Aufgabe 2: Spießgesellen

Teilnahme-ID: 59521

Bearbeiter/-in dieser Aufgabe:
Christoph Waffler

11. April 2021

Inhaltsverzeichnis

Lösungsidee

Anmerkung: Mit den Begriffen „Obstschale“, „Schale“, „Schüsseln“ und „Obstschüsseln“ sind die „Schüsseln“ der Aufgabestellung gemeint, in denen sich das Obst von Daisys Party befindet.

Lösung für Teilaufgabe a

Hier soll zuerst die Lösung für das Problem der Einstiegsaufgabe skizziert werden, d.h. die Frage soll beantwortet werden, aus welchen Schüsseln sich Donald bedienen soll.

Hierfür lassen sich seine Beobachtungen erst einmal formal darstellen, wobei die Obstsorten mit Abkürzungen und die Schüsseln mit den entsprechenden Nummern versehen werden:

Apfel = A; Banane = BA; Brombeere = BR; Erdbeere = E; Pflaume = P; Weintraube = W;

Hinweis: Es wird z.B. die Menge A an Obstsorten mit der Menge B an Obstschüsseln gleichgesetzt, wenn die Menge A in Kombination mit der Menge B von Donald beobachtet wurde.

1. Beobachtung: $\{A; BA;\ BR\} = \{1; 4; 5\}$
2. Beobachtung: $\{B; P; W\} = \{3; 5; 6\}$
3. Beobachtung: $\{A; BR; E\} = \{1; 2; 4\}$

Somit lässt mit dem Ausschlussverfahren und einigen Kombinationen (Schnittmengen bestimmen) folgendes feststellen:

$$BA = \{1; 4; 5\} \cap \{3; 5; 6\} = \{5\}$$

$$\{A; BR\} = \{1; 4; 5\} \cap \{1; 2; 4\} = \{1; 4\}$$

$$\{P; W\} = \{3; 5; 6\}/\{BA\} = \{3; 5; 6\}/\{5\} = \{3; 6\}$$

$$\{E\} = \{1; 2; 4\}/\{A; BR\} = \{1; 2; 4\}/\{1; 4\} = \{2\}$$

Da Donald eine eindeutige Bestimmung der Obstsorten A; BR; W fordert wird noch eine weitere Beobachtung benötigt, da *W* in den Schüsseln 3 und 6; *A* in den Schüsseln 1 und 4; und *BR* auch in den Schüsseln 1 und 4 sein kann.

4. Beobachtung: $\{E; P\} = \{2; 6\}$

Durch die 4. Beobachtung lässt sich folgendes erschließen:

$$\{P\} = \{3; 6\} \cap \{2; 6\} = \{6\}$$

$$\{W\} = \{3; 6\}/\{P\} = \{3; 6\}/\{6\} = \{3\}$$

Somit ist die Obstsorte Weinbeere sicher in der Schüssel Nr. 3.
Die Sorten Apfel und Brombeere konnten hingegen nicht eindeutig bestimmt werden, da beide aus den Schüsseln Nr. 1 oder 4 stammen können.
Diese Bestimmung ist aber auch nicht weiter nötig, da er beide Sorten auf seinem Obstspieß haben möchte und es daher egal ist, in welcher Schüssel die Sorte Apfel und Brombeere exakt sind.

Somit muss Donald sich für seinen gewünschten Obstspieß aus den Schüsseln Nr. 1, 3 und 4 bedienen.

Lösung für Teilaufgabe b

Da das oben verwendete Verfahren zu aufwendig und kompliziert umzusetzen ist, wird ein anderes Verfahren verwendet, das deutlich einfach ist, dafür etwas abstrakter ist.

Im Folgenden werden nun die einzelnen Schritte des Algorithmus anhand des Beispiels aus Teilaufgabe a skizziert, welche später in der Umsetzung genauer erklärt werden.

Hinweis mit „Wünsche“ ist die Menge der Obstsorten gemeint, die Donald gerne auf seinen Obstspieß hätte.

1. Erstellen eines bipartiten Graphen *G* aus der Menge der Obstsorten und der Menge der Obstschüsseln:
   Für jede eingelesene Beobachtung von Donald werden die Kantengewichte aktualisiert. So ist z.B. Menge $Obstsorten_1 = \{A; BA; BR\}$ wird zusammen mit der Menge $Schüsseln_1 = \{1; 4; 5\}$.
   Nun wird für jedes $x \in Obstsorten_1$ die Gewichtung in *G* zu allen $y \in Schüsseln_1$ um *1* erhöht.
   Nach dem ersten Durchlauf des eben genannten kann der bipartite Graph G nun wie folgt dargestellt werden:
   Im Graph G befinden sich nun zwei Knotenmengen: Zum einen die Menge der Obstsorten (dargestellt durch die Abkürzung in Abb. 1) und die Menge der Schüsseln (dargestellt in durch die entsprechenden Nummern in Abb. 1).

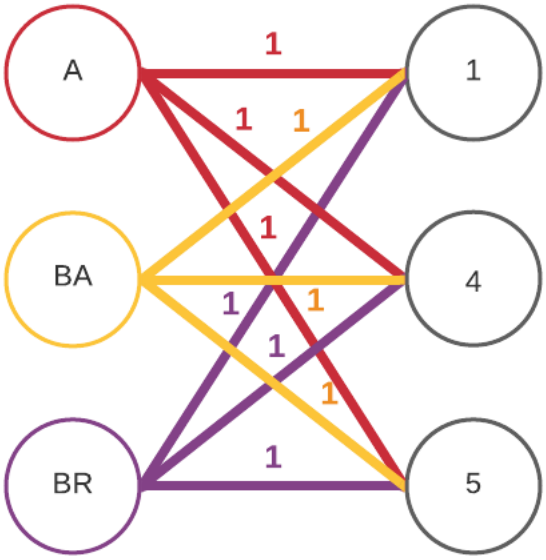

Abb. 1

2. Da kein Element der Menge der Obstsorten eine Beziehung (Kante) zu einem anderen Element derselben Menge hat (dasselbe gilt auch für die Schüsseln), ist der Graph G der aus beiden Mengen gebildet werden kann, bipartit, sprich 2-färrbar.

