: 27.06.2019

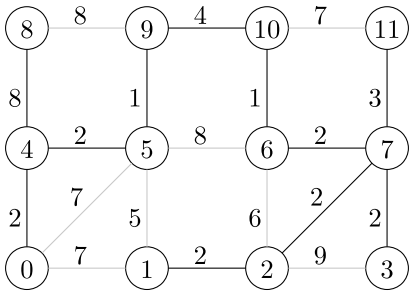
Übungsblatt mit Lösungen 11

Aufgabe T37

Finden Sie jeweils einen minimalen Spannbaum mithilfe der Algorithmen von Kruskal und Prim für folgenden Graphen:



Lösungsvorschlag



Aufgabe T38

Gegeben sind n sportliche Aktivitäten mit je einem zugeordneten Zeitintervall. Die $i$te sportliche Aktivität dauert dabei von Zeitpunkt s_i bis f_i mit $s_i < f_i$.
Entwerfen Sie einen effizienten Algorithmus, der eine möglichst große Menge an sportlichen Aktivitäten auswählt, sodass sich keine ausgewählten Aktivitäten überlappen. Was ist die asymptotische Laufzeit ihres Algorithmus? Hinweis: Verwenden Sie einen Greedy-Algorithmus.

Lösungsvorschlag

Zunächst sortieren wir die Aktivitäten ihrem Wert f_i, sodass die frühste Endzeit als erstes betrachtet wird. Dann gehen wir greedy vor:

1. Wähle Aktivität mit kleinstem f_i.
2. Lösche i und alle Aktivitäten, die sich mit i überschneiden.
3. Falls Menge der Aktivitäten nicht leer, gehe zu 1.

Die Laufzeit beträgt $O(n \log n)$, da die Zeiten sortiert werden müssen.
Da jedes gewählte Intervall die frühste Endzeit von allen übrigen hat, werden in Schritt 2 nur Intervalle entfernt die sich alle gegenseitig schneiden, und deshalb nur maximal eins aus der Menge gewählt werden kann. Demnach ist die greedy-Lösung auch optimal.

Aufgabe T39

Gegeben sei eine $\mathbf{Z}^{n\times 4}$-Matrix. Der *Wert* einer Zeile dieser Matrix sei die Summe der Einträge dieser Zeile. Wir wollen eine linear unabhängige Menge an Zeilen auswählen, sodass die Summe ihrer Werte maximal ist.

Geben Sie einen Algorithmus an, welcher dieses Problem optimal löst. Beweisen Sie, dass ihr Algorithmus optimal ist.

Lösungsvorschlag

Wir berechnen zuerst für jede Zeile ihren Wert und sortieren diese absteigend nach dem berechneten Wert. Wir nennen die Menge der noch nicht betrachteten Zeilen S und unsere Zielmenge $\mathcal{T}$. Anfangs beinhalte S alle Zeilen und $\mathcal{T}$ sei leer. Wir wiederholen nun folgendes, bis wir 4 Zeilen zu $\mathcal{T}$ hinzugefügt haben (mehr können nicht linear unabhängig sein) oder S leer ist:

Betrachte die vom Wert größte verbleibende Zeile $z \in S$ und setze $S \leftarrow S \setminus \{z\}$. Wenn z linear unabhängig zu den Zeilen in $\mathcal{T}$ ist, setze $\mathcal{T} \leftarrow \mathcal{T} \cup \{z\}$. Verwerfe z ansonsten.

Korrektheit: Wir behaupten $(Z, \mathcal{T})$ mit Z Menge der Zeilen und $\mathcal{T} \subseteq 2^Z$ als Familie linear unabhängiger Zeilen ist ein Matroid. Demnach wäre der genannte Greedy-Algorithmus optimal.

Wir müssen folgende zwei Eigenschaften aus der Vorlesung zeigen:

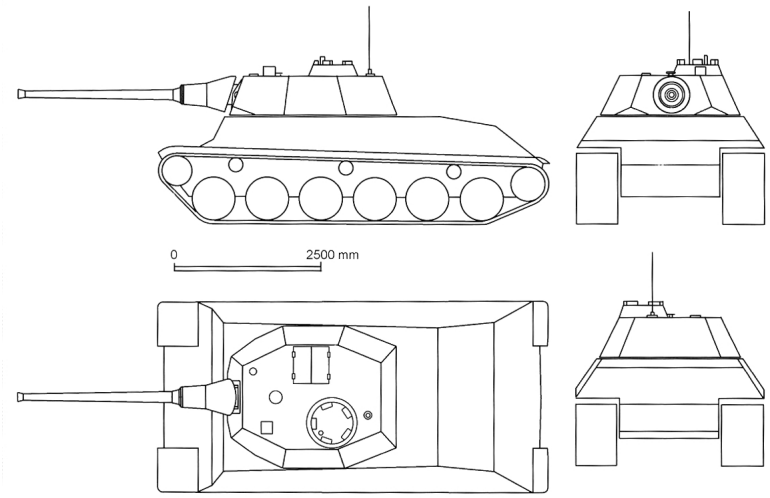
1. Falls $A \subseteq B$ und $B \in \mathcal{T}$, dann $A \in \mathcal{T}$ (M ist hereditär).
2. Falls $A, B \in \mathcal{T}$ und $|A| < |B|$ dann gibt es ein $x \in B$ so dass $A \cup \{x\} \in \mathcal{T}$ (M hat die Austauscheigenschaft).

