Aufgabe 2: Spießgesellen

Teilnahme-ID: 58240

Bearbeiter dieser Aufgabe:
Leandro Conte

10. Februar 2021

Inhaltsverzeichnis

Lösungsidee

Manuelle Vorgehensweise für Beispiel a)

Bevor ich eine Lösungsidee für allgemeine Eingaben ausgearbeitet habe, habe ich zuerst die Eingabe von a) gelöst. Gegeben sind 4 Fruchtspieße. Gesucht sind die Schüsseln von Weintraube, Brombeere und Apfel.

Mickys Spieß		Minnies Spieß		Gustavs Spieß		Daisys Spieß	
Apfel	1	Banane	3	Apfel	1	Erdbeere	2
Banane	4	Pflaume	5	Brombeere	2	Pflaume	6
Brombeere	5	Weintraube	6	Erdbeere	4		

Beim betrachten dieser Spieße sind mir folgende Sachen aufgefallen:

1. Nur die Banane ist sowohl auf Mickys, als auch auf Minnies Spieß. Da nur Schüssel 5 bei beiden Spießen beobachtet wurde, muss Banane in 5 sein.
   Stand: (Banane : 5)
2. Nur die Erdbeere ist sowohl auf Daisys, als auch auf Gustavs Spieß. Da nur Schüssel 2 bei beiden Spießen beobachtet wurde, muss Erdbeere in 2 sein.
   Stand: (Banane : 5); (Erdbeere : 2)
3. Da der Erdbeere Schüssel 2 zugewiesen ist, bleibt der Pflaume auf Daisys Spieß nur noch Schüssel 6 übrig
   Stand: (Banane : 5); (Erdbeere : 2); (Pflaume : 6)
4. Auf Minnies Spieß sind schon die Schüsseln 5 und 6 zugewiesen. Deswegen muss Weintraube in der übrig bleibenden Schüssel 3 sein.
   Stand : (Banane: 5); (Erdbeere : 2); (Pflaume : 6); (Weintraube : 3)
5. Es wurden jetzt schon die Schüsseln von 4 Früchten bestimmt. Die übrigen Früchte: Apfel und Brombeere sind beide nur auf Mickys und Gustavs Spieß. Aus den möglichen Schüsseln 1 und 4 , ist es unmöglich eine bestimmte dieser Schüsseln dem Apfel und eine andere der Brombeere zuzuweisen. Dazu reichen die gegebenen Informationen nicht aus.

   Da aber Apfel und Brombeere gefragt sind, sind auch beide Schüsseln 1 und 4 gefragt.

   Endstand:
   (Banane: 5); (Erd... : 2); (Pflaume : 6); (Wein... : 3); (Apfel : 1 oder 4); (Brom... : 1 oder 4)

Ergebnis:

Um die Fruchtsorten Weintraube, Brombeere und Apfel zu erhalten muss sich Donald aus den Schüsseln 1, 3 und 4 bedienen.

Allgemeine Fälle

Nun soll eine Methode gefunden werden, um das Problem für allgemeine Eingaben zu lösen. Als Erstes werde ich die Eingabe in Form verschiedener Mengen definieren:

O_{ges} ist die Menge der Fruchtsorten. (Jeder „Fruchtstring“ wird als Zahl von 1 bis F dargestellt)

S_{ges} ist die Menge der Schüsseln.

$|O_{ges}| = |S_{ges}| = F$

Zusätzlich sind N beobachtete Fruchtspieße bekannt.

A_i ist die Menge der Fruchtsorten einer Beobachtung i

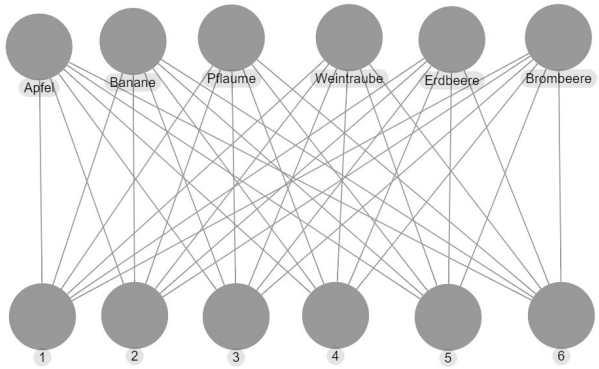
B_i ist die Menge der Schüsseln einer Beobachtung i

Jetzt sollen für eine Menge W an Fruchtsorten diejenigen Schüsseln gefunden werden, in denen sich die Früchte von W, gemäß den gegebenen Beobachtungen, befinden. Es ist nicht garantiert, dass die vorhandenen Informationen dafür ausreichen. In diesem Fall soll für jede Schüssel die Wahrscheinlichkeit gefunden werden, dass eine gesuchte Frucht in der Schüssel liegt.