   Während jede Beobachtung von Donald eingelesen wird, wird nach dem oben erwähnten Schema, zwischen jeder beobachteten Obstsorte eine Kante in G zu allen mit dieser Obstsorte beobachteten Schüssel erzeugt.
   Existiert bereits eine Kante zwischen der Sorte und Schüssel, so wird das Kantengewicht dieser Kante um *1* erhöht.

   Nach den weiteren Beobachtungen
   $\{BA; P; W\}\ mit\ \{3; 5; 6\};\ \{A; BR; E\}\ mit\ \{1; 2; 4\}; \{E; P\}\ mit\ \{2; 6\}$ ergibt sich folgender Graph G:

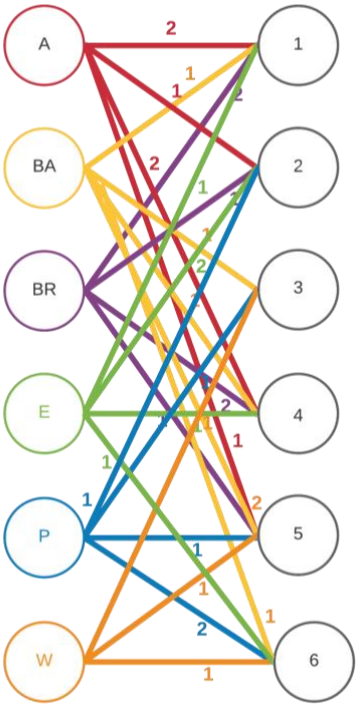

Abb. 2

3. Für jede Obstsorte a wird als erstes das höchste Kantengewicht g_a ermittelt, das eine Kante zwischen a und b, $b \in Schüsseln$ besitzt.
   Nun werden für jede Obstsorte a nur noch jene Kanten in G behalten, die das Kantengewicht g_a besitzen, d.h. alle Kanten mit einem geringen Gewicht werden entfernt.
   Daraus reduziert sich der Graph aus Abb. 2 auf folgenden:

Abb. 3

4. Nun wird Schritt 2 für die Schüsseln ausgeführt:
   Für jede Obstschüssel b wird als erstes das höchste Kantengewicht g_b ermittelt, das eine Kante zwischen b und a, $a \in Obstsorten$ besitzt.
   Nun werden für jede Schüssel b nur noch jede Kanten in G behalten, die das Kantengewicht g_b besitzen, d.h. alle Kanten mit einem geringerem Gewicht werden ebenfalls entfernt.
   Daraus reduziert sich der Graph aus Abb. 3 auf folgenden:

5. Eine Kante k zwischen a und b steht also für eine Möglichkeit, dass die Sorte a aus der Schüssel b stammen kann, falls das Kantengewicht g_k der Kante k gleich dem Maximalgewicht g_a der Obstsorte a (siehe 2. Schritt) und gleich dem Maximalgewicht g_b der Schüssel b entspricht (vgl. 3. Schritt).

   Somit steht folgendes für das Beispiel (siehe Abb. 4) fest:
   Sorte A und BR in den Schüsseln 1 und 4; BA sicher in 5; E sicher in 2; P sicher in Schüssel 6 und W sicher in 3.

6. Nun wird überprüft, ob Donalds „Wunschspieß“ eindeutig bestimmt werden kann:
   Folgende Sorten sollen auf seinem Spieß sein: $\{A; BR; W\}$.
   Die Sorte W ist sicher in Schüssel Nr. 3.
   Bei der Bestimmung der Sorte A bzw. der Sorte BR ist das nicht ganz so eindeutig.
   Daher wird nun für die Sorte A überprüft, in welchen Schüsseln diese sein kann, und welche anderen Sorten auch dort sein können; diese anderen Sorten, werden als $konkurrierendeMenge_A$ der Sorte A bezeichnet.
   Für jede Sorte der $konkurriendeMenge_A$ wird nun überprüft, ob diese Sorte auch in Donalds Wunschsorten enthalten ist.

Abb. 4

Denn falls dies nicht der Fall ist, kann es nämlich vorkommen, dass Donald eine Sorte auswählt, die er nicht haben möchte.

Dasselbe wird auch für alle andere Sorten ausgeführt, die nicht ganz eindeutig bestimmt werden können.
In dem Beispiel der Einstiegsaufgabe ist dies noch die Sorte BR, wobei dann die $konkurrierendeMenge_{BR}$ erstellt wird, und jedes Element aus der $konkurrierendeMenge_{BR}$ überprüft wird, ob es in Donalds Wunschsorten enthalten ist.

Abstrakt zusammengefasst wird hier im 5. Schritt folgendes getan, wobei alle im Folgenden erwähnten Kanten aus dem reduzierten Graph *G* nach Schritt 3 betrachtet werden:
Für jeden Wunsch *w*, $w \in Wünsche$, wird folgendes überprüft:
- Ist die Bestimmung von *w* eindeutig, d.h. im Graph *G* existiert nur noch eine Kante *k*, die *w* mit einer Schüssel *b* verbindet, wobei für das Gewicht g_k von *k* folgendes gilt:

$$g_k = g_w(maximal\ Gewicht\ von\ w) = g_b\ (maximal\ Gewicht\ von\ b)$$

− Existiert hingegen mehrere Kanten *K*, die die Sorte *w* mit einer Schüssel verbinden, wird folgendes überprüft:
Für alle Kanten *k*, $k \in K$, werden die möglichen Obstsorten *B* ermittelt, wobei gilt dass alle *b*, $b \in B$, durch eine Kante aus *K* mit *w* verbunden sind.
Für alle Obstsorten *b*, $b \in B$, werden nun alle anderen Obstsorten *A* ermittelt, wobei *w* nicht in *A* enthalten ist.
Für alle *a,* $a \in A$ gibt es eine Verbindung zu einer Obstsorte *b*, $b \in B$.
Für alle *a,* $a \in A$ muss gelten: $a \in Wünsche$, da ansonsten der Spieß nicht eindeutig bestimmt werden kann. So können die Wunschsorten nicht sicher den entsprechenden Wunschschalen zugewiesen werden.
Gilt $a \in Wünsche$, so kann der Obstspieß mit der Menge *Wünsche* immer noch eindeutig bestimmt werden, da die Menge der Schüsseln eindeutig bestimmt werden kann. Nur *w* selbst kann nicht eindeutig bestimmt werden, muss es aber auch nicht.

