Aufgabe 4: Zara Zackigs Zurückkehr

Bearbeiter/-in dieser Aufgabe:
Joshua Benning

26. August 2022

Inhaltsverzeichnis

Lösungsidee

Teilaufgabe A

Der triviale Ansatz ist es, alle möglichen Teilmengen an Karten zu generieren, und zu überprüfen, ob diese XOR-Verknüpft an allen 128 Bit 0 sind. Bei 111 Karten gibt es $\binom{111}{11}$ beziehungsweise $4.7324 * 10^{14}$ mögliche Teilmengen der Länge elf. Das ist keine Menge die durch Brute-Force in sinnvoller Zeit abzuarbeiten ist. Deswegen muss der Suchraum eingeschränkt werden.

Im Weiteren wird je nur ein Bit der Karten betrachtet. Um am Ende durch XOR auf 0 zu kommen, muss die Anzahl der Einser gerade sein. Das heißt, es gibt nur die Möglichkeit für 11*0, 9*0 und 2*1, 7*0 und 4*1, 5*0 und 6*1, 3*0 und 8*1 oder 1*0 und 10*1. Angenommen die Verteilung der 0 und 1 an einer Bitstelle ist in etwa ausgeglichen, dann gibt es also ungefähr 55*0 und 55*1. Damit gibt es nur $\binom{55}{11}+\binom{55}{9}*\binom{55}{2}+\binom{55}{7}*\binom{55}{4}+\binom{55}{5}*\binom{55}{6}+\binom{55}{3}*\binom{55}{8}+\binom{55}{1}*\binom{55}{10}$ zu betrachtende Kombinationen.

Umgerechnet entspricht das $2.1317107556355 * 10^{14}$, also einer Verbesserung um Faktor 2. Der Suchraum lässt sich weiter einschränken, wenn man die Information, dass die XOR Verknüpfung 0 sein muss, auf weitere Bit-Stellen ausweitet. Darüber hinaus kann der Suchraum trivialerweise eingeschränkt werden.

Im Detail wird folgendermaßen vorgegangen: Die 111 Karten werden an der BitStelle 0 nach 0ern und 1ern sortiert und in die Mengen M_0 und M_1 aufgeteilt. M_0 beziehungsweise M_1 haben die Größe 55 bzw 56 (111 : 2) wenn die 0er und 1er gleich verteilt sind, andernfalls verschieben sich die Zahlen etwas. Im Weiteren wird nun der Fall einer Gleichverteilung betrachtet. Die Stelle i kann so gewählt werden, dass 0er und 1er möglichst gleich verteilt sind. Weil 11 Karten gesucht sind, müssen in M_0 oder M_1 >= 6 der gesuchten Karten sein.

Jetzt werden zwei Karten aus M_0 gewählt und aus diesen entfernt (analog für M_1). Die zwei Mengen werden nun an der Bit-Stelle 1 nochmals jeweils nach 0ern und 1ern sortiert und aufgeteilt in M_{00}, M_{01}, M_{10} und M_{11}. Die haben jeweils die Größe 27 (55 : 2). Das heißt, es gibt nun 4 Mengen, und in mindestens einer der Mengen müssen >= 4 der gesuchten Karten sein. Das Problem kann also in die 4 Mengen unterteilt werden. Für jede der Mengen M_{00}, M_{01}, M_{10} und M_{11} wird des Weiteren wie folgt vorgegangen.

Zunächst werden 2 Karten aus der Menge gewählt und mit den bereits gewählten 2 Karten aus M_0 bzw M_1 XOR verknüpft. Zu den 4 Karten werden im nächsten Schritt passende 7 weitere Karten nach folgendem Verfahren gesucht:

Es wird in der neu gebildeten XOR Karte eine noch nicht verwendete Bit-Stelle mit einer 0 gesucht, im Folgenden i genannt. Das heißt die restlichen sieben Karten, müssen entweder 7*0, 5*0 und 2*1, 3*0 und 4*1 oder 1*0 und 6*1 haben. Die Karten werden nach dem Bit an dieser Bit-Stelle i sortiert und getrennt. Die zwei Mengen haben nun in etwa die Größe 13 (= 27/2). Nun wird eine weitere Bit-Stelle j betrachtet, an der ebenfalls die Verknüpfung der 4 Karten via XOR 0 ist. Für diese müssen die eben genannten Bedingungen ebenfalls erfüllt sein.

