

# Aufgabe 2: Flohmarkt in Langdorf

Teilnahme-ID: 56860

Bearbeiter/-in dieser Aufgabe:  
Christopher Besch

28. Dezember 2020

## Inhaltsverzeichnis

<b>1</b>	<b>Lösungsidee</b>	<b>1</b>
<b>2</b>	<b>Umsetzung</b>	<b>2</b>
<b>3</b>	<b>Beispiele</b>	<b>3</b>
<b>4</b>	<b>Quellcode</b>	<b>3</b>

## 1 Lösungsidee

Die von Donald beobachteten Spieße lassen sich wie folgt darstellen:

Schüsseln	Früchte
1, 4, 5	Apfel, Banane, Brombeere
3, 5, 6	Banane, Pflaume, Weintraube
1, 2, 4	Apfel, Brombeere, Erdbeere
2, 6	Erdbeere, Pflaume

**Bestimmung der legalen Früchte** Nun wird in einer Tabelle jede Schüssel mit den Früchten der Spieße, die diese Schüssel verwendeten, aufgelistet (eine Spalte pro Speiß). Zudem werden in einer weiteren Spalte alle Früchte notiert, die in Spießen vorkommen, die nicht diese Schüssel verwenden, sie werden verbotene Früchte genannt. Diese Früchte können nicht von dieser Schüssel sein.

Schüssel	Spieße von dieser Schüssel		Verbotene Früchte
1	Apfel, Banane, Brombeere	Apfel, Brombeere, Erdbeere	Banane, Weintraube, Erdbeere, Pflaume
2	Apfel, Brombeere, Erdbeere	Erdbeere, Pflaume	Apfel, Brombeere, Banane, Weintraube, Pflaume
3	Banane, Pflaume, Weintraube		Banane, Apfel, Brombeere, Erdbeere, Pflaume
4	Apfel, Banane, Brombeere	Apfel, Brombeere, Erdbeere	Banane, Weintraube, Erdbeere, Pflaume
5	Apfel, Banane, Brombeere	Banane, Pflaume, Weintraube	Apfel, Brombeere, Erdbeere, Pflaume
6	Banane, Pflaume, Weintraube	Erdbeere, Pflaume	Banane, Apfel, Brombeere, Erdbeere

Nun werden für jede Schüssel alle möglicherweise vorliegenden Früchte gesucht, diese werden im Folgenden legale Früchte genannt. An jede Schüssel liegt eine Frucht aus der Menge der legalen Früchte für diese Schüssel.

Alle legalen Früchte für eine bestimmte Schüssel müssen zwei Voraussetzungen erfüllen:

1. Die Frucht muss in allen Spießen vorkommen, die diese Schüssel benutzen.
2. Die Frucht darf für diese Schüssel nicht verboten sein.

Alle legalen Früchte werden nun markiert:

Schüssel	Spieße von dieser Schüssel		Verbotene Früchte
1	<u>Apfel</u> , Banane, <u>Brombeere</u>	<u>Apfel</u> , <u>Brombeere</u> , Erdbeere	Banane, Weintraube, Erdbeere, Pflaume
2	Apfel, Brombeere, <u>Erdbeere</u>	<u>Erdbeere</u> , Pflaume	Apfel, Brombeere, Banane, Weintraube, Pflaume
3	Banane, Pflaume, <u>Weintraube</u>		Banane, Apfel, Brombeere, Erdbeere, Pflaume
4	<u>Apfel</u> , Banane, <u>Brombeere</u>	<u>Apfel</u> , <u>Brombeere</u> , Erdbeere	Banane, Weintraube, Erdbeere, Pflaume
5	Apfel, <u>Banane</u> , Brombeere	<u>Banane</u> , Pflaume, Weintraube	Apfel, Brombeere, Erdbeere, Pflaume
6	Banane, <u>Pflaume</u> , Weintraube	Erdbeere, <u>Pflaume</u>	Banane, Apfel, Brombeere, Erdbeere

**Auswahl der Schalen für Donald** Die folgenden Früchte sind (von Donald) gefordert: Weintraube, Brombeere und Apfel

Es werden alle Schüsseln durchgegangen.

- Wenn alle legalen Früchte einer Schüssel gefordert sind, wird diese Schüssel ausgewählt, da ganz egal, welche legale Frucht die tatsächlich an der Schüssel liegende ist, die Schüssel eine geforderte Frucht liefert.
- Wenn keine der legalen Früchte gefordert sind, ist dies kein ausgewählter Schüssel.
- Wenn einige, aber nicht alle legalen Früchte gefordert sind, ist nicht genügend Information vorhanden, da nicht gesagt werden kann, ob der Schüssel eine ausgewählte Frucht enthält oder nicht.

Es ergibt sich die Menge aller ausgewählten Schüsseln: 1, 2 und 3

Die Schüsseln 1 und 2 liefern jeweils entweder Apfel oder Brombeeren. Da beide Früchte gefordert sind, ist diese Ungewissheit irrelevant.

