Aufgabe 2: Spießgesellen

Teilnahme-Id: 55628

Bearbeiter dieser Aufgabe:
Michal Boron

April 2021

Inhaltsverzeichnis

1 Lösungsidee **1**
- 1.1 Formulierung des Problems . . . 1
- 1.2 Bipartiter Graph . . . 1
- 1.3 Logik . . . 2
- 1.4 Komponenten . . . 3
- 1.5 Prüfung auf Korrektheit der Eingabe . . . 4
- 1.6 Laufzeit . . . 4

2 Umsetzung **4**

3 Beispiele **4**
- 3.1 Beispiel 0 (Aufgabenstellung) . . . 4
- 3.2 Beispiel 1 (BWINF) . . . 4
- 3.3 Beispiel 2 (BWINF) . . . 4
- 3.4 Beispiel 3 (BWINF) . . . 4
- 3.5 Beispiel 4 (BWINF) . . . 5
- 3.6 Beispiel 5 (BWINF) . . . 5
- 3.7 Beispiel 6 (BWINF) . . . 5
- 3.8 Beispiel 7 (BWINF) . . . 5

4 Quellcode **5**

1 Lösungsidee

1.1 Formulierung des Problems

TODO: use definitions

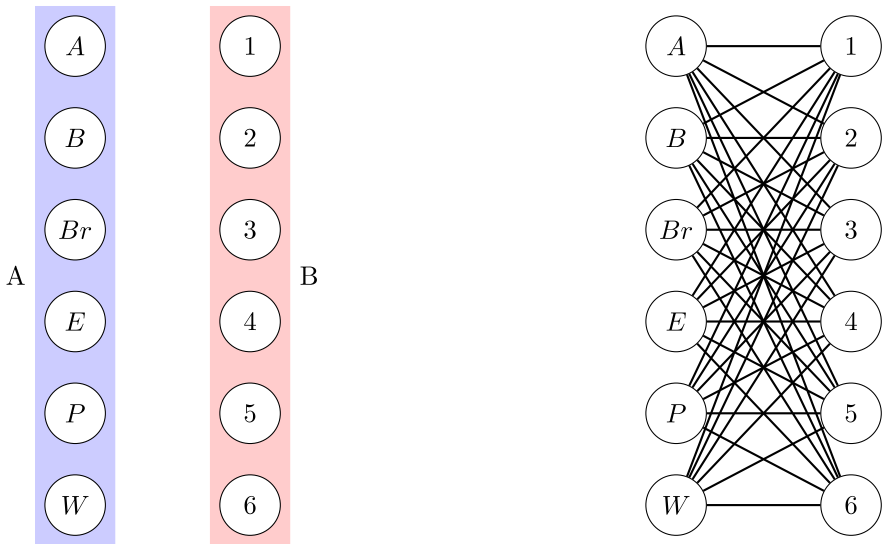
Gegeben sind eine Menge von n *Obstsorten* A und eine Menge von n ganzen Zahlen $B = \{1, 2, ..., n\}$, die für die Indizes der Obstsorten stehen. Gegeben sind auch m *Spießkombinationen*, wobei jede i–te Spießkombination aus einer Menge von Obstsorten $F_i \subseteq A$ und einer Menge der Zahlen $Z_i \subseteq B$ besteht. Für jedes i besteht die Menge Z_i nur aus den in B enthaltenen Indizes, die den Obstsorten in F_i entsprechen, deshalb haben auch die beiden Mengen F_i und Z_i dieselbe Anzahl an Elementen. Außerdem gegeben ist auch eine *Wunschliste* $W \subseteq A$.
Die Aufgabe ist, zu entscheiden, ob die Menge der Indizes der in W gegebenen Obstsorten $W' \subseteq B$ anhand der m Spießkombinationen eindeutig bestimmt werden kann. Falls ja, soll sie auch ausgegebn werden.