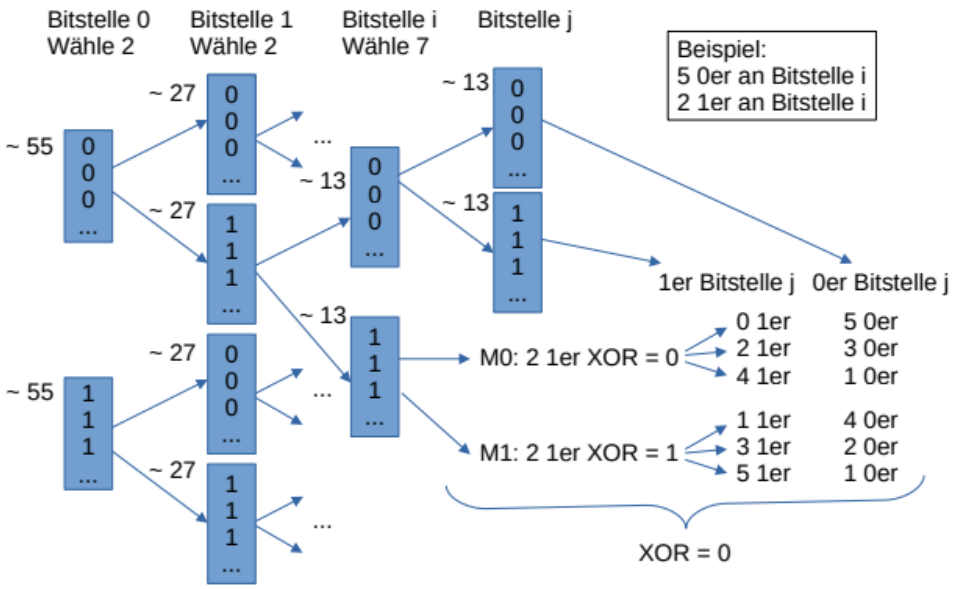

Abbildung 1: Skizze des Algorithmus; Beispiel: Wähle 5-0er und 2-1er

Die 27 Karten haben an i den Wert 0. An der Stelle j gibt es etwa 13-0er und 13-1er. Im ersten Fall werden 7-0er an der Bit-Stelle i gewählt. Nun muss die XOR Verknüpfung an der Stelle j auch 0 sein. Das bedeutet es ist möglich die verschiedenen Fälle auf die bereits oben Genannten zu

reduzieren (Also 7*0, 5*0 und 2*1, 3*0 und 4*1 oder 1*0 und 6*1 an der Stelle j). Im zweiten Fall werden am Index i nicht nur 0er, sondern auch 1er gewählt, nämlich 5*0 und 2*1. Die zwei 1er an der Stelle i XOR sind 0. An der Stelle j können sie XOR 0 oder 1 sein. Abhängig davon, ob 0 oder 1, müssen am Index j die anderen Karten (aus den 0ern bei i) ebenfalls XOR 0 oder 1 sein, um insgesamt auf XOR 0 zu kommen. Für die Prüfung der sieben Karten mit den vier vorher gewählten Karten, werden dann entsprechend nur die Kombinationen geprüft, die sowohl an Index i, als auch an Index j XOR auf 0 kommen. Die Fälle 3 und 4 sind analog. Es gilt demnach folgende Fälle zu unterscheiden:

- 7*0 bei i
  - 7*0 bei j
  - 5*0 2*1 bei j
  - 3*0 4*1 bei j
  - 1*0 6*1 bei j
- 5*0 2*1 bei i
  - XOR bei 2*1 bei j = 0
    - 5*0 bei j
    - 3*0 2*1 bei j
    - 1*0 4*1 bei j
  - XOR der 2*1 bei j = 1
    - 4*0 1*1 bei j
    - 2*0 3*1 bei j
    - 0*0 5*1 bei j
- 3*0 4*1 bei i
  - XOR der 4*1 bei j = 0
    - 3*0 bei j
    - 1*0 2*1 bei j
  - XOR der 4*1 bei j = 1
    - 2*0 1*1 bei j
    - 0*0 3*1 bei j
- 1*0 6*1 bei i
  - XOR der 6*1 bei j = 0
    - 1*0 bei j

- XOR der 6*1 bei j = 1
  - 1*1 bei j

Durch Anwenden des Algorithmus ergibt sich, unter der Annahme, dass 0er und 1er in etwa gleich verteilt sind, folgender Suchraum:

$$
\begin{aligned}
&\binom{55}{2} * \binom{27}{2} * 2 * \frac{1}{24} \\
&[\binom{13}{7} + \binom{13}{5} * \binom{7}{2} + \binom{13}{3} * \binom{7}{4} + \binom{13}{1} * \binom{7}{6}] \\
&+[\binom{13}{5} + \binom{13}{3} * \binom{7}{2} + \binom{13}{1} * \binom{7}{4} + \binom{13}{4} * \binom{7}{1} \\
&+\binom{13}{2} * \binom{7}{3} + \binom{7}{5}] * \binom{27}{2} \\
&+[\binom{13}{3} + \binom{13}{1} * \binom{7}{2} + \binom{13}{2} * \binom{7}{1} + \binom{7}{3}] * \binom{27}{4} \\
&+[\binom{13}{1}] * \binom{27}{6}] \\
&= 43436.25 * (38844 + 5441904 + 20007000 + 4144140) \\
&= 1.287090687168 \times 10^{12}
\end{aligned}
$$

Abbildung 2: Berechnung Suchraum

Trotz der Einschränkung des Suchraums um ungefähr Faktor 367 konnte das Programm keine Lösung für Beispiel "stapel2.txt" finden.