Darstellung als Graph und „Gruppen“

Es gibt F! Möglichkeiten den Schüsseln die Früchte zuzuweisen. Nicht jede Möglichkeit ist mit den gegebenen Beobachtungen vereinbar. Man kann das ganze System als bipartiten Graphen visualisieren. Statt einem bestimmtem Matching wird hier jedoch für jede Schüssel die Anzahl an Matchings gesucht, bei denen eine gesuchte Frucht mit der bestimmten Schüssel gematcht ist.

Zu Beginn, wenn also noch keine der Beobachtungen verarbeitet wurde, ist jedes Element in O_{ges} mit jedem Element in S_{ges} verbunden. Alle Kombination sind möglich.



Der Graph

Der Graph enthält am Anfang genau eine „Gruppe“.

„Gruppe“ steht hier für einen Teilgraphen für den gilt:

1. Kein Element des Teilgraphen ist mit einem Element außerhalb des Teilgraphen verbunden.
2. Der Teilgraph enthält genauso viele Früchte wie Schüsseln.
3. Jede Frucht des Teilgraphen ist mit jeder Schüssel des Teilgraphen verbunden.

Solch eine Gruppe hat folgende Eigenschaften

1. Da keines der Elemente der Gruppe eine Beziehung zu den anderen Elementen außerhalb der Gruppe hat sind auch die Kombinationsmöglichkeiten, Matchings, in der Gruppe unabhängig vom Rest.
2. Da jede Schüssel mit jeder Frucht verbunden ist, ist die Wahrscheinlichkeit, dass eine Schüssel s der Gruppe eine Frucht f der gleichen Gruppe enthält gleich 1 dividiert durch die gesamte Anzahl an Früchten in der Gruppe.

Eine Gruppe kann ohne neue Informationen nicht verändert werden und jedes Element kann in maximal einer Gruppe sein.

Wenn man also weiß, dass eine Schüssel in einer Gruppe ist und welche Früchte in dieser Gruppe sind, ist das berechnen der Wahrscheinlichkeit, dass eine bestimmte Frucht in der Schüssel liegt, einfach.

$$P(\text{Frucht f in Schüssel s}) = \frac{1}{|\textit{Früchte der Gruppe von s}|} \textit{ für } f \in (\textit{Gruppe von } s), \textit{ansonsten } 0$$

Folgende Hypothese soll deswegen bewiesen werden:

Jedes Element des Graphen ist an jedem Punkt während der Verarbeitung in genau einer Gruppe.

Um dies zu beweisen untersuche ich, wie das Anwenden neuer Information/Beobachtungen den Graph beeinflusst. Da der gesamte Graph in seiner Anfangskonfiguration bereits einer Gruppe entspricht, muss nur bewiesen werden, dass das Ergebnis der Anwendung einer Beobachtung auf eine Gruppe diese komplett in eine oder mehrere Gruppen einteilt.

Was bedeute das Anwenden einer Beobachtung?

Wenn eine Beobachtung auf den Graphen angewandt wird muss fortan gelten:

1. Jede Frucht in A_i darf ab jetzt nur noch mit Schüsseln verbunden sein die in B_i sind.

2. Jede Schüssel in B_i darf ab jetzt nur noch mit Früchten verbunden sein die in A_i sind.

Damit vorhergegangene Beobachtungen weiterhin gelten werden Kanten immer nur entfernt und nie hinzugefügt.

