



Übungsblatt 10

Algorithmen I - Sommersemester 2024

Abgabe im ILIAS bis 05.07.2024, 18:00 Uhr

Die Abgabe erfolgt alleine oder zu zweit als eine PDF-Datei über das Übungsmodul in der Gruppe eures Tutoriums im ILIAS. Beschriftet die Abgabe deutlich mit Matrikelnummer und Name.

- Achtet bei handschriftlichen Abgaben auf Lesbarkeit.
- Achtet darauf, ob Algorithmen in Worten oder Pseudocode beschrieben werden sollen.
- Es werden immer asymptotisch möglichst effiziente Algorithmen erwartet, wenn nicht anders angegeben.
- Wenn Korrektheits- oder Laufzeitanalysen gefordert sind, behandelt diese separat von der Algorithmenbeschreibung.

Gesamtpunkte: 20 (+ 137 Bonus)

Aufgabe 1 - Tree-nanigans: Auf der Suche nach dem richtigen Weg mit KI und Tiefensuche - Fehlerfindung à la Forrest Gump (7 Punkte)

Das Erstellen von Aufgaben und Musterlösungen ist harte Arbeit. Daher wollten wir in dieser Aufgabe etwas Zeit sparen und uns Musterlösungen mit Hilfe von ChatGPT generieren lassen. Ursprünglich sollte die Aufgabe sein die folgende Aussage über semi-zusammenhängende¹ Graphen zu beweisen oder zu widerlegen:

Jeder gerichtete, azyklische Graph mit eindeutiger Quelle² ist semi-zusammenhängend.

Für diese Aussage ließen wir die KI mehrere Beweise mit verschiedenen Ansätzen generieren. Im folgenden haben wir die Antworten durch eine Nummerierung der Sätze ergänzt.

¹Wir nennen einen gerichteten Graphen $G = (V, E)$ *semi-zusammenhängend*, wenn für alle Paare von Knoten $u, v \in V$ gilt, dass u von v aus erreichbar ist oder v von u aus (oder beides).

²Eine Quelle ist ein Knoten ohne eingehende Kanten.

Prompt (direkter Beweis)

Beweise die folgende Aussage: jeder gerichtete azyklische Graph mit eindeutiger Quelle ist semi-zusammenhängend.

Antwort

- 1 Um die Aussage zu beweisen, dass jeder gerichtete azyklische Graph (DAG) mit einer eindeutigen Quelle semi-zusammenhängend ist, müssen wir zuerst die Definitionen klären:
 - 1.1 Ein gerichteter Graph ist azyklisch, wenn es darin keinen Zyklus gibt, das heißt, es gibt keine Möglichkeit, einen Pfad zu folgen und dabei zu einem Knoten zurückzukehren.
 - 1.2 Eine eindeutige Quelle in einem gerichteten Graphen ist ein Knoten, der keine eingehenden Kanten hat.
 - 1.3 Ein Graph ist semi-zusammenhängend, wenn es zwischen jeder möglichen Kombination von Knoten, mit Ausnahme der Quelle, mindestens einen gerichteten Pfad gibt.
- 2 Um zu zeigen, dass ein gerichteter azyklischer Graph mit einer eindeutigen Quelle semi-zusammenhängend ist, betrachten wir:
- 3 *Existenz eines Pfades zwischen jedem Knotenpaar (außer der Quelle):*
 - 3.1 Da der Graph azyklisch ist, gibt es zwischen jedem Knotenpaar einen gerichteten Pfad.
 - 3.2 Wenn wir einen Pfad zwischen zwei Knoten betrachten, die nicht die Quelle sind, können wir diesen Pfad verwenden, um eine Verbindung zwischen diesen Knoten herzustellen.
- 4 *Verbindung zur Quelle:*
 - 4.1 Da die Quelle des Graphen eindeutig ist, kann kein anderer Knoten direkt mit der Quelle verbunden sein.
 - 4.2 Aber da der Graph azyklisch ist, kann jeder Knoten über einen gerichteten Pfad mit der Quelle verbunden werden.
- 5 *Folgerung:*
 - 5.1 Somit gibt es zwischen jedem Knotenpaar im Graphen, mit Ausnahme der Quelle, mindestens einen gerichteten Pfad.
 - 5.2 Daher ist der Graph semi-zusammenhängend.
- 6 Dieser Beweis zeigt, dass ein gerichteter azyklischer Graph mit einer eindeutigen Quelle semi-zusammenhängend ist.

