Aufgabe 2: Spießgesellen?

Teilnahme-Id: 00175

Bearbeiter/-in dieser Aufgabe:
Sebastian Brunnert

15. Januar 2021

Inhaltsverzeichnis

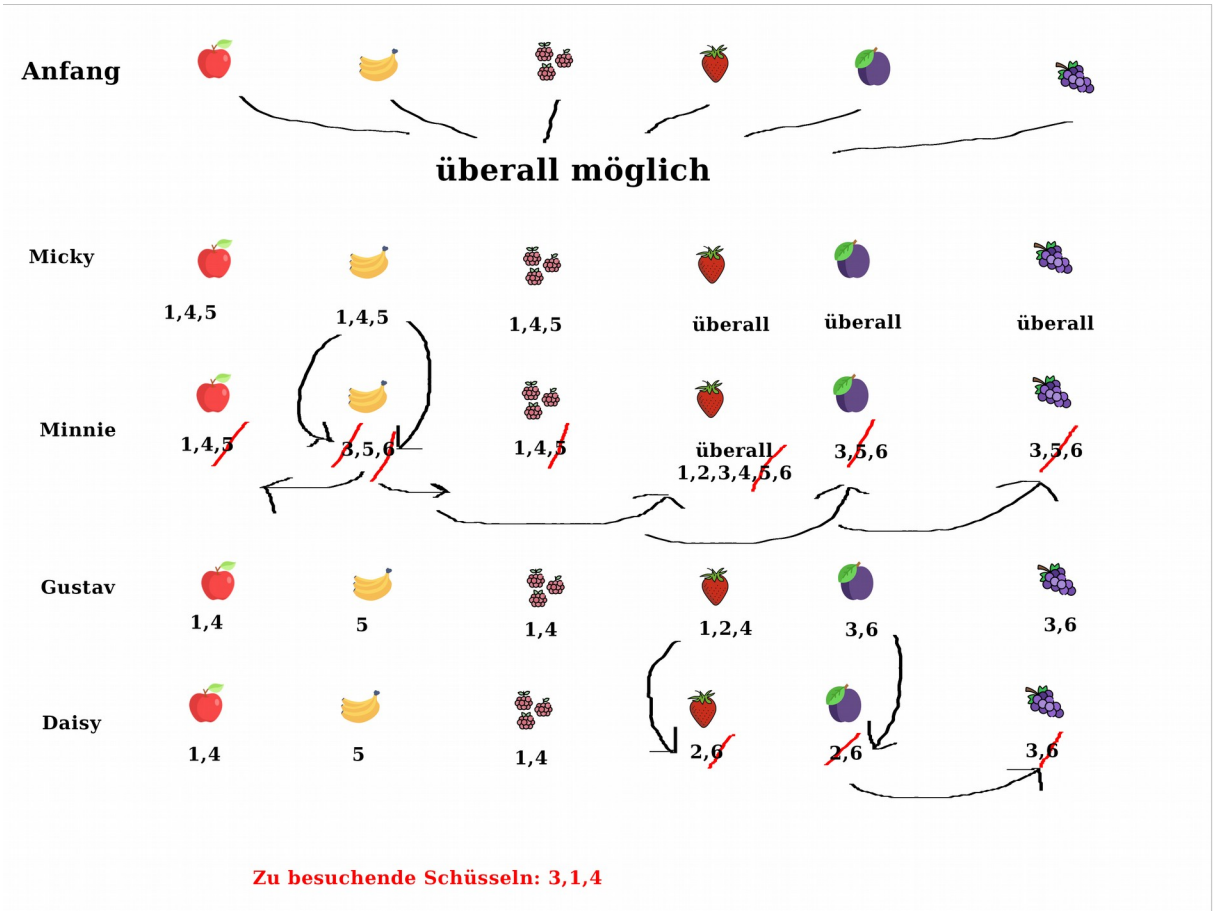
Lösungsidee

Wenn Donald gewisse Beobachtungen macht, so können gewisse Rückschlüsse auf die möglichen Verteilungen der Obstsorten in den Schüsseln gemacht werden. Zu Anfang macht Donald bspw. die Beobachtung, dass aus Warteschlange 1 und 2 die Obstsorten a und b resultieren. Resultieren nun aus Warteschlange 2 und 3 b c, so kann sicher gesagt werden, dass sich bei Warteschlange 2 das Obst b befindet, da dieses doppelt vorkommt – genau wie die Warteschlange 2. Grundlegend kann also gesagt werden, dass erst jene Warteschlangen gefunden werden müssen, die für jede Obstsorte in Betracht kommen. Dazu wird zu Anfang davon ausgegagen, dass jede Obstsorte in jeder Schüssel vorkommen kann. Anschließend werden die Beobachtungen ausgewertet: Das heißt es werden für jede Beobachtung alle Schüsseln, die nicht besucht wurden, für jede Obstsorte die aber auf dem Spieß ist, entfernt. Nun besitzt jede Obstsorte mögliche Schüsseln.

Zuletzt müssen also nur noch mittels eiens Algorithmus alle Schüssel-Möglichkeiten auf eine Obstsorte verteilt werden, sodass jede Schüssel mindestens eine mögliche Schüssel besitzt. Bei Obstsorten, die nur eine einzige Schüssel-Möglichkeit besitzen, ist klar, dass dies die Schüssel sein wird. Diese Möglichkeit darf also bei anderen Obstsorten nicht mehr verwendet werden. Existieren mehr als zwei Möglichkeiten bei einer Obstsorte, muss nur geprüft werden, ob eine Schüssel existiert, die alleinig diese Möglichkeit besitzt. Ist dies der Fall, so ist die gesuchte Schüssel diese Möglichkeit.

Donald kann nun einfach ablesen, welche Warteschlangen bzw. Schüsseln er besuchen muss.

Im gegebenen Beispiel wird dieses Verfahren genau so angewandt: Es wird für jede Obstsorte angelegt, welche möglich Schüsseln möglich sind. Anfangs sind dies alle. Nachdem Minnie seinen Spieß zusammengestellt hat, ist klar, dass die Banenen sich in der Schüssel 5 befinden, da sowohl Micky und Minnie sich an den Schüsseln 3,6 und eben der 5 bedienten und beide Bananen auf dem Spieß haben. Also können andere Obstsorten nicht mehr in der Schüssel 5 sein. Solche und andere Rückschlüsse lassen sich z.B. auch bei der Auswahl Daisys finden. Zusammenfassend kann gesagt werden, dass Donald die Schüsseln 1,3 sowie 4 besuchen muss für seinen Wunsch-Spieß:



Umsetzung

Die Umsetzung dieser Aufgabe geschieht in der Programmiersprache JavaScript auf NodeJS. Zu Anfang des Programmes muss simpel die gewünschte Datei eingelesen werden. In dem Array zeilen werden die sinnvoll formatierten Zeilen gespeichert.

Jetzt kann die Datenstruktur für den Algorithmus selber erstellt werden. Dazu baue ich ein Objekt obstsorten, in welches später als Key der Name der Obstsorte und als Value ein Array mit möglichen Schüssel dieser Obstsorte angegebn wird. Diese Datenstruktur wird nun gefüllt, in dem in Allen Zeilen (sowohl in Donalds Wünschen als auch in seinen Beobachtungen) nach Obstsorten gesucht wird. Aufgefüllt wird das Array nun mit diesen Keys und einem Array gefüllt mit Zahlen von 1 bis Anzahl der Schüsseln, da zu Anfang ja jede Schüssel in Betracht kommt.

Nun wird anhand der Spieße und der besuchten Schüsseln der Partybesucher jede Schüssel für jedes Obst entfernt mittels der praktischen .filter Methode aus dem Array für ein Obst entfernt, das nach den Beobachtungen zufolge defintiv nicht möglich ist. Dazu wird durch alle Beobachtungen iteriert und es werden aus dem Array möglicher Schüsseln nur jene übergelessen, die zuvor existent waren und auch bei dem aktuellen Spieß besucht wurden. Nun liegt also für jedes Obst eine Array an Schüssel-Indexen fest, in der sich möglicherweise das Obst befindet. Also muss jeder Obstsorte eine bestimmte Schüssel zugeordnet werden. Im Grund genommen, verwende ich dazu selbigen entwickelten Algorithmus wie in der ersten Aufgabe der ersten Runde. Dabei wird durch alle zu findenden Keys iterert und wie in der Lösungsidee beschrieben, wird geprüft, ob die Länge der Möglichkeiten für dieses Obst gleich 1 ist. Ist dies der Fall so wurde die Zuteilung für dieses Obst gefunden und diese Schüssel muss aus allen anderen Arrays an Möglichkeiten entfernt werden, was auch mittels eines for-Loops geschieht. Entspricht die Anzahl der Möglichkeiten allerdings nicht Eins, so wird durch die Möglichkeiten