Die erste Eigenschaft ist trivial: Eine linear unabhängige Menge an Vektoren bleibt linear unabhängig, wenn wir einen der Vektoren aus dieser Menge entfernen.

Die zweite Eigenschaft könenn wir per Widerspruch verifizieren: Angenommen die zweite Bedingung gilt nicht, dann gibt es zwei unabhängige Mengen A und B mit $|A| < |B|$, sodass sich jedes Element in B als Linearkombination von Elementen in A darstellen lässt. Insbesondere hat der durch B aufgespannte Vektorraum Dimension $|A|$. Da $|B| > |A|$, ist die Menge B also nicht linear unabhängig, was ein Widerspruch ist.

Damit ist $(Z, \mathcal{T})$ ein Matroid.

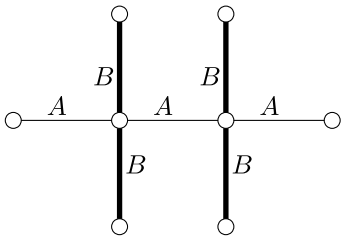
Aufgabe T40



Lösungsvorschlag

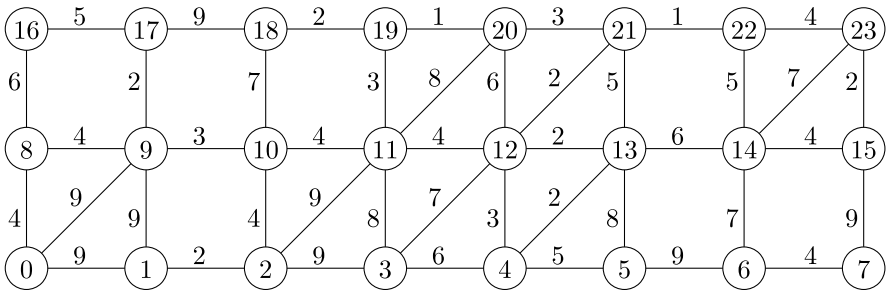
Aufgabe T41
Es sei $G = (V, E)$ ein Graph. Wir definieren $\mathcal{P} \subseteq 2^E$ folgendermaßen: $P \in \mathcal{P}$ genau dann, wenn die Zusammenhangskomponenten des Graphen (V, P) isolierte Knoten oder Pfade sind. Ist $(E, \mathcal{P})$ ein Matroid?

Lösungsvorschlag
Wir zeigen, dass $(V, \mathcal{P})$ nicht die Austauscheigenschaft erfüllt, und somit kein Matroid ist. Die Austauscheigenschaft besagt, falls $A, B \in \mathcal{P}$ und $|A| < |B|$ dann gibt es ein $x \in B$ sodass $A \cup \{x\} \in \mathcal{P}$.
Die zweite Eigenschaft ist nicht für alle Paare an Mengen aus $\mathcal{P}$ erfüllt. Betrachte dazu zum Beispiel folgenden Graphen:

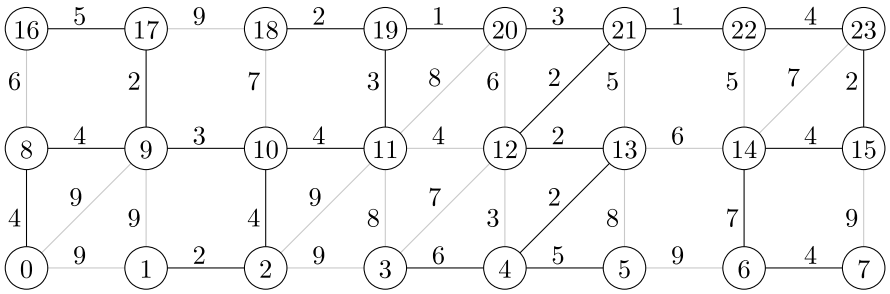


In diesem Beispiel ist $|B| > |A|$ und sowohl A als auch B sind unabhängig. Allerdings ist A kein Pfad mehr, wenn eine Kante aus B zu A hinzugefügt wird.
Die angegebene Struktur ist damit kein Matroid.

Aufgabe H31 (10 Punkte)
Finden Sie einen minimalen Spannbaum mithilfe des Algorithmus von Kruskal, Prim oder Borůvka[1] für folgenden Graphen. Es reicht aus, wenn Sie den resultierenden, minimalen Spannbaum angeben.



Lösungsvorschlag



Aufgabe H32 (10 Punkte)
Gegeben sei ein Graph $G = (V, E)$. Wir definieren eine Menge $\mathcal{W} \subseteq 2^E$ mit $W \in \mathcal{W}$ genau dann, wenn eine Kante $e \in W$ exisiert, sodass $(V, W \setminus \{e\})$ ein Wald ist. Zeigen oder widerlegen Sie, dass $(E, \mathcal{W} \cup \{\emptyset\})$ ein Matroid ist.