7. Kann in Schritt 5 die Menge *Wünsche* eindeutig bestimmt werden, so werden die Schüsseln des entsprechenden Obstspießes ausgegeben.
   Ist dies nicht der Fall, so werden diejenigen Sorten von *Wünsche* und deren entsprechenden „Konkurrenten“ ausgegeben, d.h. die „Konkurrenten“ sind die Sorten, die auch aus den Schüsseln stammen können, aber nicht in der Menge *Wünsche* enthalten sind.

Erweiterung:

Da Daisys Gäste einen riesen Hunger haben und sie weiß, dass Donald keine Überraschungen liebt, stellt sie mehrere Schalen mit den gleichen Obstsorten auf. Somit befindet sich in einer Schale genau eine Obstsorte, aber eine Obstsorte befindet sich nicht nur in einer Schale.

Doch Donald ist davon gar nicht begeistert und hat mich gebeten, ihm auch hierfür zu helfen.
Die oben beschriebene Lösungsidee ist bereits in der Lage einer Obstsorte mehrere Schüsseln zu

zuweisen.
Es kann jedoch vorkommen, dass nur jene Obstschale ausgegeben wird, die am häufigsten in Kombination mit der Obstsorte beobachtet wird.

Umsetzung

Die Lösungsidee wurde in C++ unter MacOS 11.2.3 umgesetzt.

Falls das Programm nicht ausführbar ist, einfach mit „ g++ -std=c++11“ kompilieren.

1. Einlesen der Eingabe und Erstellen eines bipartiten Graphen aus den Obstsorten und Schüsseln:
   Da alle Obstsorten als String eingegeben werden, werden diese in einer *unordered_map<string, int> words2num* und *unordered_map<int, string> num2words* gespeichert, um die Obstsorte (*string*) zu einem *int* zu konvertieren und umgekehrt.

   Die Methode *add_word_to_converter(string word)* fügt die übergebene Obstsorte *word* zu beiden Hashtabellen hinzu, falls *word* noch nicht hinzugefügt wurde.
   Dabei wird jedem neu hinzugefügten Word eine Nummer zugewiesen, die sich pro neues Wort um 1 erhöht.

   Die Hilfsmethoden *getline2vector_string(string input_line)* und *getline2vector_int(string input_line)* konvertieren die eingegeben Zeile zu einem *vector* mit *strings* bzw. mit *ints*.

   Als erstes werden alle Wünsche von Donald eingelesen und mithilfe von *getline2vector_string* in einem *vector<string> wünsche* gespeichert und mit *add_word_to_converter* zu beiden Hashtabellen hinzugefügt.
   Ebenfalls werden die Wünsche in einem *vector<bool> isWunsch* auf *true* gesetzt, da *isWunsch* später in Schritt 4 der Umsetzung noch benötigt wird.

   Anschließend wird eine neue Adjazenzmatrix mit der Dimension *MAXN x MAXN* erstellt, wobei *MAXN* die Anzahl der maximalen Wörter +1 ist.
   Die Adjazenzmatrix heißt *adj* und ist vom Typ *vector<vector<int>>*.

   Nun werden alle Beobachtungen eingelesen:
   Zuerst werden die Obstschalen eingelesen und die dabei beobachteten Obstsorten.
   Die Obstsorten werden ebenfalls zum beiden Hashtabellen hinzugefügt.

   Anschließend wird für jede Sorte *sorte* (vorherig konvertiert zu einem *int* der entsprechenden Nummer der Sorte) folgendes ausgeführt:
   Iteriere über jede *schale* der Obstschalen und erhöhe dabei die Kante in der Adjazenzmatrix *adj* um 1, falls bereits eine Kante existiert.
   Andernfalls erstelle eine neue Kante und setzte deren Gewicht auf 1.
   Dies wird umgesetzt, indem die erste Dimension von *adj* die jeweiligen Sorten und die 2. Dimension von *adj* die entsprechenden Nummern der Schalen sind.
   Falls *adj[sorte][schale]* nun kleiner als *0* ist, wird *adj[sorte][schale]* auf 0 gesetzt.
   Unabhängig davon wird *adj[sorte][schale]* um eins erhöht.

2. Nun wird für jede Obstsorte *i* das maximale Kantengewicht ermittelt.
   Hierfür wird über jede Obstschale *j* iteriert und der maximale Wert $maxV_i$,
   mit $maxV_i = \max\{maxV_i, adj[i][j]\}$ ermittelt.
   Anschließend wird $maxV_i$ im *vector<int> maxVObstsorten[i]* gespeichert.

3. Schritt 4 wird analog auch für jede Obstschale *j* durchgeführt, wobei dafür über jede Obstsorte *i* iteriert wird und $maxV_j = \max\{maxV_j, adj[i][j]\}$ ist.
   Nun wird $maxV_j$ im *vector<int> maxVObstschalen[j]* gespeichert.

4. Nun werden Schritt 2, 3 und 4 der Lösungsidee in einem Schritt kombiniert ausgeführt:
   Als erstes werden zwei *vector<set<int>> sorte2schale* und *schale2sorte* initialisiert, in denen sich alle möglichen Schalen einer Sorte bzw. alle möglichen Sorten einer Schale befinden werden. Hier werden *sets* verwendet, damit nicht z.B. eine Schale doppelt bei einer Sorte auftritt, da in einem *set* kein Element doppelt vorkommen kann.

   Für jede *sorte* aller Obstsorten wird über jede *schale* aller Schalen iteriert:
   Falls die 2 Bedingungen
   $$adj[sorte][schale] = maxVObstsorten[sorte]$$
   $$adj[sorte][schale] = maxVSchalen[schale]$$
   erfüllt sind, kann sich die Sorte *sorte* unter den gegebenen Beobachtungen in der Schale *schale* befinden.
   Daher wird zu *sorte2schale[sorte] schale* und *schale2sorte[schale] sorte* hinzugefügt.

   Wurde diese doppelte Iteration ausgeführt, so wird der „reduzierte Graph *G*“ aus der Lösungsidee nun durch die beiden ‚Adjazenzlisten‘ *sorte2schale* und *schale2sorte* dargestellt.