der Schüsseln wieder durchgelaufen. Für jede Möglichkeit wird geprüft, ob andere Obstsorten diese Schüssel-Möglichkeit besitzen. Wenn dies bei keiner der Fall ist, muss diese die eindeutige Schüssel sein. Das wird auch so vermerkt. Ansich muss dieses Verfahren solange durchgeführt werden, bis keine Änderungen mehr möglich sind, weswegen der Boolean aktiv eingeführt wurde. Jetzt aber kann immer noch weiter spezifisiert werden, wo welche Obstsorte zu finden ist. Wenn nämlich zwei Obstsorten die möglichen Schüsseln 1 und 2 besitzen und zwei für zwei weitere Obstsorten 1,2,3 und 4 in Betracht kommen, kann sicher gesagt werden, dass die zwei weiteren Obstsorten in 3 und 4 zu finden sind, da ja sonst die ersten beiden Obstsorten nirgends zu finden sein wären. Dies wird implementiert, indem erst durch alle Obstsorten iteriert wird und der Integer count immer dann erhöht wird, wenn eine gefundene Obstsorte genau jene möglichen Schüsseln besitzt, wie die die Iterationsvariabel obstsorte. Wenn nun mehr als zwei und ausreichend viele Duplikate gefunden wurden, können von den übrigen Obstsorten diese Schüsselmöglichkeiten abgezogen werden.

Nachdem nun durch dieses Verfahren Schüssel-Möglichkeiten gufunden (bei den Beispielen existieren oftmals mehrere Möglichkeiten, diese heben sich aber gegenseitig auf), kann ermittelt werden, welche Schlangen Donald besuchen muss. Dazu werden alle Schüssel-Möglichkeiten für jede Obstsorte, die sich Donald wünscht, in das Array warteschlangen gegeben. Dieses wird schlussendlich nun ausgegeben. Duplikate werden mittels eines Sets entfernt.

Beispiele

node spiessgesellen.js spiesse1.txt	[1, 2, 4, 7, 5]
node spiessgesellen.js spiesse2.txt	[1, 5, 10, 11, 6, 7]
node spiessgesellen.js spiesse3.txt	Anhand der Daten kann nicht eindeutig gesagt werden, welche Schüsseln besucht werden müssen. Aber unter diesen Schüssel befinden sich die gewünschten Obstsorten: [5, 8, 2, 7, 10, 11, 1, 12]
node spiessgesellen.js spiesse4.txt	[9, 13, 8, 6, 2, 7, 14, 12]
node spiessgesellen.js spiesse5.txt	[1, 2, 3, 4, 5, 6, 9, 10, 12, 16, 19, 20, 14]
node spiessgesellen.js spiesse6.txt	[7, 10, 18, 20, 4, 11, 15, 6]
node spiessgesellen.js spiesse7.txt	Anhand der Daten kann nicht eindeutig gesagt werden, welche Schüsseln besucht werden müssen. Aber unter diesen Schüssel befinden sich die gewünschten Obstsorten: [3, 10, 20, 26, 24, 4, 5, 6, 7, 16, 17, 23, 8, 14, 18, 25]