Prompt (Widerspruchsbeweis)

Bitte zeige die Aussage mit Beweis durch Widerspruch.

Antwort

- 1 Um die Aussage „Jeder gerichtete azyklische Graph mit eindeutiger Quelle ist semi-zusammenhängend“ durch Widerspruch zu beweisen, nehmen wir an, es gibt einen gerichteten azyklischen Graphen mit einer eindeutigen Quelle, der nicht semi-zusammenhängend ist.
- 2 Angenommen, es gibt einen gerichteten azyklischen Graphen G mit einer eindeutigen Quelle q , der nicht semi-zusammenhängend ist.
- 3 Das bedeutet, dass es mindestens ein Paar von Knoten u und v gibt, für das es keinen gerichteten Pfad von u nach v gibt, außer wenn u oder v die Quelle q ist.
- 4 Da der Graph azyklisch ist, kann es keinen Pfad von v nach u geben, da ein solcher Pfad einen Zyklus erzeugen würde.
- 5 Daher müssen wir nur den Fall betrachten, dass es keinen gerichteten Pfad von u nach v gibt, außer wenn u die Quelle q ist.
- 6 Wenn u nicht die Quelle q ist, dann gibt es keinen Pfad von u nach v .
- 7 Aber da G azyklisch ist, muss u über einen Pfad mit q verbunden sein.
- 8 Da es nur einen Weg von q nach u gibt, muss dieser Pfad direkt sein.
- 9 Nun betrachten wir v .
- 10 Da v nicht die Quelle q ist und es keinen Pfad von u nach v gibt (außer wenn u die Quelle ist), muss es auch einen direkten Pfad von q nach v geben.
- 11 Aber das ist ein Widerspruch zur Annahme, dass es keinen Pfad von u nach v gibt, außer wenn u die Quelle ist.
- 12 Also ist unsere Annahme, dass es einen gerichteten azyklischen Graphen mit einer eindeutigen Quelle gibt, der nicht semi-zusammenhängend ist, falsch.
- 13 Folglich ist jeder gerichtete azyklische Graph mit einer eindeutigen Quelle semi-zusammenhängend.

Prompt (Induktionsbeweis)

Zeige nun die Aussage mit Induktion.

Antwort

- 1 Um die Aussage „Jeder gerichtete azyklische Graph mit eindeutiger Quelle ist semi-zusammenhängend“ mit Induktion zu beweisen, gehen wir folgendermaßen

vor:

- 2 *Induktionsanfang*: Betrachten wir den einfachsten Fall, einen gerichteten azyklischen Graphen mit nur zwei Knoten u und v , wobei u die Quelle ist.
- 3 In diesem Fall gibt es einen gerichteten Pfad von u nach v , da es keine anderen Knoten gibt, die den Pfad blockieren könnten.
- 4 Daher ist der Graph semi-zusammenhängend.
- 5 *Induktionsannahme*: Angenommen, die Aussage gilt für jeden gerichteten azyklischen Graphen mit einer eindeutigen Quelle und n Knoten, wobei $n \geq 2$.
- 6 *Induktionsschritt*: Wir wollen zeigen, dass die Aussage auch für einen gerichteten azyklischen Graphen mit $n + 1$ Knoten und einer eindeutigen Quelle gilt.
- 7 Betrachten wir einen solchen Graphen G' mit $n + 1$ Knoten und einer eindeutigen Quelle q .
- 8 Wir entfernen nun einen beliebigen Knoten w aus G' , wodurch wir einen Graphen G mit n Knoten erhalten.
- 9 Da G' azyklisch ist, bleibt G azyklisch.
- 10 Nach der Induktionsannahme ist G semi-zusammenhängend, da es n Knoten hat und eine eindeutige Quelle.
- 11 Das bedeutet, dass es zwischen jedem Knotenpaar in G , außer der Quelle, mindestens einen gerichteten Pfad gibt.
- 12 Jetzt fügen wir den entfernten Knoten w wieder hinzu, um G' zu erhalten.
- 13 Da w kein Zyklus verursacht, kann es keinen gerichteten Pfad zwischen w und einem anderen Knoten außer q geben, da sonst ein Zyklus entstehen würde.
- 14 Da q die einzige Quelle ist, ist der einzige Weg, um w mit anderen Knoten zu verbinden, über q .
- 15 Aber da es bereits einen Pfad von q zu jedem anderen Knoten gibt, bleibt dieser Pfad bestehen, wenn w hinzugefügt wird.
- 16 Somit gibt es zwischen jedem Knotenpaar in G' , außer der Quelle, mindestens einen gerichteten Pfad.
- 17 Da der Induktionsanfang gezeigt wurde und der Induktionsschritt korrekt ist, folgt daraus, dass die Aussage für alle gerichteten azyklischen Graphen mit einer eindeutigen Quelle gilt.

Bei der lästigen Qualitätsprüfung wurde allerdings angemerkt, dass die generierten Beweise voller Fehler sind. Wir setzen deshalb nun auf eure Hilfe um Fehler in den Antworten zu identifizieren.

Im folgenden sind *Fehler* für uns falsche Folgerungen, falsche Anwendungen einer Beweistechnik, unvollständige bzw. fehlerhafte Definitionen oder auch falsche Teilaussagen. Explizit nicht zählen wir Rechtschreibfehler, Grammatikfehler, ausgedachte Namen und die vorgegebene Aussage. Ebenso sind eine falsche Folgerung und die gefolgerte falsche Aussage nicht als getrennte Fehler zu betrachten. Der selbe Fehler darf mehrfach auftreten.

- a) Identifiziere zehn Fehler in den Beweisen und Definitionen und gib an, in welcher Zeile du diese findest und warum diese falsch sind. (5 Punkte)
- b) Zeige oder widerlege die ursprüngliche Aussage. (2 Punkte)

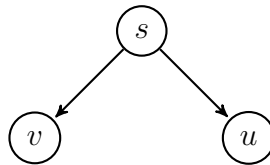
Hinweis: Ebenfalls wurde künstliche Intelligenz zur Generierung von Aufgabentiteln eingesetzt.

Lösung 1

- a) Hier findet sich eine unvollständige Fehlerliste:
 - Direkter Beweis:
 - In 1.2 wird nur eine Quelle definiert. Eine eindeutige Quelle muss noch zusätzlich der einzige solche Knoten sein.
 - In 1.3 darf die Quelle nicht ausgenommen werden.
 - Die Folgerung in 3.1. ist aus der Luft gegriffen.
 - In 4.1. ist unklar was mit verbunden gemeint ist. Falls eine Kante (unabhängig von ihrer Richtung) gemeint ist, kann die Quelle sehr wohl ausgehende Kanten haben.
 - Fehler in 4.2: Siehe 3.1.
 - Widerspruchsbeweis:
 - In Zeile 3 darf die Quelle nicht ausgeschlossen werden.
 - Da es nach Annahme keinen Pfad von u nach v gibt, muss ein Pfad von v nach u nicht zwingend einen Zyklus induzieren (Zeile 4).
 - In Zeile 7 wird erneut ein Pfad aus der Azyklizität gefolgert.
 - Zeile 8: Aus der Eindeutigkeit folgt nicht die Direktheit.
 - Zeile 10: Aus der Nichtexistenz des Pfades folgt nicht, dass ein anderer Pfad existiert
 - Induktionsbeweis:
 - Zeile 3: Die Begründung ist nicht ausreichend. Wenn es die Kante uv nicht gibt, wird der Pfad ebenfalls nicht blockiert.