5. Als nächstes wird die Zuordnung ausgegeben und gleichzeitig überprüft, ob sie eindeutig ist.
   Die boolesche Variable *works* wird zu Beginn auf *true* gesetzt. Falls es sich im Folgenden herausstellt, dass die Zuweisung nicht eindeutig ist, wird *works* auf *false* gesetzt.

   Für jeden Wunsch *w* wird folgendes ausgeführt:
   Falls die Größe des *sets sorte2schale[w]* gleich 1 ist, so ist *w* sicher und eindeutig in der Schale mit der Nummer, die das *set sorte2schale[w]* enthält.

   Falls jedoch das *set* leer ist, kann *w* keiner Schale zu geordnet werden.

   Ansonsten gibt es noch die Bedingung, dass das *set* größer als 1 ist. Das heißt, dass sich die Sorte *w* in mehreren Schüsseln befinden kann.
   Jede mögliche Schüssel wird ausgegeben und alle anderen Sorten, die sich auch darin befinden können.
   Für jede andere Sorte, die sich darin befinden können, wird mithilfe des *vector<bool> isWunsch* überprüft, ob diese eine Wunschsorte ist.

Ist dies nicht der Fall, so wird *works* auf *false* gesetzt, da die Zuweisung der Wunschsorten nicht sicher auf Wunschschalen fällt.

6. Ganz am Ende des Programms wird noch ausgegeben, ob jetzt Donalds Wunschsorten sicher bestimmt werden können oder nicht (abhängig von der Variable *works*).
   Können diese eindeutig bestimmt werden, so werden die Nummern der entsprechenden Schalen ausgegeben.

Erweiterung:

Falls eine Sorte in mehreren Schalen sein kann, wird dies ebenfalls ausgegeben.
In den Beispielen finden sich entsprechende Eingaben, die eine solche Situation darstellen.

Laufzeitkomplexität

Im folgenden Abschnitt wird nun die Laufzeit des Programms und der benötigte Speicher in Abhängigkeit von der Eingabe bestimmt.

N wird definiert als die Anzahl der Sorten und *M* als die Anzahl der Schalen. (Da laut eigener Erweiterung $N \neq M$ eintreten kann und sich somit der Speicherbedarf und die Laufzeit verändert).
P wird als die Anzahl der Beobachtungen definiert.

Es wird somit eine Funktion $f(N, M, P)$ gesucht, die das Laufzeitverhalten des Programms beschreibt.

Es wird vorausgesetzt, dass die Ein- und Ausgabe mit *getline()* und *cin*, und mit *cout* konstante Zeit benötigt, und wird deshalb außer Acht gelassen.

- Das Einlesen Donalds Wünschen benötigt maximal $O(N)$ Zeit, falls er alle Obstsorten auf seinem Spieß haben möchte.
  Das Überprüfen, ob eine Wunschsorte bereits in der Hashtabelle *words2num* enthalten ist, wird mit *word2num.find()* implementiert, wodurch im worst case $O(N)$ Zeit benötigt wird.

- Nun folgt das Einlesen der Beobachtungen von Donald:
  Es werden ebenfalls pro Beobachtung die Obstsorten zur Hashtabelle überprüft, ob sie schon hinzugefügt wurden, was im worst case $O(N)$ Zeit benötigt.
  Anschließend wird einer in verschachtelten foreach-Schleife über alle Sorten und alle Schalen iteriert, und dabei die Gewichtung in der Adjazenzmatrix verändert, was $O(1)$ *vector<vector<int>>* implementiert wurde.
  Beide foreach-Schleifen benötigen also $O(N * M)$ Zeit, die *P* mal ausgeführt werden.
  Somit ergibt sich eine Laufzeit von $O(P * N * M)$

- Anschließend wird der *maxVObstsorten* und der *maxVSchalen* ermittelt.
  Dabei wird in einer foreach-Schleife über alle Obstsorten und alle Schalen iteriert und anschließend dasselbe nur mit umgekehrter Verschachtelung.
  Daraus ergibt sich im worst case eine Laufzeit von $O(2 * N * M)$.

- Anschließend werden die wird *sort2schale* und *schale2sorte* erstellt.
  Hierfür wird wieder in einer verschachtelten for-Schleife über alle Sorten und alle Schalen iteriert.
  Darin werden mit konstanter Zeit die *maxV*-Werte mit den entsprechenden Kantengewichten der Adjazenzmatrix überprüft.
  Falls alle Werte gleich sind, wird die aktuelle Schale bzw. Sorte zum Set hinzugefügt, was im worst case $O(\log(N) + \log(M)$ Zeit benötigt, da es 2 Sets sind.
  Dadurch ergibt sich abschließend für diese gesamte Durchführung eine maximale Laufzeit von $O(N * M * (\log(N) + \log(M))$.

- Nun folgt die Ausgabe und die Überprüfung, ob die Zuweisung eindeutig ist.
  Hinweis: Im Set *finalSchalen* befinden sich am Ende alle Nummern der Schalen, in denen sich die Wunschsorten befinden.
  Für alle Wünsche, maximal *N*, wird folgendes ausgeführt:

- Als erstes die Größe des Sets in *sorte2schale[wunsch]* überprüft, was konstante Zeit benötigt.
- Ein Wunsch kann sich in maximal *M* Schalen befinden, welche zu *finalSchalen* hinzugefügt werden müssen (logarithmische Komplexität), daraus ergibt sich die Laufzeit von $O(M * \log(M))$.
- In den *M* Schalen können sich pro Schale *N* Sorten befinden, welche alle überprüft werden müssen, ob sie eine Wunschsorte sind. Dies wird für alle *N* möglichen Sorten mit dem *vector<bool> isWunsch* in konstanter Zeit ausgeführt. Für diesen Teilschritt ergibt sich die Laufzeit von $O(M * N)$.

Zusammenfassend benötigt man für diesen gesamten Schritt im worst case eine Laufzeit von $O(N * (M * \log(M) + M * N) = O(N * M * \log(M) + M * N^2)$.

Dies tritt jedoch nur in den seltensten Fällen auf, da hierfür jede Sorte mit jede Schale gleich oft beobachtet werden muss, d.h. alle beobachteten Spieße enthalten alle möglichen Obstsorten aus allen möglichen Schalen.