Angenommen eine Beobachtung i, mit Früchten A_i und Schüsseln B_i, soll auf eine Gruppe g mit C_g Früchten und D_g Schüsseln angewandt werden. Es gibt zwei Fälle. Entweder die Gruppe bleibt unverändert, oder sie wird in zwei gespalten. Der 1. Fall passiert dann, wenn $A_i = C_g \wedge B_i = D_g$. Warum entstehen im 2. Fall zwei neue Gruppen? C_g ist vollständig mit D_g verbunden. Wenn jetzt alle Kanten $A_i \cap C_g$ zu $B_i \cap D_g{}'$ und $A_i{}' \cap C_g$ zu $B_i \cap D_g$ entfernt werden, entstehen zwei Gruppen. Eine besteht aus den Früchten $A_i \cap C_g$ und den Schüsseln $B_i \cap D_g$ und die andere aus den Früchten $A_i{}' \cap C_g$ und den Schüsseln $B_i{}' \cap D_g$. Solange die neu angewandte Beobachtung nicht im Widerspruch zu einer alten steht sind aus der ursprünglichen Gruppe zwei neue geworden, für die beide alle Voraussetzungen einer Gruppe gelten.

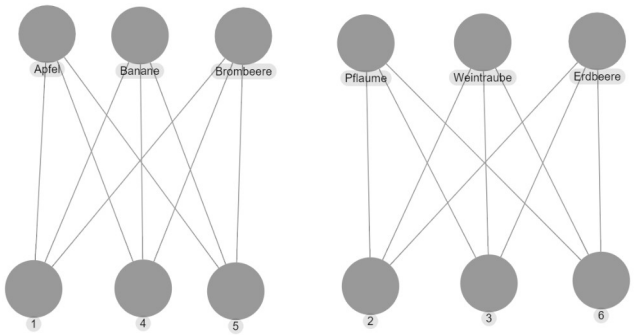
Da nicht alle Elemente einer Beobachtung in der gleichen Gruppe sein müssen, kann es sein, dass die Beobachtung auf mehrere Gruppen angewandt werden muss. Auch in diesem Fall besteht der Graph nach Anwendung weiterhin nur aus Gruppen.

Da der Graph nur durch die Anwendung von Beobachtungen modifiziert wird und anfangs aus einer Gruppe besteht ist deswegen die Aussage, dass nach Anwendung aller Beobachtungen (und auch davor) jedes Element exakt einer Gruppe angehört, korrekt.

Ein Beispiel

Ich werde jetzt die Lösungsidee anhand eines Beispiels, der Aufgabenstellung a), illustrieren.

Es sollen zuerst die Informationen der 1. Beobachtung auf den Graph angewandt werden.
$A_1 = \{\text{Apfel; Banane; Brombeere}\}$ $B_1 = \{1; 4; 5\}$

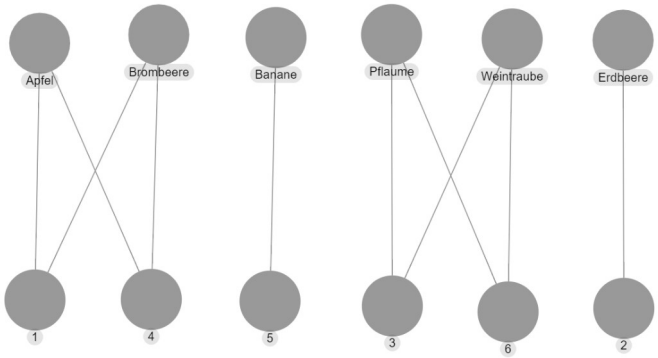


Nachdem die 1. Beobachtung angewandt wurde

Der ursprüngliche Gruppe wurde in zwei gespalten.

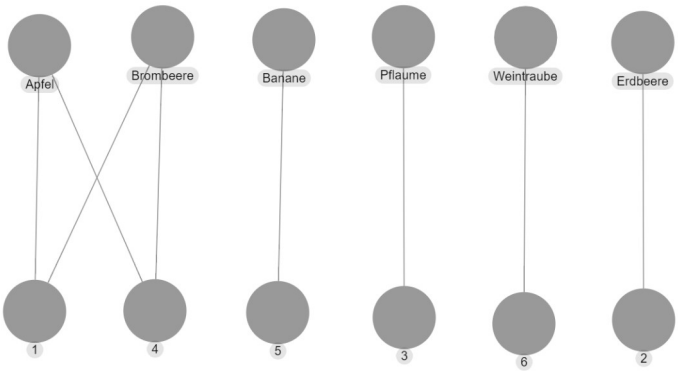
Als nächstes wird die 2. Beobachtung angewandt.