1.2 Bipartiter Graph

TODO: use definitions
Annahmen beschreiben

Man kann die beiden Mengen A und B zu Knoten eines bipartiten Graphen $G = (A \cup B = V, E)$ umwandeln. Die Menge der Kanten E wird im Folgenden festgelegt. Man stellt den Graphen als eine Adjazenzmatrix M der Größe $n \times n$ dar. Als M_i bezeichne ich die Liste der Länge n, die die Beziehungen des Knotens $i \in A$ zu jedem Knoten $j \in B$ als 1 (Kante) 0 (keine Kante) darstellt.
Nach der Aufgabenstellung gehört jeder Obstsorte aus A genau ein Index aus B. Dennoch man kann am Anfang keiner Obsorte einen Index zuweisen. Deshalb verbinden wir zunächst jeden Knoten aus A mit jedem Knoten aus B durch eine Kante. Am Anfang ist M dementsprechend voll mit 1–en. Bei der Erstellung der Adjazenzmatrix können wir den Vorteil nutzen, dass die jeweilige Liste von Nachbarn des jeden Knotens $x \in A$ nur aus 0–en und 1-en besteht, indem wir diese Liste als Bitmasken darstellen (mehr dazu in der Umsetzung).

Abbildung 1: Beide Abbildungen stellen den Graphen für das Beispiel aus der Aufgabenstellung dar. Die Buchstaben stehen für die entsprechenden Obstsorten aus diesem Beispiel (s. auch 3.1).



(a) Die entsprechenden Mengen des Graphen

(b) Der Graph am Anfang

Jede i–te *Spießkombinationen* bringt uns Informationen über die Obstsorten in F_i. Als $a \rightsquigarrow b$ bezeichnen wir, dass a den Index b haben kann. Wir können Folgendes festellen.

Lemma 1. *Für jede i–te Spießkombination mit $F_i \subseteq A$ und $Z_i \subseteq B$ gilt, dass $\forall x \in F_i, \forall y \in Z_i : x \rightsquigarrow y$. Es gilt gleichzeitig, dass $\nexists p \in F_i, \forall q \in B \setminus Z_i : p \rightsquigarrow q$*

TODO: Beweis

Beweis. □

Nach Lemma 1 dürfen wir alle Kanten, die aus jedem Knoten $x \in F_i$ zu einem Knoten $y \in B \setminus Z_i$ führen, aus E entfernen und nur die Kanten lassen, die zu allen $z \in Z_i$ führen. Da wir Bitmasken für die Darstellung jeder Liste M_i $(i \in A)$ verwenden, können wir die Laufzeit bei der Analyse der jeweiligen Spießkombination optimieren, weil ich für die Operation des Entfernens Logikgatter verwende.

1.3 Logik

TODO: Veranschauung

Betrachten wir, bzw. *analysieren* wir, eine Spießkombination s, die aus den Mengen $F_s \subseteq A$ und $Z_s \subseteq B$ besteht. Wir erstellen 3 Bitmasken bf, bn und br jeweils der Länge n. Die Bitmaske bf besteht aus n 1–en.In der Maske bn stehen die 1–Bits an allen Stellen, die den Indizes in Z_s entsprechen. Die Bitmaske

br wird auf folgende Weise definiert:

$$br := \neg(bn) \land bf.$$

So können wir auf allen Listen M_i, wobei $i \in F_s$, die AND–Operation mit der Maske bn durchführen:

$$M_i := M_i \land bn.$$

Analog führen wir die AND–Operation mit der Maske br auf allen Listen M_j, wobei $j \in A \setminus F_s$, durch:

$$M_j := M_j \land br.$$

Was die beschriebenen Operationen verursachen, erläutere ich anhand der folgenden Fallunterscheidung.