- Am Ende werden noch die maximal *M* Sorten ausgegeben

Das gesamte Programm besitzt somit eine Laufzeitkomplexität von
$f(N, M, P) = N + P * N * M + 2 * N * M + N * M * (\log(N) + \log(M)) + N * M * \log(M) + M * N^2$

Nehmen wir an, dass $P \leq N \leq M$ in allen Eingaben vorliegt, so gilt $f(N, M, P) \in O(n^3)$ mit $n = M$.

Das Programm besitzt also im worst case eine kubische Laufzeit.

Da eine Adjazenzmatrix verwendet wurde, ist die Speicherkomplexität $O(N * M)$.

Beispiele

Beispiel der BWINF Webseite

Einstiegsaufgabe aus der Aufgabenstellung (mit 3 Beobachtungen)

Eingabe:

6

Weintraube Brombeere Apfel

3

1 4 5

Apfel Banane Brombeere

3 5 6

Banane Pflaume Weintraube

1 2 4

Apfel Brombeere Erdbeere

Ausgabe:

Sorte: Weintraube

> kann sich in folgenden Schalen befinden: Nr. 3 6

>> in Schale Nr. 3 können sich auch folgende Obstsorten befinden: Pflaume

>> in Schale Nr. 6 können sich auch folgende Obstsorten befinden: Pflaume

Sorte: Brombeere

> kann sich in folgenden Schalen befinden: Nr. 1 4

>> in Schale Nr. 1 können sich auch folgende Obstsorten befinden: Apfel

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Apfel

Sorte: Apfel

> kann sich in folgenden Schalen befinden: Nr. 1 4

>> in Schale Nr. 1 können sich auch folgende Obstsorten befinden: Brombeere

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Brombeere

>>> Zusammenfassung: Donalds Wunschsorten können nicht sicher bestimmt werden!

Einstiegsaufgabe aus der Aufgabenstellung (nach 4 Beobachtungen)

Sorte: Weintraube

> ist sicher in Schale Nr. 3

Sorte: Brombeere

> kann sich in folgenden Schalen befinden: Nr. 1 4

>> in Schale Nr. 1 können sich auch folgende Obstsorten befinden: Apfel

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Apfel

Sorte: Apfel

> kann sich in folgenden Schalen befinden: Nr. 1 4

>> in Schale Nr. 1 können sich auch folgende Obstsorten befinden: Brombeere

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Brombeere

>>> Zusammenfassung: Donalds Wunschsorten können sicher bestimmt werden!

>>> Donalds Sorten für seinen Wunschspieß befinden sich in den Schalen Nr. 1 3 4

spiesse1.txt

Sorte: Clementine

> ist sicher in Schale Nr. 1

Sorte: Erdbeere

> kann sich in folgenden Schalen befinden: Nr. 2 4

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Himbeere

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Himbeere

Sorte: Grapefruit

> ist sicher in Schale Nr. 7

Sorte: Himbeere

> kann sich in folgenden Schalen befinden: Nr. 2 4

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Erdbeere

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Erdbeere

Sorte: Johannisbeere

> ist sicher in Schale Nr. 5

>>> Zusammenfassung: Donalds Wunschsorten können sicher bestimmt werden!

>>> Donalds Sorten für seinen Wunschspieß befinden sich in den Schalen Nr. 1 2 4 5 7

spiesse2.txt

Sorte: Apfel

> ist sicher in Schale Nr. 1

Sorte: Banane

> kann sich in folgenden Schalen befinden: Nr. 5 10 11

>> in Schale Nr. 5 können sich auch folgende Obstsorten befinden: Clementine Himbeere

>> in Schale Nr. 10 können sich auch folgende Obstsorten befinden: Clementine Himbeere

>> in Schale Nr. 11 können sich auch folgende Obstsorten befinden: Clementine Himbeere

Sorte: Clementine

> kann sich in folgenden Schalen befinden: Nr. 5 10 11

>> in Schale Nr. 5 können sich auch folgende Obstsorten befinden: Banane Himbeere

>> in Schale Nr. 10 können sich auch folgende Obstsorten befinden: Banane Himbeere

>> in Schale Nr. 11 können sich auch folgende Obstsorten befinden: Banane Himbeere

Sorte: Himbeere

> kann sich in folgenden Schalen befinden: Nr. 5 10 11

>> in Schale Nr. 5 können sich auch folgende Obstsorten befinden: Banane Clementine

>> in Schale Nr. 10 können sich auch folgende Obstsorten befinden: Banane Clementine

>> in Schale Nr. 11 können sich auch folgende Obstsorten befinden: Banane Clementine

Sorte: Kiwi

> ist sicher in Schale Nr. 6

Sorte: Litschi

> ist sicher in Schale Nr. 7

>>> Zusammenfassung: Donalds Wunschsorten können sicher bestimmt werden!

>>> Donalds Sorten für seinen Wunschspieß befinden sich in den Schalen Nr. 1 5 6 7 10 11

spiesse3.txt

Sorte: Clementine

> ist sicher in Schale Nr. 5

Sorte: Erdbeere

> ist sicher in Schale Nr. 8

Sorte: Feige

> kann sich in folgenden Schalen befinden: Nr. 7 10

>> in Schale Nr. 7 können sich auch folgende Obstsorten befinden: Ingwer

>> in Schale Nr. 10 können sich auch folgende Obstsorten befinden: Ingwer

Sorte: Himbeere

> ist sicher in Schale Nr. 1

Sorte: Ingwer

> kann sich in folgenden Schalen befinden: Nr. 7 10

>> in Schale Nr. 7 können sich auch folgende Obstsorten befinden: Feige

>> in Schale Nr. 10 können sich auch folgende Obstsorten befinden: Feige

Sorte: Kiwi

> ist sicher in Schale Nr. 12

Sorte: Litschi

> kann sich in folgenden Schalen befinden: Nr. 2 11

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Grapefruit

>> in Schale Nr. 11 können sich auch folgende Obstsorten befinden: Grapefruit

>>> Zusammenfassung: Donalds Wunschsorten können nicht sicher bestimmt werden!

spiesse4.txt

Sorte: Apfel

> ist sicher in Schale Nr. 9

Sorte: Feige

> ist sicher in Schale Nr. 13

Sorte: Grapefruit

> ist sicher in Schale Nr. 8

Sorte: Ingwer

> ist sicher in Schale Nr. 6

Sorte: Kiwi

> ist sicher in Schale Nr. 2

Sorte: Nektarine

> ist sicher in Schale Nr. 7

Sorte: Orange

> ist sicher in Schale Nr. 14

Sorte: Pflaume

> ist sicher in Schale Nr. 12

>>> Zusammenfassung: Donalds Wunschsorten können sicher bestimmt werden!