$A_2 = \{\text{ Banane; Pflaume; Weintraube }\}$ $B_2 = \{3; 5; 6\ \}$



Nachdem die 2. Beobachtung angewandt wurde

Im Fall von a) sieht der Graph dann so aus:



Nachdem alle Beobachtungen angewandt worden sind

Der Graph enthält 5 Gruppen, die nicht weiter reduziert werden können.

Zusammenfassung des Vorgangs

Als erstes wird ein Graph erstellt, auf den dann nacheinander alle Beobachtungen, wie im Beispiel gezeigt, angewandt werden.

Nachdem der Graph auf ein Minimum an Kanten, bzw. Gruppen, beschränkt wurde, wird dann anhand der Gruppen dieses Graphen für jede Schüssel die Wahrscheinlichkeit berechnet, dass eine gesuchte Frucht in der Schüssel liegt. Die Wahrscheinlichkeit, dass eine gesuchte Frucht in der Schüssel s ist beträgt $\frac{|W \cap C_g|}{|C_g|}$, für $s \in D_g$

In diesen 2 Schritten werden die anfänglich in Form der Spieße gegebenen Informationen in ihre nützlichste Form „raffiniert“.

Umsetzung

Struktur

Ich habe meine Haupt-Implementation in zwei Teile geteilt. Dadurch werden die einzelnen Teile übersichtlicher und die Logik einfacher zu erweitern. Zusätzlich befinden sich im Ordner Aufgabe2 noch drei Erweiterungsdateien, sowie alle beispiels Ein- und Ausgaben.

aufgabe2_common.py

aufgabe2_common.py enthält die wichtigsten Datencontainer und sekundären Verarbeitungsfunktionen. Die Eingabe wird von der *readInput* Funktion gelesen, welche einen Dateinamen erhält und ein Objekt der *TaskInput* Klasse zurückgibt. Dieses Objekt kann dann verarbeitet werden. Die Programmausgabe wird in Objekten der *TaskOutput* Klasse dargestellt und kann durch die *writeOutput* Funktion in eine Datei geschrieben werden.

Diese Ausgabefunktion erhält die Parameter *verbose* und *detailed*, die bestimmen was ausgegeben werden soll. Eine Standard Ausgabe Datei enthält in der 1. Zeile für jede Schüssel die Wahrscheinlichkeit das sie eine gewünschte Frucht enthält, in der 2. Zeile alle Schüsseln in denen sicher eine gewünschte Frucht liegt und in der 3. Zeile alle Schüsseln in denen eine gewünschte Frucht liegen könnte, mit der wahrscheinlichsten Schüssel zuerst. Wenn *verbose*=True ist, werden die einzelnen Zeilen in der Datei beschrieben.

Beim Einlesen werden die Strings, welche die Frucht-Typen darstellen in Zahlen beginnend bei 0 umgewandelt. Das gleiche geschieht mit den Schüsseln, ihre Zahl ist intern um 1 verringert. Ich verwende das typing Modul und die darin enthaltenen *NewType* Funktion um die Schüsseln und Früchte als Typ *Bowl* und *Fruit* zu schreiben, obwohl sie eigentlich *int* sind.

aufgabe2.py

aufgabe2.py enthält die Umsetzung der Verarbeitungsschritte aus der Lösungsidee.

Die Funktion process_static nimmt ein TaskInput Objekt und gibt ein TaskOutput Objekt zurück.

In dieser Funktion wird zuerst wird eine Liste *pFruits* initialisiert. *pFruits* ist die Adjazenzliste der Schüsseln zu den Früchten, nur dass die verbundenen Früchte in einem *set* sind und nicht einer weiteren Liste. Dies vereinfacht darauffolgende Operationen.