- Laut Zeile 7 soll ein Induktionsschritt von n auf $n+1$ Knoten erfolgen, in Zeile 8 wird dann aber ein Knoten entfernt, was einem Schritt von $n+1$ nach n entspricht.
- In Zeile 10 ist falsch angenommen, dass G eine eindeutige Quelle hat.
- In Zeile 11 muss die Quelle nicht ausgenommen werden.
- Ein Pfad von w zu beliebigen Knoten ist möglich ohne Zyklen zu erzeugen (Zeile 13).
- Somit kann, entgegen Zeile 14, w auch mit anderen Knoten verbunden werden.

- b) Im folgenden Gegenbeispiel ist s die eindeutige Quelle. Es gibt keinen gerichteten Kreis, jedoch ist u weder von v noch v von u erreichbar.



Aufgabe 2 - One cut-vertex is all it takes, falling apart with me (8 Punkte)

In einem Graph $G = (V, E)$ bezeichnen wir einen Knoten $v \in V$ als Cut-Vertex, wenn G durch Löschen von v in mehrere Zusammenhangskomponenten zerfällt. In dieser Aufgabe wollen wir alle Cut-Vertices eines Graphen mithilfe von DFS finden.

Hinweis: Wie immer, wenn dies nicht anders spezifiziert ist, nehmen wir G als einen ungerichteten Graphen an.

- a) Zeige, dass die Wurzel s eines DFS-Baums genau dann ein Cut-Vertex von G ist, wenn sie mehr als ein Kind hat. (2 Punkte)
- b) Zeige, dass ein Knoten v , der nicht die Wurzel des DFS-Baums ist, genau dann ein Cut-Vertex von G ist, wenn gilt: v hat ein Kind u im DFS-Baum, sodass es keine Rückkante aus dem Teilbaum unter u zu Vorfahren von v gibt. (2 Punkte)
- c) Beschreibe, wie mithilfe der low-Werte entschieden werden kann, ob ein (nicht Wurzel-) Knoten ein Cut-Vertex ist. Begründe deine Antwort. (1 Punkt)
- d) Gib einen Algorithmus an, der in $O(n + m)$ Zeit alle Cut-Vertices in G bestimmt. Begründe dessen Korrektheit und dass er die geforderte Laufzeit hat. (3 Punkte)

Lösung 2

- a) Sei $s \in V$ die Wurzel eines DFS-Baumes von G . Wir zeigen:

s hat mehr als ein Kind im DFS-Baum $\Leftrightarrow s$ ist ein Cut-Vertex von G

\Rightarrow Seien $u, v \in V$ zwei Kinder von s im DFS-Baum, u und v sind also in unterschiedlichen Teilbäumen unter s . Da G ungerichtet ist, enthält der DFS-Baum keine Querkanten, also keine Kanten zwischen verschiedenen Teilbäumen. Insbesondere ist (u, s, v) der einzige Pfad zwischen u und v in G . Durch das Entfernen von v wird dieser Pfad gelöscht, d.h. G zerfällt in mindestens zwei Zusammenhangskomponenten. Also ist s ein Cut-Vertex.

\Leftarrow Wenn s keine Kinder enthält, besteht der Graph nur aus der Wurzel s und damit kann s kein Cut-Vertex sein. Wenn s genau ein Kind u hat, können wir s aus dem DFS-Baum löschen und der Baum bleibt zusammenhängend. Also ist auch G nach Löschen von s noch zusammenhängend.

Das heißt, wenn s ein Cut-Vertex von G ist, dann hat s mindestens zwei Kinder im DFS-Baum.

b) Sei $v \in V$ ein Knoten, der nicht die Wurzel des DFS-Baumes ist. Wir zeigen:
Es gibt ein Kind u von v so, dass es keine Rückkante aus dem Teilbaum unter u zu einem Vorfahren von v gibt, genau dann, wenn v ein Cut-Vertex ist.