>>> Donalds Sorten für seinen Wunschspieß befinden sich in den Schalen Nr. 2 6 7 8 9 12 13 14

spiesse5.txt

Sorte: Apfel

> kann sich in folgenden Schalen befinden: Nr. 1 4 19

>> in Schale Nr. 1 können sich auch folgende Obstsorten befinden: Grapefruit Mango

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Grapefruit Mango

>> in Schale Nr. 19 können sich auch folgende Obstsorten befinden: Grapefruit Mango

Sorte: Banane

> kann sich in folgenden Schalen befinden: Nr. 3 9

>> in Schale Nr. 3 können sich auch folgende Obstsorten befinden: Quitte

>> in Schale Nr. 9 können sich auch folgende Obstsorten befinden: Quitte

Sorte: Clementine

> ist sicher in Schale Nr. 20

Sorte: Dattel

> ist sicher in Schale Nr. 6

Sorte: Grapefruit

> kann sich in folgenden Schalen befinden: Nr. 1 4 19

>> in Schale Nr. 1 können sich auch folgende Obstsorten befinden: Apfel Mango

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Apfel Mango

>> in Schale Nr. 19 können sich auch folgende Obstsorten befinden: Apfel Mango

Sorte: Himbeere

> ist sicher in Schale Nr. 5

Sorte: Mango

> kann sich in folgenden Schalen befinden: Nr. 1 4 19

>> in Schale Nr. 1 können sich auch folgende Obstsorten befinden: Apfel Grapefruit

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Apfel Grapefruit

>> in Schale Nr. 19 können sich auch folgende Obstsorten befinden: Apfel Grapefruit

Sorte: Nektarine

> ist sicher in Schale Nr. 14

Sorte: Orange

> kann sich in folgenden Schalen befinden: Nr. 2 16

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Sauerkirsche

>> in Schale Nr. 16 können sich auch folgende Obstsorten befinden: Sauerkirsche

Sorte: Pflaume

> ist sicher in Schale Nr. 10

Sorte: Quitte

> kann sich in folgenden Schalen befinden: Nr. 3 9

>> in Schale Nr. 3 können sich auch folgende Obstsorten befinden: Banane

>> in Schale Nr. 9 können sich auch folgende Obstsorten befinden: Banane

Sorte: Sauerkirsche

> kann sich in folgenden Schalen befinden: Nr. 2 16

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Orange

>> in Schale Nr. 16 können sich auch folgende Obstsorten befinden: Orange

Sorte: Tamarinde

> ist sicher in Schale Nr. 12

>>> Zusammenfassung: Donalds Wunschsorten können sicher bestimmt werden!

>>> Donalds Sorten für seinen Wunschspieß befinden sich in den Schalen Nr. 1 2 3 4 5 6 9 10 12 14 16 19 20

spiesse6.txt

Sorte: Clementine

> ist sicher in Schale Nr. 7

Sorte: Erdbeere

> ist sicher in Schale Nr. 10

Sorte: Himbeere

> ist sicher in Schale Nr. 18

Sorte: Orange

> ist sicher in Schale Nr. 20

Sorte: Quitte

> ist sicher in Schale Nr. 4

Sorte: Rosine

> kann sich in folgenden Schalen befinden: Nr. 11 15

>> in Schale Nr. 11 können sich auch folgende Obstsorten befinden: Ugli

>> in Schale Nr. 15 können sich auch folgende Obstsorten befinden: Ugli

Sorte: Ugli

> kann sich in folgenden Schalen befinden: Nr. 11 15

>> in Schale Nr. 11 können sich auch folgende Obstsorten befinden: Rosine

>> in Schale Nr. 15 können sich auch folgende Obstsorten befinden: Rosine

Sorte: Vogelbeere

> ist sicher in Schale Nr. 6

>>> Zusammenfassung: Donalds Wunschsorten können sicher bestimmt werden!

>>> Donalds Sorten für seinen Wunschspieß befinden sich in den Schalen Nr. 4 6 7 10 11 15 18 20

spiesse7.txt

Sorte: Apfel

> kann sich in folgenden Schalen befinden: Nr. 3 10 20 26

>> in Schale Nr. 3 können sich auch folgende Obstsorten befinden: Grapefruit Xenia Litschi

>> in Schale Nr. 10 können sich auch folgende Obstsorten befinden: Grapefruit Xenia Litschi

>> in Schale Nr. 20 können sich auch folgende Obstsorten befinden: Grapefruit Xenia Litschi

>> in Schale Nr. 26 können sich auch folgende Obstsorten befinden: Grapefruit Xenia Litschi

Sorte: Clementine

> ist sicher in Schale Nr. 24

Sorte: Dattel

> kann sich in folgenden Schalen befinden: Nr. 6 16 17

>> in Schale Nr. 6 können sich auch folgende Obstsorten befinden: Mango Vogelbeere

>> in Schale Nr. 16 können sich auch folgende Obstsorten befinden: Mango Vogelbeere

>> in Schale Nr. 17 können sich auch folgende Obstsorten befinden: Mango Vogelbeere

Sorte: Grapefruit

> kann sich in folgenden Schalen befinden: Nr. 3 10 20 26

>> in Schale Nr. 3 können sich auch folgende Obstsorten befinden: Apfel Xenia Litschi

>> in Schale Nr. 10 können sich auch folgende Obstsorten befinden: Apfel Xenia Litschi

>> in Schale Nr. 20 können sich auch folgende Obstsorten befinden: Apfel Xenia Litschi

>> in Schale Nr. 26 können sich auch folgende Obstsorten befinden: Apfel Xenia Litschi

Sorte: Mango

> kann sich in folgenden Schalen befinden: Nr. 6 16 17

>> in Schale Nr. 6 können sich auch folgende Obstsorten befinden: Dattel Vogelbeere

>> in Schale Nr. 16 können sich auch folgende Obstsorten befinden: Dattel Vogelbeere

>> in Schale Nr. 17 können sich auch folgende Obstsorten befinden: Dattel Vogelbeere