Dann werden auf diesen Graphen, wie von der Lösungsidee beschrieben, alle Beobachtungen angewandt. Dazu wird die Funktion *applyObservation* verwendet. Diese nimmt eine Beobachtung in Form eines *Observation* Objekts und eine Referenz zu *pFruits*. Die Standard *set* Struktur besitzt alle notwendigen Methoden, wie *intersection_update* und *difference_update*, um Beobachtungen, wie in der Lösungsidee angegeben, auf den Graph anzuwenden. Die in der Lösungsidee genannten Gruppen sind implizit im Graphen enthalten.

Nachdem alle Beobachtungen angewandt wurden wird eine zweite *Adjazenzliste pBowls* erstellt, diesmal für jede Frucht die verbundenen Schüsseln. Diese zweite Liste enthält keine neuen Information, sondern ist nur eine andere Art den Graphen darzustellen.

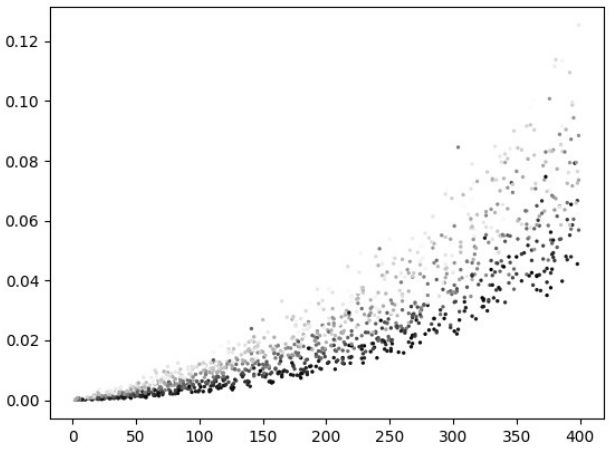
Bevor alles zurückgegeben wird, werden noch mit der *getProbabilities* Funktion die Wahrscheinlichkeiten der Schüsseln berechnet und als Liste zurückgegeben.

Diese Wahrscheinlichkeiten, sowie *pBowls*, *pFruits* und eine Referenz zur Eingabe, werden dann in *TaskOutput* zusammengefasst und zurückgegeben. Die Funktion *writeOutput* aus *aufgabe2_common.py* schreibt diese Ausgabe dann in eine Datei.

Laufzeit

Das Erstellen von pFruits, von pBowls und das berechnen der Wahrscheinlichkeiten hat jeweils eine maximale Laufzeit von $O(F^2)$. Das Anwenden der Beobachtungen hat eine maximale Laufzeit von $O(N \cdot F \cdot a)$. a ist hier die durchschnittliche Beobachtungsgröße.

Die Laufzeit der gesamten Verarbeitung beträgt maximal $O(3 \cdot F^2 + F \cdot N \cdot a)$. Das Programm ist also schnell genug für praktische Anwendungszwecke, es sei denn das Buffet enthält 10^6 verschiedene Schüsseln.



x: F y: Laufzeit
(Je heller der Punkt ist, desto größer ist N)

aufgabe2_generator.py

Die Funktion *generateCase* gibt eine zufällige Eingabe innerhalb spezifizierter Beschränkungen zurück. Zusätzlich kann ein seed eingegeben werden.

aufgabe2_tester.py

Hier sind verschiedene Funktionen enthalten, mit denen die Leistung der Verarbeitungsfunktionen getestet und verglichen werden kann, wie zum Beispiel *testRandom* mit der die Laufzeit-Abbildung erstellt wurde.

Erweiterungen

Interaktives Lösungsprogramm

In einem realistischen Szenario würde Donald nicht alle Schüsseln auf einmal benutzen sondern von Schüssel zu Schüssel gehen. Es würde für ihn Sinn machen die Schüsseln mit der größten Wahrscheinlichkeit zuerst zu benutzen, da er so schneller gesuchte Früchte finden kann. Eine Möglichkeit für eine Erweiterung wäre also zum Beispiel ein Programm, dass einen optimalen Pfad von Schüssel zu Schüssel findet. Dieses neue Pfad Problem ist interessant, da es eine ganze Reihe an neuen Problemen einführt, wie zum Beispiel, dass sich der Schüssel zu Schüssel Graph, während Donald ihn durchgeht verändert, und ich weiß nicht, ob es für das Problem eine optimale Lösung gibt.