\Rightarrow Wir verwenden, dass es in einem DFS-Baum eines ungerichteten Graphs keine Querkanten gibt. Wenn es keine Rückkante aus dem Teilbaum unter u zu einem Vorfahren von v gibt, so ist der Pfad von einem beliebigen Vorfahren w von v zu u und jedem beliebigen Nachfahren von u also eindeutig. Wenn v gelöscht wird, ist u also nicht mehr von w aus erreichbar. Damit ist v ein Cut-Vertex.

\Leftarrow Wir beweisen diese Richtung mit Kontraposition. Wir setzen voraus, dass jedes Kind u von v eine Rückkante im Teilbaum unter sich zu einem Vorfahren von v hat. Sei $u \in V$ ein beliebiger Kind-Knoten von v . Dann existiert ein Pfad von u zu einem Vorfahren w von v , der nicht v enthält. Entfernen wir v , bleibt dieser Pfad also erhalten. Damit bleiben alle Knoten im Teilbaum unter u weiterhin von den Vorfahren von v im DFS-Baum erreichbar. Somit gibt es keine zwei Knoten, die sich durch das Entfernen von v nicht mehr erreichen können. Damit ist v kein Cut-Vertex.

c) Knoten v ist genau dann Cut-Vertex, wenn es ein Kind u von v mit $\text{low}(u) \geq \text{dfs}(v)$ gibt.

Sei u ein Kind-Knoten von v . Wir zeigen zunächst, dass $\text{low}(u) < \text{dfs}(v)$ genau dann gilt, wenn es im Teilbaum unter u eine Rückkante zu einem Vorfahren w von v gibt. Gibt es eine solche Rückkante, kann auch u diesen Vorfahren w erreichen, wodurch $\text{low}(u) \leq \text{dfs}(w) < \text{dfs}(v)$ gilt. Gibt es umgekehrt keine solche Rückkante im Teilbaum unter u , kann u keinen Knoten erreichen, der eine kleinere DFS-Nummer als v hat, da es keine Querkanten gibt. Also gilt in diesem Fall $\text{low}(u) \geq \text{dfs}(v)$.

Mithilfe der Aussage aus Teilaufgabe b) ist damit die Behauptung gezeigt.

d) Wir führen eine Tiefensuche auf G aus und merken uns für jeden Knoten seine DFS-Nummer, seinen low-Wert und seinen Vorfahre im DFS-Baum. Außerdem

bestimmen wir für jeden Knoten v die Menge $\text{children}(v)$ der Kindknoten von v . Hat die Wurzel s mehr als ein Kind (also $|\text{children}(s)| > 1$), so ist s ein Cut-Vertex. Für jeden weiteren Knoten v überprüfen wir, ob er einen Kind-Knoten u hat, sodass $\text{low}(u) \geq \text{dfs}(v)$.

Wir haben in Teilaufgabe a) und b) bewiesen, dass diese Bedingung äquivalent dazu ist, dass v ein Cut-Vertex ist. Damit findet unser Algorithmus alle Cut-Vertices in G .

Die Tiefensuche iteriert ein Mal über jeden Knoten und jede Kante, d.h. die dafür benötigte Zeit liegt in $\Theta(n + m)$. Wir können dabei sowohl die DFS-Nummern als auch die Vorgänger und low-Werte aller Knoten on-the-fly berechnen. Das Bestimmen der Kind-Knoten für jeden Knoten benötigt Zeit in $\Theta(n)$. Anschließend wird für die Wurzel des DFS-Baumes in konstanter Zeit und für jeden anderen Knoten v in Zeit $\Theta(\text{children}(v))$ eine Bedingung überprüft. Da jeder Knoten nur Kind maximal eines Knoten im DFS-Baum sein kann, benötigt das Überprüfen insgesamt Zeit in $\Theta(n)$. Damit liegt der Gesamtzeitbedarf in $\Theta(n + m)$.