Sorte: Sauerkirsche

> kann sich in folgenden Schalen befinden: Nr. 8 14

>> in Schale Nr. 8 können sich auch folgende Obstsorten befinden: Yuzu

>> in Schale Nr. 14 können sich auch folgende Obstsorten befinden: Yuzu

Sorte: Tamarinde

> kann sich in folgenden Schalen befinden: Nr. 5 23

>> in Schale Nr. 5 können sich auch folgende Obstsorten befinden: Zitrone

>> in Schale Nr. 23 können sich auch folgende Obstsorten befinden: Zitrone

Sorte: Ugli

> kann sich in folgenden Schalen befinden: Nr. 18 25

>> in Schale Nr. 18 können sich auch folgende Obstsorten befinden: Banane

>> in Schale Nr. 25 können sich auch folgende Obstsorten befinden: Banane

Sorte: Vogelbeere

> kann sich in folgenden Schalen befinden: Nr. 6 16 17

>> in Schale Nr. 6 können sich auch folgende Obstsorten befinden: Dattel Mango

>> in Schale Nr. 16 können sich auch folgende Obstsorten befinden: Dattel Mango

>> in Schale Nr. 17 können sich auch folgende Obstsorten befinden: Dattel Mango

Sorte: Xenia

> kann sich in folgenden Schalen befinden: Nr. 3 10 20 26

>> in Schale Nr. 3 können sich auch folgende Obstsorten befinden: Apfel Grapefruit Litschi

>> in Schale Nr. 10 können sich auch folgende Obstsorten befinden: Apfel Grapefruit Litschi

>> in Schale Nr. 20 können sich auch folgende Obstsorten befinden: Apfel Grapefruit Litschi

>> in Schale Nr. 26 können sich auch folgende Obstsorten befinden: Apfel Grapefruit Litschi

Sorte: Yuzu

> kann sich in folgenden Schalen befinden: Nr. 8 14

>> in Schale Nr. 8 können sich auch folgende Obstsorten befinden: Sauerkirsche

>> in Schale Nr. 14 können sich auch folgende Obstsorten befinden: Sauerkirsche

Sorte: Zitrone

> kann sich in folgenden Schalen befinden: Nr. 5 23

>> in Schale Nr. 5 können sich auch folgende Obstsorten befinden: Tamarinde

>> in Schale Nr. 23 können sich auch folgende Obstsorten befinden: Tamarinde

>>> Zusammenfassung: Donalds Wunschsorten können nicht sicher bestimmt werden!

Eigene Beispiele

test1.txt (eine Sorte in mehreren Schalen möglich)

Eingabe:

15

Clementine Erdbeere Grapefruit Himbeere Johannisbeere

4

6 10 3

Banane Feige Ingwer

2 1 9 8 4

Apfel Clementine Dattel Erdbeere Himbeere

5 8 10 4 2 11 12

Apfel Erdbeere Feige Himbeere Johannisbeere

6 4 2 5 9

Dattel Erdbeere Himbeere Ingwer Johannisbeere

Ausgabe:

Sorte: Clementine

> ist sicher in Schale Nr. 1

Sorte: Erdbeere

> kann sich in folgenden Schalen befinden: Nr. 2 4

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Himbeere

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Himbeere

Sorte: Grapefruit

> kann sich in folgenden Schalen befinden: Nr. 7 13 14 15

>> in Schale Nr. 7 können sich auch folgende Obstsorten befinden:

>> in Schale Nr. 13 können sich auch folgende Obstsorten befinden:

>> in Schale Nr. 14 können sich auch folgende Obstsorten befinden:

>> in Schale Nr. 15 können sich auch folgende Obstsorten befinden:

Sorte: Himbeere

> kann sich in folgenden Schalen befinden: Nr. 2 4

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Erdbeere

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Erdbeere

Sorte: Johannisbeere

> ist sicher in Schale Nr. 5

>>> Zusammenfassung: Donalds Wunschsorten können nicht sicher bestimmt werden!

Wie man an diesem Beispiel gut sehen kann, kann das Programm die Eingabe auch verarbeiten, falls eine sich ein Obstsorte in mehreren Obstschalen befinden kann.

test2.txt

Eingabe:

15

Clementine Erdbeere Grapefruit Himbeere Johannisbeere Apfel

5

6 10 3

Banane Feige Ingwer

2 1 9 8 4

Apfel Clementine Dattel Erdbeere Himbeere

5 8 10 4 2 11 12

Apfel Erdbeere Feige Himbeere Johannisbeere

6 4 2 5 9

Dattel Erdbeere Himbeere Ingwer Johannisbeere

14 14

Apfel

Ausgabe

Sorte: Clementine

> ist sicher in Schale Nr. 1

Sorte: Erdbeere

> kann sich in folgenden Schalen befinden: Nr. 2 4

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Himbeere

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Himbeere

Sorte: Grapefruit

> kann sich in folgenden Schalen befinden: Nr. 7 13 15

>> in Schale Nr. 7 können sich auch folgende Obstsorten befinden:

>> in Schale Nr. 13 können sich auch folgende Obstsorten befinden:

>> in Schale Nr. 15 können sich auch folgende Obstsorten befinden:

Sorte: Himbeere

> kann sich in folgenden Schalen befinden: Nr. 2 4

>> in Schale Nr. 2 können sich auch folgende Obstsorten befinden: Erdbeere

>> in Schale Nr. 4 können sich auch folgende Obstsorten befinden: Erdbeere

Sorte: Johannisbeere

> ist sicher in Schale Nr. 5

Sorte: Apfel

> kann sich in folgenden Schalen befinden: Nr. 8 14

>>> Zusammenfassung: Donalds Wunschsorten können nicht sicher bestimmt werden!