Ein weiteres einfacheres Problem ist jedoch auch, dass die berechneten Wahrscheinlichkeiten nicht unabhängig von neuen Informationen sind. Findet Donald in Schüssel s Frucht f , dann entspricht dies einer neuen Beobachtung mit $A_i = \{f\}$ und $B_i = \{s\}$. Dadurch könnten sich die Gruppen und die aus ihnen berechneten Wahrscheinlichkeiten verändern und manche Schüsseln könnten ausgeschlossen werden.

aufgabe2_interactive.py

Diese Erweiterungsdatei enthält deswegen, damit Donald nicht manuell die Wahrscheinlichkeiten aktualisieren muss, ein interaktives Programm, welches die nächste Schüssel vorschlägt und nach der anfänglichen Berechnung neue Beobachtungen verarbeitet. Dazu verwendet es die *applyObservation* Funktion aus *aufgabe2.py*, um den Graphen auf dem neuesten Stand zu halten.

Kollidierende Eingaben

Die aktuelle Implementation nimmt an, das die Eingabe korrekt ist und keine Widersprüche enthält. Es könnte jedoch in der echten Welt vorkommen, dass Donald einen Spieß falsch beobachtet und deswegen die in das Programm eingegebenen Beobachtungen nicht miteinander übereinstimmen. Eine mögliche Erweiterung wäre einmal ein Programm, dass die Eingabe auf Fehler überprüft, oder auch ein Programm das keine absoluten Beobachtungen verarbeitet, sondern nur Beobachtungen mit einer bestimmten Wahrscheinlichkeit, dass sie richtig sind. Das Prüf-Programm wäre einfach zu implementieren, da man nur überprüfen muss, dass bestimmte Mengen, die Schnittmengen der Gruppen mit den Beobachtungen die korrekte Größe haben. Die andere Erweiterung hingegen würde eine komplett neue Verarbeitungsmethode brauchen.

Beispiele

Für alle Ausgaben ist detailed=True und verbose=True

Beispiel 0

Eingabe: Informationen aus a) (spiesse0.txt)

Ausgabe:

```
Ergebnisse für spiesse0.txt
Laufzeit: 8.049999999999724e-05
Wahrscheinlichkeit, dass Schüssel i gesuchte Frucht enthält: 1 0 1 1 0 0
Schüsseln, die gesuchte Früchte enthalten: 1 3 4
Schüsseln, die gesuchte Früchte enthalten könnten:
Mögliche Schüsseln für jede Frucht:
Weintraube 3
Brombeere 1 4
Apfel 1 4
Banane 5
Pflaume 6
Erdbeere 2
```

Beispiel 1

Eingabe: spiesse1.txt

Ausgabe:

```
Ergebnisse für spiesse1.txt
Laufzeit: 9.549999999999836e-05
Wahrscheinlichkeit, dass Schüssel i gesuchte Frucht enthält: 1 1 0 1 1 0 1 0 0 0
Schüsseln, die gesuchte Früchte enthalten: 1 2 4 5 7
Schüsseln, die gesuchte Früchte enthalten könnten:
Mögliche Schüsseln für jede Frucht:
Clementine 1
Erdbeere 2 4
Grapefruit 7
Himbeere 2 4
Johannisbeere 5
Banane 3
Feige 10
Ingwer 6
Apfel 8
Dattel 9
```

Beispiel 2

Eingabe: spiesse2.txt

Ausgabe:

```
Ergebnisse für spiesse2.txt
Laufzeit: 0.00016690000000000454
Wahrscheinlichkeit, dass Schüssel i gesuchte Frucht enthält: 1 0 0 0 1 1 1 0 0 1 1 0
Schüsseln, die gesuchte Früchte enthalten: 1 5 6 7 10 11
Schüsseln, die gesuchte Früchte enthalten könnten:
Mögliche Schüsseln für jede Frucht:
Apfel 1
Banane 10 11 5
Clementine 10 11 5
```

Himbeere 10 11 5
Kiwi 6
Litschi 7
Dattel 9 2
Erdbeere 8
Feige 9 2
Johannisbeere 4
Ingwer 12
Grapefruit 3

Beispiel 3

Eingabe: spiesse3.txt

Ausgabe:

Ergebnisse für spiesse3.txt
Laufzeit: 0.0005143000000000023
Wahrscheinlichkeit, dass Schüssel i gesuchte Frucht enthält: 1 0.5 0 0 1 0 1 1 0 1 0.5 1 0 0 0
Schüsseln, die gesuchte Früchte enthalten: 1 5 7 8 10 12
Schüsseln, die gesuchte Früchte enthalten könnten: 2 11
Mögliche Schüsseln für jede Frucht:
Clementine 5
Erdbeere 8
Feige 10 7
Himbeere 1
Ingwer 10 7
Kiwi 12
Litschi 2 11
Grapefruit 2 11
Johannisbeere 9
Nektarine 4
Orange 3
Dattel 13
Apfel 14 6
Banane 14 6
Unknown0 15

Beispiel 4

Eingabe: spiesse4.txt

Ausgabe:

Ergebnisse für spiesse4.txt
Laufzeit: 0.0001785000000000504
Wahrscheinlichkeit, dass Schüssel i gesuchte Frucht enthält: 0 1 0 0 0 1 1 1 1 0 0 1 1 1 0 0 0
Schüsseln, die gesuchte Früchte enthalten: 2 6 7 8 9 12 13 14
Schüsseln, die gesuchte Früchte enthalten könnten:
Mögliche Schüsseln für jede Frucht:
Apfel 9
Feige 13
Grapefruit 8
Ingwer 6
Kiwi 2
Nektarine 7
Orange 14
Pflaume 12
Clementine 15
Erdbeere 16
Himbeere 11
Mango 1
Banane 17

Quitte 4
Litschi 3
Dattel 10
Johannisbeere 5

Beispiel 5

Eingabe: spiesse5.txt

Ausgabe:

Ergebnisse für spiesse5.txt
Laufzeit: 0.00039979999999999877
Wahrscheinlichkeit, dass Schüssel i gesuchte Frucht enthält: 1 1 1 1 1 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1
Schüsseln, die gesuchte Früchte enthalten: 1 2 3 4 5 6 9 10 12 14 16 19 20
Schüsseln, die gesuchte Früchte enthalten könnten:
Mögliche Schüsseln für jede Frucht:
Apfel 1 19 4
Banane 9 3
Clementine 20
Dattel 6
Grapefruit 1 19 4
Himbeere 5
Mango 1 19 4
Nektarine 14
Orange 2 16
Pflaume 10
Quitte 9 3
Sauerkirsche 2 16
Tamarinde 12
Johannisbeere 13
Kiwi 7 15
Litschi 7 15
Rosine 17
Ingwer 11
Erdbeere 8
Unknown0 18

Beispiel 6

Eingabe: spiesse6.txt

Ausgabe:

Ergebnisse für spiesse6.txt
Laufzeit: 0.0007527999999999979
Wahrscheinlichkeit, dass Schüssel i gesuchte Frucht enthält: 0 0 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 1 0 0 0
Schüsseln, die gesuchte Früchte enthalten: 4 6 7 10 11 15 18 20
Schüsseln, die gesuchte Früchte enthalten könnten:
Mögliche Schüsseln für jede Frucht:
Clementine 7
Erdbeere 10
Himbeere 18
Orange 20
Quitte 4
Rosine 11 15
Ugli 11 15
Vogelbeere 6
Apfel 3
Banane 8
Feige 22
Ingwer 14
Johannisbeere 12

Kiwi 23
Litschi 13
Nektarine 2
Weintraube 21
Dattel 5
Pflaume 9
Tamarinde 19
Grapefruit 17
Mango 1
Sauerkirsche 16

Beispiel 7

Eingabe: spiesse7.txt

Ausgabe:

Ergebnisse für spiesse7.txt
Laufzeit: 0.00037739999999999996
Wahrscheinlichkeit, dass Schüssel i gesuchte Frucht enthält: 0 0 0.75 0 1 1 0 1 0 0.75 0 0 0 1 0 1 1 0.5 0 0.75 0 0 1 1 0.5 0.75
Schüsseln, die gesuchte Früchte enthalten: 5 6 8 14 16 17 23 24
Schüsseln, die gesuchte Früchte enthalten könnten: 3 10 20 26 18 25
Mögliche Schüsseln für jede Frucht:
Apfel 10 3 20 26
Clementine 24
Dattel 17 6 16
Grapefruit 10 3 20 26
Mango 17 6 16
Sauerkirsche 14 8
Tamarinde 5 23
Ugli 25 18
Vogelbeere 17 6 16
Xenia 10 3 20 26
Yuzu 14 8
Zitrone 5 23
Erdbeere 15
Feige 2 11 13 22
Himbeere 2 11 13 22
Ingwer 4
Kiwi 1
Litschi 10 3 20 26
Nektarine 7
Orange 2 11 13 22
Quitte 2 11 13 22
Banane 25 18
Pflaume 9 21
Weintraube 9 21
Johannisbeere 19 12
Rosine 19 12

Beispiel 8 – Interaktive Erweiterung

Eingabedatei: beispiel8.txt Die gegebenen Informationen erlauben mehrere Kombinationen

stdio:

Options: (r)ead file (q)uit r Please enter filename: beispiel8.txt Options: (g)et bowl suggestion (u)se bowl (e)xit g Try bowl number 1 (Probability: 0.5) Options: (g)et bowl suggestion (u)se bowl (e)xit u	Options: (r)ead file (q)uit r Please enter filename: beispiel8.txt Options: (g)et bowl suggestion (u)se bowl (e)xit g Try bowl number 1 (Probability: 0.5) Options: (g)et bowl suggestion (u)se bowl (e)xit u Which bowl did you use 1 Which fruit did you find?

Which bowl did you use 1 Which fruit did you find? Apfel All desired fruit types where found Options: (r)ead file (q)uit q	Pflaume Options: (g)et bowl suggestion (u)se bowl (e)xit g Try bowl number 2 (Probability: 1) Options: (g)et bowl suggestion (u)se bowl (e)xit u Which bowl did you use 2 Which fruit did you find? Apfel All desired fruit types where found Options: (r)ead file (q)uit q

Quellcode

```
def applyObservation(obs: Observation, pFruits: List[Set[Fruit]]):
    for bowl in aBowls(len(pFruits)):
        if bowl in obs.bowls:
            pFruits[bowl].intersection_update(obs.fruits)
        else:
            pFruits[bowl].difference_update(obs.fruits)


def getProbabilities(fruitWanted: Set[Fruit], pFruits: List[Set[Fruit]]) -> List[Union[int, float]]:
    containsWanted: List[Union[int, float]] = [-1 for _ in range(len(pFruits))]
    for bowl in aBowls(len(pFruits)):
        prob = len(pFruits[bowl].intersection(fruitWanted))/len(pFruits[bowl])
        prob = round(prob) if round(prob) == prob else prob
        containsWanted[bowl] = prob
    return containsWanted


def process_static(tInput: TaskInput) -> TaskOutput:
    sTime = timeit.default_timer()

    pFruits: List[Set[Fruit]] = [set(aFruits(tInput.F)) for _ in aBowls(tInput.F)] #Time: O(F^2) Space: O(F^2)

    for obs in tInput.observations: #Time: O(N*F*avg(len(obs.fruits)))
        applyObservation(obs, pFruits)

    pBowls: List[Set[Bowl]] = [set() for _ in aFruits(tInput.F)]
    for bowl in aBowls(tInput.F):  #Max Time: O(F^2) Max Space: O(F^2)
```

```
        for fruit in pFruits[bowl]:
            pBowls[fruit].add(bowl)

    containsWanted = getProbabilities(tInput.fruitWanted, pFruits) #Avg Time: O(F) Max Time: O(F^2)

    tOutput = TaskOutput(tInput)
    tOutput.containsWanted = containsWanted
    tOutput.addpBowls(pBowls)
    tOutput.addpFruits(pFruits)
    tOutput.runtime = timeit.default_timer()-sTime
    return tOutput


def main():
    for i in range(8):
        tInput = readInput(f"spiesse{i}.txt")
        print(f"Processing spiesse{i}.txt")
        tOutput = process_static(tInput)
        print(f"Took {tOutput.runtime}")
        writeOutput(f"ausgabe{i}.txt", tOutput, verbose=True, detailed=True)


if __name__ == "__main__":
    main()
```