Mit dieser Methode kann in allen Fällen eine Aussage gefällt werden, ob genügend Information vorliegt und wenn dies der Fall ist, die Menge der ausgewählten Schüsseln angegeben werden.

## 2 Umsetzung

Die Lösungsidee wird in C++ implementiert.

**Einlese der Eingabedatei** Als erster Schritt wird in der Funktion `read_file` die Eingabedatei gelesen. Hierbei wird überprüft, ob die Eingabedatei dem gegebenen Format entspricht, wenn nicht wird das Programm abgebrochen. Hierzu wird ein Makro `raise_error()` verwendet, dass die Ausführung des Programmes abbricht und eine möglichst informative Fehlermeldung zurückgibt. Dieses Makro wird ebenfalls für alle Methoden aller Klassen verwendet, um z.B. Segmentation Faults zu verhindern.

Die Fruchtamen werden mithilfe einer Instanz der Klasse `LookupTable` aufsteigend und anfangend bei 0 in Glanzzahlen, Frucht IDs, übersetzt. Dadurch ist die Anzahl an verschiedenen Früchten nicht limitiert. Ein weiterer Vorteil ist, dass da die ID aufsteigend und bei 0 anfangend verteilt werden, die IDs als Indizes in Bit Fields verwendet werden können (hierzu wird `std::vector<bool>` verwendet, da die Anzahl der Früchte bei compile time nicht bekannt ist). So liegt für jede Frucht ein Bit/Bool vor, ist er `true`, ist diese Frucht enthalten, ist sie `false`, ist sie nicht enthalten. Diese kompakte Datenstruktur ist

ressourcenschonender im Vergleich zu einem `std::vector<std::string>` mit den Namen aller enthaltenen Früchten.

Die Namen für die Schlüssel werden auf die gleiche Weise behandelt. Zwar ist angegeben, dass alle Schlüssel nummeriert sind, es ist allerdings nicht klar, wie die Nummern verteilt werden, daher ergibt es Sinn, die Namen der Schlüssel als Strings zu behandeln und ebenfalls eine Instanz von **LookupTable** zu verwenden. So können auch für die Schlüssel Bit Fields verwendet werden. Ein zusätzlicher Vorteil ist, dass so jegliche Namen für die Schlüssel erlaubt sind.

Als Größe für die Bit Fields wird die angegebene Anzahl an Früchten verwendet.

Es werden Instanzen der Klasse **Skewer** erstellt, diese enthalten die Menge der Früchte auf einem Spieß und die Menge der Schlüssel, von denen die Früchte genommen sind. Hierbei werden, wie bereits erwähnt, Bit Fields verwendet. An dieser Stelle werden fehlende Schlüssel und fehlende Früchte hinzugefügt, sodass die Anzahl an Schlüssel und Früchten der angegebenen Anzahl an Früchten entspricht. Diese neuen Früchte und Schlüssel erhalten Namen wie „new\_bowl0“ oder „new\_fruit0“. Es wird überprüft, ob bereits eine Frucht mit dem Namen „new\_fruit0“ vorliegt, ist dies der Fall wird „new\_fruit1“ versucht usw. Gleiches gilt für neue Schlüsselnamen.

**Bestimmung der Legalen Früchte** In der Funktion **determine\_legal\_fruits** werden die legalen Früchte bestimmt, wie bereits in der Lösungsidee beschriebene.

Zuerst werden dazu für jede Schale **Bowl** Instanzen erstellt. Diese enthalten **allowed\_fruit\_sets**, ein `std::vector` mit einem Bit Field pro Spieß, der diese Schale verwendet, alle verbotenen Früchte (**disallowed\_fruits**), alle legalen Früchte (**legal\_fruits**) und zwei bools, **selected** und **possible\_selected**, auf die später eingegangen wird. Es werden alle Spieße durchgegangen, wenn der Spieß diese Schale verwendet, werden dessen Früchte zu **allowed\_fruit\_sets** hinzugefügt, wenn nicht zu **disallowed\_fruits**.

Nun werden die legalen Früchte für jede Schale bestimmt. Dafür werden für jede Frucht die beiden in der Lösungsidee genannten Voraussetzungen überprüft.

**Auswahl der Schalen** Im letzten Schritt werden alle Schalen durchgegangen. Hierbei wird überprüft, wie viele der legalen Früchte einer Schale gefordert sind. Sind alle gefordert wird **selected** auf *true* gesetzt, sind einige aber nicht alle gefordert, wird **possible\_selected** auf *true* und **possible** auf *false* gesetzt.

Wenn **possible** nach Durchlauf aller Schalen *true* bleibt, ist genügend Information vorhanden, um eine Aussage treffen zu können.

**main** In der main Funktion werden die drei genannten Funktionen ausgeführt und die Ergebnisse in der Konsole ausgegeben.

## 3 Beispiele

## 4 Quellcode