[1] `https://dml.cz/bitstream/handle/10338.dmlcz/500114/Boruvka_01-0000-6_1.pdf`

Lösungsvorschlag
$(E, \mathcal{W})$ ist ein Matroid falls er folgende Eigenschaften erfüllt.

1. Falls $A \subseteq B$ und $B \in \mathcal{W}$, dann $A \in \mathcal{W}$.
2. Falls $A, B \in \mathcal{W}$ und $|A| < |B|$ dann gibt es ein $e \in B$ so dass $A \cup \{e\} \in \mathcal{W}$.

Wir zeigen, dass $(E, \mathcal{F})$ beide Eigenschaften erfüllt und somit ein Matroid ist. Die erste Eigenschaft ist erfüllt, da das Entfernen einer Kante aus einem Graphen keine Kreise erzeugen kann.
Wir zeigen nun die zweite Eigenschaft über eine Fallunterscheidung: Nehmen wir an, weder A noch B enthalte einen Kreis. Dann folgt die Aussage unmittelbar aus dem Beweis für den graphischen Matroiden aus der Vorlesung.
Falls A keinen Kreis enthält, können wir eine beliebige Kante zu A hinzufügen, diese kann maximal einen Kreis schließen.
Es verbleibt der Fall, in dem sowohl A als auch B bereits einen Kreis enthalten. Wir nehmen nun aus A und aus B jeweils eine Kante e_A, e_B hinaus, welche sie zu Wäldern zerfallen lässt. B ist nun immer noch größer als A, d.h. können wieder nach dem Beweis für den graphischen Matroiden aus der Vorlesung verwenden, dass A anschließend Kreisfrei ist. Nun fügen wir die vorher herausgenommmenen Kanten einfach wieder hinzu.
Damit ist $(E, \mathcal{F})$ ein Matroid.

Aufgabe H33 (10 Punkte)
Das klassische *Rucksackproblem* ist wie folgt definiert. Gegeben seien n Gegenstände welche jeweils ein Gewicht $w_i > 0$ und einen Wert $v_i > 0$ besitzen. Es gilt nun einen Rucksack, welcher ein Gesamtgewicht von maximal $b > 0$ Gewichtseinheiten tragen kann, so zu packen, dass der Wert der enthaltenen Gegenstände maximiert wird. Dabei muss für jeden Gegenstand entschieden werden, ob dieser eingepackt wird oder nicht. Man sucht also eine Menge $I \subseteq \{1, \ldots, n\}$ sodass $\sum_{i \in I} w_i \leq b$, welche $\sum_{i \in I} v_i$ maximiert. Dieses Problem ist im allgemeinen schwierig zu lösen.
Wir betrachten nun eine Abwandlung dieses Problems, das *fraktionale Rucksackproblem*. Wir dürfen nun zusätzlich Gegenstände beliebig zerteilen. Für ein beliebiges $0 \leq \lambda_i \leq 1$ dürfen wir einen Teil des Gegenstands i mit Gewicht $\lambda_i w_i$ und Wert $\lambda_i v_i$ einpacken. Man sucht also eine Sequenz $\lambda_1, \ldots, \lambda_n$ mit $0 \leq \lambda_i \leq 1$ und $\sum_{i=1}^{n} \lambda_i w_i \leq b$, welche $\sum_{i=1}^{n} \lambda_i v_i$ maximiert.
Geben Sie einen effizienten Algorithmus an, welcher das fraktionale Rucksackproblem optimal löst. Argumentieren Sie, warum ihr Algorithmus optimal ist.

Lösungsvorschlag
Wir lösen dieses Problem greedy. Unser Algorithmus arbeitet wie folgt: Für jeden Gegenstand berechnen wir seinen Wert relativ zu seinem Gewicht, d.h. für jeden Gegenstand i berechnen wir $rel_i = \frac{v_i}{w_i}$. Anschließend sortieren wir die Gegenstände absteigend nach rel_i. Wir packen nun so lange Gegenstände mit dem relativ größten Gewicht in unseren Rucksack, bis dieser entweder voll ist, oder noch Kapazitäten über sind, der aktuell betrachtete Gegenstand aber nicht mehr komplett in den Rucksack passt. In letzterem Fall zerteilen wir den Gegenstand so, dass wir mit diesem den Rest des Rucksacks auffüllen.
Die Optimalität dieses Algorithmus ist leicht zu zeigen. Zuerst beobachten wir, dass durch die Zerteilung eines Gegenstandes dessen relativer Wert nicht sinkt, da wir dabei das Gewicht und den Wert des Gegenstandes um den gleichen Betrag dividieren, sich diese in der Definition also aufheben.
Die Sortierung nach absteigendem relativen Wert gibt uns nun an, für welche Einheit an Gewicht wir den meisten Wert erhalten. Damit maximieren wir den Gewinn, welchen wir pro Gegenstand erreichen können. Sollten wir den letzten Gegenstand zerteilen müssen, zeigt uns unsere obige Beobachtung, dass wir weiterhin optimal sind.