Quellcode

Der vollständige Quellcode, ist in der Datei spiessgesellen.js zu finden.

```
// Ein Objekt wird erstellt. In diesem:
// - Gibt es für jede Obstsorte einen Key
// - Value ist dann jeweils jede mögliche Schüssel
var obstsorten = {};

// Nun wird die beschriebene Datenstruktur angelegt
// Dazu werden überall wo möglich Obstsorten gesucht, die angeboten werden
zeilen[1].split(" ").forEach(e => {
   // Zu Anfang ist jede Schüssel möglich, also wird ein Array als Value angegebn mit Elementen von 1-Anzahl Schüsseln
   obstsorten[e] = Array.from({length: parseInt(zeilen[0])}, (_, i) => i + 1);
});
// Hier (in den Resultaten der Partyteilnehmer) wird ebenfalls gesucht
for(var i = 4; i < zeilen.length; i+=2) {
   zeilen[i].split(" ").forEach(e => {
      obstsorten[e] = Array.from({length: parseInt(zeilen[0])}, (_, i) => i + 1);
   });
}

// Nun wird anhand der Ergebnisse und der besuchten Schüsseln der Partybesucher jede Schüssel für jedes
// Obst entfernt, die nach den Beobachtungen definitiv nicht möglich ist entfernt.
for(var i = 4; i < zeilen.length; i+=2) {
   zeilen[i].split(" ").forEach(e => {
      obstsorten[e] = obstsorten[e].filter(e => zeilen[i-1].split(" ").map(x=>+x).includes(e))
   });
}

// Nun müssen alle möglichen Schüsseln zu einer Obstsorte zugeordnet werden mittels eines Algorithmus
// Dazu wird jede Obstsorte iteriert, um für diese die die Schüssel zu finden
```

```
    var aktiv = true;
    while(aktiv) {
      aktiv = false;
      for(obstsorte in obstsorten) {
        // Obstsorten, bei denen nur eine Schüssel möglich ist, wird diese zugeteilt
        if(obstsorten[obstsorte].length == 1) {
          for(andereObstsorte in obstsorten) {
            if(andereObstsorte != obstsorte && obstsorten[andereObstsorte].includes(obstsorten[obstsorte]
[0])) {
              // Allen anderen wird diese Schüssel entzogen

obstsorten[andereObstsorte].splice(obstsorten[andereObstsorte].indexOf(obstsorten[obstsorte][0]),1);
              aktiv = true;
            }
          }
        } else {
          // Andernfalls wird geprüft, ob dieses Obst eine Schüssel besitzt, die bei keinem anderen Obst in
Betracht
          // kommt. Dann nämlich ist klar, dass dieses Obst in dieser Schüssel ist.
          // Dazu werden alle Möglichkeiten einer Obstsorte iteriert
          obstsorten[obstsorte].forEach(moeglichkeit => {
            var moeglich = true;
            // Dazu werden alle anderen Obstsorten iteriert
            for(andereObstsorte in obstsorten) {
              if(andereObstsorte != obstsorte && obstsorten[andereObstsorte].includes(moeglichkeit)) {
                // Wenn diese Obstsorte diese Schüssel hat, ist sie nicht mehr möglich
                moeglich = false;
              }
            }

            // Falls sie möglich ist, wird das auch so gekennzeichnet
            if(moeglich) {
              obstsorten[obstsorte] = [moeglichkeit];
              aktiv = true;
            }
```

```
            });
        }
    }
}

// Duplikate werden gesucht
for(obstsorte in obstsorten) {
    var count = 0;

    for(andereObstsorte in obstsorten) {
        if(obstsorten[andereObstsorte].equals(obstsorten[obstsorte])) {
            // Bei gefundem Duplikat counter um 1 erhöhen
            count++;
        }
    }

    // Wenn ausreichen viele Duplikate gefunden wurden, können bei anderen die Schüsseln entfernt werden
    if(count >= 2 && count >= obstsorten[obstsorte].length) {
        for(andereObstsorte in obstsorten) {
            if(!obstsorten[andereObstsorte].equals(obstsorten[obstsorte])) {
                obstsorten[andereObstsorte] = obstsorten[andereObstsorte].filter(ele => !
obstsorten[obstsorte].includes(ele));
            }
        }
    }
}


// Nun muss nur noch ermittelt werden, welche Warteschlangen besucht werden müssen
warteschlangen = [];
// Dazu wird durch alle Wünsche iteriert ...
zeilen[1].split(" ").forEach(wunsch => {
    // ... und alle möglichen Schüsseln einer Obstsorte (es kann vorkommen, dass das mehr als eine sind,
```

```
    // hebt sich aber bei den Beispielen auf) müssen zu ener Liste mit zu besuchenden Warteschlangen hinzugefügt werden
    warteschlangen.push(...obstsorten[wunsch]);
  });
  // Duplikate werden entfernt und das Ergebnis wird ausgegeben
  warteschlangen = [...new Set(warteschlangen)];
  if(warteschlangen.length > zeilen[1].split(" ").length) {
    console.log("Anhand der Daten kann nicht eindeutig gesagt werden, welche Schüsseln besucht werden müssen. Aber unter diesen Schüssel befinden sich die gewünschten Obstsorten: ")
  }
  console.log(warteschlangen);
```