Teilaufgabe B

Nachdem die Schlüsselkarten, bevor sie den Häusern zugewiesen wurden, aufsteigend der Größe nach, sortiert wurden, gibt es aus den k + 1 zur Auswahl stehenden Karten nur je 2 Optionen für jedes Haus. Zunächst werden die k + 1 Karten, aufsteigend der Größe nach, sortiert. Nun kommt für das Haus an der Position k nur die Karte mit Index k oder k + 1 in Frage:

- Wenn die Sicherungskarte vor Index k liegt, dann ist die gesuchte Schlüsselkarte am Index k + 1
- Wenn die Sicherungskarte am Index k liegt, dann ist die gesuchte Schlüsselkarte am Index k + 1
- Wenn die Sicherungskarte nach Index k liegt, dann ist die gesuchte Schlüsselkarte nach wie vor am Index k

Nachdem es nur diese 3 Fälle, beziehungsweise daraus resultierende Indizes für die passende Schlüsselkarte, gibt, wird, beim Anwenden der eben beschriebenen Überlegungen, nur maximal 1 Fehlversuch benötigt, um die richtige Schlüsselkarte zu identifizieren. Der Index ist entweder k oder k + 1.

Umsetzung

Die Implementation erfolgt in Java. Zunächst wird das File eingelesen und die Karten in Objekte überführt. Jeder Karte enthält nun das Bit-Array, dargestellt als boolean-Array und zudem noch eine ID, zur eindeutigen Identikation. Ein nennenswerter Algorithmus ist die modizierte rekursive Breitensuche zur Bildung von Teilmengen der Größe k:

```
private <T> void getSubsets(T[] superSet, int k, int index,
            int currentIndex, T[] current, List<T[]> solution) {
    if (currentIndex == k) {
        solution.add(current);
        return;
    }
    if (index == superSet.length) return;
    T x = superSet[index];
    T[] copy = Arrays.copyOf(current, k);
    copy[currentIndex] = x;
    getSubsets(superSet, k, index + 1, currentIndex, current, solution);
    getSubsets(superSet, k, index + 1, currentIndex + 1, copy, solution);
}
```

Programmauszug 1: Suche von Teilmengen

Interessant ist zudem das Suchen der passenden sieben Karten an der Bit-Stelle i und j, wie auch in der Abbildung1 veranschaulicht. Die Implementierung reduziert sich auf eine einfache for Schleife, welche mittels Modulo Operator bestimmt welche Teilmenge betrachtet werden muss:

```
for (int i = 0; i < zeros; i++) {
    if (i % 2 == 0) {
        checkForResult(zeroZero, zeros, i, zeroOne, ones, xor, chosen, indexModZero);
    } else {
        checkForResult(zeroZero, zeros, i, zeroOne, ones, xor, chosen, indexModOne);
    }
}
```

Programmauszug 2: Teilmengen Betrachtungen

Um Arbeitsspeicher zu sparen, werden, sobald sieben Karten ausgewählt wurden, diese mittels XOR geprüft, ob sie die Lösung ermittelt wurde. Wenn nicht, werden sie verworfen. Zur Laufzeitverbesserung wurde der Suchraum unterteilt und auf verschiedene Threads parallelisiert.

Beispiele

Beispiel 0

Schlüsselkarten + Sicherungskarte: 2, 3, 5, 9, 11

Beispiel 1

Schlüsselkarten + Sicherungskarte: 1, 2, 4, 6, 7, 9, 11, 14, 15

Beispiel 5

Schlüsselkarten + Sicherungskarte: 70, 77, 163, 167, 185

Quellcode

Der Quellcode befasst sich hauptsächlich mit der Parallelisierung und dient deshalb nur schwer zur Veranschaulichung des Lösungsverfahren. Des weiteren wurde der Quellcode beim Versuch der weiteren Suchraumeinschränkung von der hier beschriebenen Form abgebracht und eine alte Version ist leider nicht mehr verfügbar.

Quellen / Material

Aufgabestellung: https://bwinf.de/bundeswettbewerb/40/2/ [01.09.2022 ~ 18:52 Uhr]

Beispieldaten: https://bwinf.de/bundeswettbewerb/40/2/ [01.09.2022 ~ 18:52 Uhr]