Aufgabe 3 - Apes Components together strong! (5 Punkte)

Diese Aufgabe baute leider auf Übung 6 auf. Es gibt daher 5 Punkte geschenkt.

Aufgabe 4 - Grüner Zusammenhang (0 Punkte + 7 Bonus)

Wir betrachten einen (ungerichteten) Graphen $G = (V, E)$ mit n Knoten und m Kanten, dessen Knoten durch die Zahlen $0, \dots, n - 1$ repräsentiert sind. Wir gehen im folgenden davon aus, dass jede Kante entweder *schwarz* oder *grün* gefärbt ist. Darauf aufbauend sagen wir, dass ein Knoten v einen Knoten w *grün erreicht*, wenn es einen Pfad von v nach w gibt, der nur grüne Kanten benutzt.

- a) Zeige, dass grüne Erreichbarkeit eine Äquivalenzrelation auf den Knoten des Graphen darstellt. (1 Punkt)

Wir bezeichnen die Knotenmengen, die äquivalent bezüglich grüner Erreichbarkeit sind als grüne Zusammenhangskomponenten (GRZ). Wir definieren den GRZ-Graphen G_{GRZ} von G wie folgt: für jede grüne Zusammenhangskomponente C in G enthält G_{GRZ} einen Knoten v_C und die Knoten v_C und v_D zweier GRZ C und D sind genau dann mit einer Kante (v_C, v_D) verbunden, wenn es Knoten $c \in C$ und $d \in D$ gibt, die mit einer Kante $\{c, d\} \in E$ verbunden sind.

Im folgenden ist ein Algorithmus gesucht, der G_{GRZ} berechnet. Hierfür nehmen wir an, dass die Eingabe aus einer Adjazenzliste von G sowie einer Liste `greenEdges`: $\langle \mathbb{N}_0 \times \mathbb{N}_0 \rangle$ aller grüner Kanten besteht.

- b) Gib **Pseudocode** an, der für einen gegebenen Graphen G einen GRZ-Graphen G_{GRZ} in $\mathcal{O}(n + m)$ Zeit konstruiert. Sowohl der Eingabegraph als auch die Ausgabe sollen als Adjazenzliste (siehe Vorlesung 8, Folie 10) gespeichert sein. Beachte

hierbei insbesondere, dass die Adjazenzliste keine Kante mehrfach enthalten soll. Die Zuordnung zwischen GRZ in G und Knoten von G_{GRZ} ist dabei nicht wichtig. Nutze die Signatur $\text{GRZGRAPH}(G: \text{Graph}, \text{greenEdges}: \langle \mathbb{N}_0 \times \mathbb{N}_0 \rangle): [\text{List}(\mathbb{N}_0); k]$. Die Rückgabe ist hierbei die Adjazenzliste des GRZ-Graphen mit k grünen Zusammenhangskomponenten. (4 Punkte)

Hinweis: *Wie immer darfst du bekannte Algorithmen einfach als Subroutine verwenden. Eventuell erinnerst du dich an Radixsort aus Vorlesung 6?*

- c) Begründe kurz die Korrektheit deines Pseudocodes. (1 Punkt)
- d) Begründe, warum dein Pseudocode die Laufzeitanforderung erfüllt. (1 Punkt)

Lösung 4

a) Wir zeigen die drei benötigten Eigenschaften:

- **Reflexivität:** Grüne Erreichbarkeit ist reflexiv, da wir für jeden Knoten v den Pfad $P = (v)$ zu sich selbst finden können. Der Pfad enthält keine Kante, somit sind alle Kanten dieses Pfades grün.
- **Symmetrie:** Für zwei Knoten v und u nehmen wir an, dass es einen grünen Pfad $P = (u, v_1, \dots, v_k, v)$ von u zu v gibt. Wir müssen zeigen, dass es auch einen grünen Pfad von v zu u gibt. Hierfür nutzen wir den Pfad $P' = (v, v_k, \dots, v_1, u)$, der genau der Umkehrung von P entspricht. Alle Kanten des Pfades sind grün, da wir die selben Kanten wie der grüne Pfad P nutzen. Auch handelt es sich bei P' um einen Pfad von v zu u , da die Kanten aus P ungerichtet, also in beide Richtungen nutzbar sind.
- **Transitivität:** Für Knoten u, v und w nehmen wir nun an, dass es grüne Pfade $P_1 = (u, v_1, \dots, v_k, v)$ von u zu v und $P_2 = (v, w_1, \dots, w_l, w)$ von v zu w gibt. Wir geben einen grünen Pfad von u zu w an. Dafür folgen wir zuerst P_1 von u zu v und dann P_2 von v zu w . Somit erhalten wir $P' = (u, v_1, \dots, v_k, v, w_1, \dots, w_l, w)$. Da die Teilpfade grün sind, ist auch der Gesamtpfad grün.

b) Wir geben den folgenden Pseudocode und seine Hilfsfunktionen an:

```

1: GRZGRAPH( $G$ : Graph, greenEdges: List( $\mathbb{N}_0, \mathbb{N}_0$ )): [List( $\mathbb{N}_0$ );  $k$ ]
2:   grzNum : [ $\mathbb{N}_0 \cup \{-1\}$ ;  $n$ ] =  $[-1, \dots, -1]$ 
3:   greenG: Graph  $\leftarrow$  ADJLIST( $n$ , greenEdges)           // Kopie von  $G$  ohne
   schwarze Kanten
4:    $k$  :  $\mathbb{N}_0 = 0$ 
5:   for  $s \in \{0, \dots, n-1\}$  do
6:     if grzNum[ $s$ ] =  $-1$  then
7:       | BFS(greenG,  $s$ , grzNum,  $k$ )           // Finde alle von  $s$  erreichbaren
   Knoten und „färbe“ sie in grzNum mit Wert  $k$ ; Umsetzung wie in VL 08
8:       |  $k \leftarrow k + 1$ 
9:     end
10:  end
11:  grzEdges: List( $\mathbb{N}_0, \mathbb{N}_0$ )
12:  for  $\{u\} \in V(G)$  do
13:    for  $\{v\} \in N(u)$  do
14:      | if grzNum[ $u$ ]  $\neq$  grzNum[ $v$ ] then
15:        | grzEdges.pushBack( $\{\text{grzNum}[u], \text{grzNum}[v]\}$ ) // Finde alle
   Kanten zwischen GRZs
16:        | end
17:      end
18:    end
19:  grzEdges  $\leftarrow$  RADIXSORT(grzEdges) // sortiere grzEdges nach erstem und
   anschließend zweitem Element
20:  grzEdges  $\leftarrow$  REMOVEDUPLICATE(grzEdges) // entferne Duplikate aus
   sortierter Liste
21:  result  $\leftarrow$  ADJLIST( $k$ , grzEdges) // erstelle Adjazenzliste mit  $k$  Knoten
   und Kantenmenge grzEdges
22:  return result

```

Hier wird auf folgenden Hilfsfunktionen zurückgegriffen:

```

1: REMOVEDUPLICATE(input: List): List
2:   result = List
3:   prev =  $\perp$ 
4:   for elem  $\in$  input do
5:     if elem  $\neq$  prev then
6:       | result.pushback(elem)
7:     end
8:     prev  $\leftarrow$  elem
9:   end
10:  return result

```

```

1: ADJLIST( $n: \mathbb{N}_0$ , edges: List( $\mathbb{N}_0, \mathbb{N}_0$ )): [List( $\mathbb{N}_0$ );  $n$ ]
2:   result : [List( $\mathbb{N}_0$ );  $n$ ] = [ $\langle \rangle, \dots, \langle \rangle$ ]
3:   for  $\langle v, w \rangle \in$  edges do
4:     | result[ $v$ ].pushBack( $w$ )
5:   end
6:   return result