1. Falls es sich um einen Knoten $x \in F_s$ handelt.
   a) Falls ein Knoten y zu Z_s gehört, aber an der Stelle y in M_x 0 steht, ergibt sich laut Lemma 1 ein Widerspruch. [MB: No i co z tego?? Dopisać]
   b) Falls ein Knoten y zu Z_s gehört und an der Stelle y in M_x 1 steht, bleibt es auch 1.
   c) Falls ein Knoten y nicht zu Z_s gehört und an der Stelle y in M_x 0 steht, bleibt es auch 0.
   d) Falls ein Knoten y nicht zu Z_s gehört, aber an der Stelle y in M_x 1 steht, wird die Stelle y in M_x zu 0.
2. Falls es sich um einen Knoten $x \in A \setminus F_s$ handelt.
   a) Falls ein Knoten y nicht zu Z_s gehört, aber an der Stelle y in M_x 0 steht, ergibt sich laut Lemma 1 ein Widerspruch.
   b) Falls ein Knoten y nicht zu Z_s gehört und an der Stelle y in M_x 1 steht, bleibt es auch 1.
   c) Falls ein Knoten y zu Z_s gehört, aber an der Stelle y in M_x 1, wird die Stelle y in M_x zu 0.
   d) Falls ein Knoten y zu Z_s gehört und an der Stelle y in M_x 0 steht, bleibt es auch 0.

1.4 Komponenten

Nach der Analyse der allen m Spießkombinationen verfügen wir über einen Graphen G, in dem viele Kanten in E entfernet wurden. Auf diese Weise können wir schon anfangen, die Indizes der Obstsorten aus W festzulegen.

Definition 1 (Matching)**.** *Sei $\mathcal{G} = (V, E)$ ein ungerichtetes Graph. Als ein **Matching** bezeichnen wir eine Teilmenge $S \subseteq E$, sodass für alle $v \in V$ gilt, dass höchstens eine Kante aus S inzident zu v ist. Wir bezeichnen einen Knoten $v \in V$ als in S **gematcht**, wenn eine Kante aus S inzident zu v sit.*

Definition 2 (Perfektes Matching)**.** *Sei $\mathcal{G} = (V, E)$ ein ungerichtetes Graph. Ein **perfektes Matching** $\mathcal{M} \subseteq E$ ist so ein Matching, in dem alle Knoten aus V gematcht sind.*

Definition 3 (Nachbarschaft)**.** *Sei $\mathcal{G} = (V, E)$ ein ungerichtetes Graph. Für alle $X \subseteq V$ definieren wir die **Nachbarschaft** von X als $\mathcal{N}(X) = \{y \in V : (x, y) \in E \text{ für einige } x \in X\}$.*

Satz 1 (Satz von Hall)**.** *Sei $\mathcal{G} = (L \cup R, E)$ ein bipartites, ungerichtetes Graph. Es existiert ein perfektes Matching genau dann, wenn es für alle Teilmengen $K \subseteq L$ gilt: $|K| \leqslant |N(K)|$.*

Beweis. Auf den Beweis verzichte ich. Ein Beweis ist beispielsweise hier [1] zu finden. □

Lemma 2. *Seien $x \in A$ ein Knoten in G, seine Kardinalität $\Delta(x) = 1$, und der einzelne Nachbar von x sei $y \in B$. Dann gehört der Index y der Obstsorte x.*

Beweis. Nach dem Satz von Hall ist für $x \in A$ die Bedingung $|x| = \Delta(x) = 1 \leqslant |N(x)| = 1$ erfüllt. Deshalb existiert ein perfektes Matching für $x \in A$ und das ist auch das einzelne mögliche Matching. Nach der Aufgabenstellung hat jede Obstsorte genau einen Index, also ist der Index der Obstsorte x somit gefunden. □