Quellcode

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> vi;
typedef vector<string> vs;
typedef pair<int, int> pii;
typedef vector<pii> vpii;
typedef vector<vpii> vvpii;
typedef vector<vi> vvi;
typedef tuple<int, int, int> tiii;
typedef tuple<int, int, int, int> tiiii;
typedef vector<bool> vb;
```

```
#define what(x) cerr<<#x<<" is "<<x<<endl;

const int INF = numeric_limits<int>::max()/2;


int MAXN; // maximale Anzahl der N Obstsorten
vi wunsche; // die Wunschsorten

int M; // Anzahl der beobachteten Spieße

// Unordered map zum Konvertieren von Obstbezeichnungen zu Nummern
unordered_map<string, int> words2num;
unordered_map<int, string> num2words;

// Anzahl der verwendeten Obstnamen (wichtig für die Konvertierung der Namen)
// Zähler beginnt bei MAXN, damit Schalen und Sorten nicht vertauscht werden können
int counter_used_words;

// Hinzufügen eines Wortes zum Konverter
void add_word_to_converter(string& word) {
    if (words2num.find(word) == words2num.end()) {
        // Wort muss noch hinzugefügt werden

        words2num[word] = counter_used_words;
        num2words[counter_used_words] = word;
        counter_used_words ++;
    }
}

// gibt die Strings in einer Zeile in einem Vector zurück
vs getline2vector_string(string& input_line) {
```

```
    istringstream buffer(input_line);
    vs arr ((istream_iterator<string>(buffer)),
            istream_iterator<string>());
    return arr;
}

// gibt die Ints in einer Zeile in einem Vector zurück
vi getline2vector_int(string& input_line) {
    // konvertiert die Eingabe zu einem Vector aus Strings,
    // dieser wird nun zu Zahlen konvertiert
    vs strings = getline2vector_string(input_line);
    vi arr;
    for (string s : strings) {
        int a = stoi(s);
        arr.push_back(a);
    }
    return arr;
}


int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);

    cin >> MAXN;
    MAXN++;

    // Anzahl der bis jetzt verschiednen Wörter wird gezählt
    counter_used_words = 1;

    // Einlesen von Donalds Wünschen in einer Zeile
```

```
cin.ignore();
string donalds_wunsche_line;
getline(cin, donalds_wunsche_line);

vs donalds_wunsche = getline2vector_string(donalds_wunsche_line);

vb isWunsch(MAXN, false);

for (string word : donalds_wunsche) {
    // Donalds Wünsche zum Konvertieren hinzugefügt
    add_word_to_converter(word);

    // Nummer des Wunsches wird zu 'wunsche' hinzugefügt
    wunsche.push_back(words2num[word]);

    isWunsch[words2num[word]] = true;
}

vvi adj(MAXN, vi(MAXN, -1));

// Anzahl der Beobachtungen
cin >> M;
cin.ignore();

for (int i = 0; i<M; ++i) {
    string obstschalen_line, obstsorten_line;
    getline(cin, obstschalen_line);
    getline(cin, obstsorten_line);

    // Konvertieren der Zeileneingaben zu vector<string>
    vi obstschalen = getline2vector_int(obstschalen_line);
```

```
    vs obstsorten_string = getline2vector_string(obstsorten_line);

    vi obstsorten;
    for (string sorte : obstsorten_string) {
        add_word_to_converter(sorte);
        obstsorten.push_back(words2num[sorte]);
    }

    // Hinzufügen der Kantengewichtungen zwischen der aktuellen Obstsorte und Schalennummer
    for (int sorte : obstsorten) {
        for (int schale : obstschalen) {
            if (adj[sorte][schale] < 0)
                adj[sorte][schale] = 0;

            adj[sorte][schale] ++;
        }
    }
}

// maxValue-Verbindung für die jede Obstsorte bestimmen
vi maxVObstsorten(MAXN);

// für alle Obstsorten
for (int i = 1; i < MAXN; ++i) {
    int maxV = -INF;

    for (int j = 1; j < MAXN; ++j) {
        maxV = max(maxV, adj[i][j]);
    }

    maxVObstsorten[i] = maxV;
```

```
    }

    // maxValue-Verbindung für jede Schale bestimmen
    vi maxVSchalen(MAXN);

    // für alle Schalen
    for (int j = 1; j < MAXN; ++j) {
        int maxV = -INF;

        for (int i = 1; i < MAXN; ++i) {
            maxV = max(maxV, adj[i][j]);
        }

        maxVSchalen[j] = maxV;
    }

    vector<set<int> > sorte2schale(MAXN);
    vector<set<int> > schale2sorte(MAXN);
    for (int sorte = 1; sorte < MAXN; ++sorte) {
        for (int schale = 1; schale < MAXN; ++schale) {
            if (adj[sorte][schale] == maxVObstsorten[sorte] and adj[sorte][schale] == maxVSchalen[schale])
{
                // Schale ist möglich
                sorte2schale[sorte].insert(schale);
                schale2sorte[schale].insert(sorte);
            }
        }
    }

    bool works = true;
```

```
    set<int> finalSchalen;

    for (int w : wunsche) {
        cout << "Sorte: " << num2words[w] << "\n";

        if (sorte2schale[w].size() == 1) {
            cout << "> ist sicher in Schale Nr. " << *sorte2schale[w].begin() << "\n";

            finalSchalen.insert(*sorte2schale[w].begin());
        }
        else if (sorte2schale[w].empty()) {
            works = false;
            cout << "> kann keiner Schale zugewiesen werden!\n";
        }
        else {
            cout << "> kann sich in folgenden Schalen befinden: Nr. ";
            for (int schale : sorte2schale[w]) {
                cout << schale << " ";

                finalSchalen.insert(schale);
            }
            cout << "\n";

            for (int schale : sorte2schale[w]) {
                if (schale2sorte[schale].size() > 1) {

                    cout << ">> in Schale Nr. " << schale << " können sich auch folgende Obstsorten
befinden: ";

                    for (int sorte : schale2sorte[schale]) {
                        if (sorte != w) {
```

```
                cout << num2words[sorte] << " ";
            }

            // falls die Sorte kein Wunsch ist, ist sie ein Konkurrent
            // die Schalen für den Obstspieß sind dann nicht mehr eindeutig
            if (!isWunsch[sorte]) works = false;

        }
        cout << "\n";
    }

  }
 }

 cout << "\n";
}

// Fazit:
cout << "\n>>> Zusammenfassung: Donalds Wunschsorten können ";
if (!works) cout << "nicht ";
cout << "sicher bestimmt werden!\n";

if (works) {
 cout << ">>> Donalds Sorten für seinen Wunschspieß befinden sich in den Schalen Nr. ";
 for (int s : finalSchalen) cout << s << " ";
 cout << "\n";
}

return 0;
}
```