```

- c) Wir argumentieren, dass der erstellte Graph ein GRZ-Graph von G ist. Die Subroutine ADJLIST erstellt eine Adjazenzliste eines Graphen mit gegebener Knotenanzahl und Kantenmenge. Wir müssen also zeigen, dass k und grzEdges korrekt bestimmt wurden. Mithilfe von BFS ab s werden alle Knoten die über grüne Kanten von s erreicht werden können bestimmt und die Aktuelle nummer der GRZ wird in grzNum vermerkt. Somit sind die Repräsentanten ab 0 aufeinanderfolgend durchnummeriert und die letzte Nummer eines GRZ entspricht der Anzahl der GRZs minus 1.

Anschließend werden für alle Kanten $\{u, v\} \in E$, die verschiedene GRZs verbinden, ein Tupel aus dem Repräsentanten von u und dem von v in grzEdges eingefügt. Nach dem Entfernen doppelt eingefügter Kanten-Tupel enthält grzEdges also genau eine Kante pro Paar von GRZ, die über eine Kante verbunden sind und die Zahlen in diesen Tupeln entsprechen genau den Repräsentanten dieser GRZs.

Es folgt, dass der angegebene Algorithmus einen GRZ-Graphen von G berechnet.

- d) Das Erstellen des Hilfsgraphen der alle grünen Kanten von G enthält dauert $\mathcal{O}(n + m)$ Zeit. Das Finden der GRZs erfolgt auf diesem Graphen dann durch BFS ebenfalls in $\mathcal{O}(n + m)$ Zeit. Anschließend wird in $\mathcal{O}(n)$ Zeit über das Array der Repräsentanten iteriert um die Anzahl an GZR zu bestimmen. In Zeilen 10 bis 14 wird dann in $\mathcal{O}(m)$ Zeit über alle grünen Kanten von G iteriert und Kanten zwischen verschiedenen GZR werden in jeweils $\mathcal{O}(1)$ Zeit in eine Liste eingefügt. Diese Liste kann im Anschluss in $\mathcal{O}(n + m)$ Zeit mit Radix-Sort sortiert werden, da sie höchstens m Tupel mit jeweils zwei Zahlen kleiner gleich n enthält. Das Entfernen von Duplikaten erfolgt dann in $\mathcal{O}(m)$ Zeit durch einmaliges Iterieren über die sortierte Liste von Kanten. Im Anschluss kann die gewünschte Adjazenzliste in $\mathcal{O}(n + m)$ Zeit erstellt werden.

Insgesamt ergibt sich die geforderte Laufzeit von $\mathcal{O}(n + m)$.

Aufgabe 5 - Zum Knobeln (0 Punkte + 130 Bonus)

Um zu verhindern, dass du George Dantzig nachmachst³, gleich vorab: Das hier vorgestellte Problem ist bislang noch ungelöst.

Ein Magisches Quadrat der Größe 3×3 ist ein Gitter mit drei Zeilen und drei Spalten. In diesem Magischen Quadrat summiert sich jede Zeile, jede Spalte und jede Diagonale auf die selbe Zahl. Wir sind nun an einem Magischen Quadrat interessiert, dass

³https://de.wikipedia.org/wiki/George_Dantzig#Leistungen

nur Quadratzahlen (also Quadrate von natürlichen Zahlen) enthält. Außerdem sollen die verwendeten Quadratzahlen paarweise verschieden sein. Gib ein solches magisches Quadrat an. (130 Punkte)

Lösung 5



Wir geben daher eine bekannte Nichtlösung an. Das Parker Square hat die folgende

Form.
$$\begin{pmatrix} 29^2 & 1^2 & 47^2 \\ 41^2 & 37^2 & 1^2 \\ 23^2 & 41^2 & 29^2 \end{pmatrix}$$