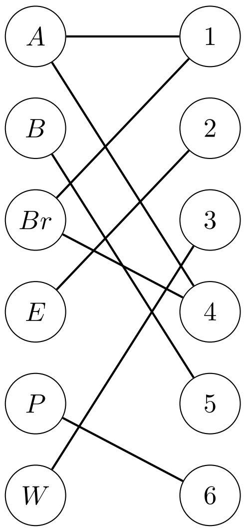
TODO: Lemma: 2. Fall $\to \Delta(x) > 1$
\+ Beweis

[1] Anup Rao. Lecture 6 Hall's Theorem. October 17, 2011. University of Washington. [Zugang 21.01.2021] `https://homes.cs.washington.edu/~anuprao/pubs/CSE599sExtremal/lecture6.pdf`

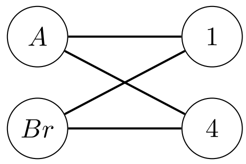
Lemma 3. *Seien $x \in A$ ein Knoten in G und seine Kardinalität $\Delta(x) = k > 1$. Dann gehört x zu einer Zusammenhangskomponente C, die aus insgesamt $2k$ Knoten $x_1, ..., x_k \in A$ und $y_1, ..., y_k \in B$ und k^2 Kanten, die das jeweilige Paar von Knoten $(x_i, y_j) \in E$ für alle $1 \leqslant i, j \leqslant k$ verbinden, besteht.*

Beweis. Der Beweis erfolgt durch Widerspruch. □

TODO: DFS beschreiben



(a) Der Graph nach der Analyse der allen Spießkombinationen



(b) Die ürbige Zusammenhangskomponente

1.5 Prüfung auf Korrektheit der Eingabe

1.6 Laufzeit

2 Umsetzung

3 Beispiele

3.1 Beispiel 0 (Aufgabenstellung)

Textdatei: `spiesse0.txt`

Apfel, Brombeere, Weintraube

1, 3, 4

3.2 Beispiel 1 (BWINF)

Textdatei: `spiesse1.txt`

Clementine, Erdbeere, Grapefruit, Himbeere, Johannisbeere

1, 2, 4, 5, 7

3.3 Beispiel 2 (BWINF)

Textdatei: `spiesse2.txt`

Apfel, Banane, Clementine, Himbeere, Kiwi, Litschi

1, 5, 6, 7, 10, 11

3.4 Beispiel 3 (BWINF)

Textdatei: `spiesse3.txt`

Clementine, Erdbeere, Feige, Himbeere, Ingwer, Kiwi, Litschi

unlösbar: Litschi gehört zur Komponente mit Grapefruit. Dabei ist Grapefruit kein Wunsch.

3.5 Beispiel 4 (BWINF)

Textdatei: `spiesse4.txt`

Apfel, Feige, Grapefruit, Ingwer, Kiwi, Nektarine, Orange, Pflaume

2, 6, 7, 8, 9, 12, 13, 14

3.6 Beispiel 5 (BWINF)

Textdatei: `spiesse5.txt`

Apfel, Banane, Clementine, Dattel, Grapefruit, Himbeere, Mango, Nektarine, Orange, Pflaume, Quitte, Sauerkirsche, Tamarinde

1, 2, 3, 4, 5, 6, 9, 10, 12, 14, 16, 19, 20

3.7 Beispiel 6 (BWINF)

Textdatei: `spiesse6.txt`

Clementine, Erdbeere, Himbeere, Orange, Quitte, Rosine, Ugli, Vogelbeere

4, 6, 7, 10, 11, 15, 18, 20

3.8 Beispiel 7 (BWINF)

Textdatei: `spiesse7.txt`

Apfel, Clementine, Dattel, Grapefruit, Mango, Sauerkirsche, Tamarinde, Ugli, Vogelbeere, Xenia, Yuzu, Zitrone

unlösbar: Apfel, Grapefruit und Xenia gehören zur Komponente mit Litschi. Dabei ist Litschi kein Wunsch. Ugli gehört zur Komponente mit Banane. Dabei ist Banane kein Wunsch.

4 Quellcode

./tex/spiesse.m