

Aufgabe 5: Marktwaaage

Team-ID: 00741

Team-Name: LouisKunze

Bearbeiter/-innen dieser Aufgabe:
Louis Schell

22. November 2021

Inhaltsverzeichnis

1	Aufgabenstellung	1
1.1	Vorraussetzungen	1
2	Lösungsidee	2
2.1	Der Algorithmus	3
2.2	Laufzeitanalyse	3
3	Umsetzung	3
3.1	Multipel generieren	4
3.2	Permutationen der Multipel generieren	4
4	Beispiele	5
4.1	Beispiel 0	5
4.2	Beispiel 1	7
4.3	Beispiel 2	9
4.4	Beispiel 3	10
4.5	Beispiel 4	12
4.6	Beispiel 5	14
5	Quellcode	17

1 Aufgabenstellung

Gegeben ist eine Balkenwaage mit zwei Seiten L und R , und eine Menge an Gewichten G . Ein Gewicht $w \in \mathbb{G}$ kann man auf eine der beiden Seiten, oder gar nicht auf die Waage legen. Mit verschiedenen Kombinationen an Gewichten lassen sich sehr viele verschiedene Massen wiegen. Von der Aufgabenstellung gesucht sind genau die Kombinationen, mit der man Massen im Intervall $I_0 = [0g, 500g]$ und $I_1 = [9500g, 10000g]$ wiegen kann. Kann die Masse nicht genau mit einer Kombination der vorhandenen Gewichte gewogen werden, so soll die Kombination von Gewichten die zur ähnlichsten Masse führt ausgegeben werden.

1.1 Vorraussetzungen

Sei die zu wiegende Masse auf der linken Seite platziert. Definiere m als die Anzahl der vorkommenden, verschieden Gewichte und n als die . Sei w_i die Masse des Gewichts und n_i die Anzahl der Gewichte mit dieser Masse. Eine Lösung ist gefunden, wenn die Differenz der Summe der Gewichte auf der rechten Seite und die Summe der Gewichte auf der linken Seite der Waage gleich der gesuchten Masse M_{ges} ist.

$$(1) \quad \sum_{j=0} L[j] + M_{ges} = \sum_{i=0} R[i]$$

$$(2) \quad \Leftrightarrow M_{ges} = \sum_{i=0} R[i] - \sum_{j=0} L[j]$$

Nur die Differenz der beiden Summen ist ausschlaggebend. Ist die Differenz negativ kann die rechte Seite einfach mit der linken Seite getauscht werden. Dies ist analog dazu, die zu wiegende Masse einfach auf die andere Seite zu legen.

$$(3) \quad \Leftrightarrow M_{ges} = \left| \sum_{i=0} R[i] - \sum_{j=0} L[j] \right|$$

M_{ges} kann also auch als ein Array geschrieben werden, statt zwei Arrays zu verwenden. Dazu negiert man die Gewichte aus dem einem Array einfach.

2 Lösungsidee

Die obenstehenden Gleichungen geben an, wann eine Lösung gefunden ist. Das Ziel ist es, rückwärts zu arbeiten, und eine Permutation von Gewichten zu finden, die (2) erfüllt. Wenn es um Permutationen geht, müssen diese erstmal generiert werden. Dazu verwendet man häufig Brute-Force Methoden, speziell weil Probleme vergleichbar mit diesem oft NP-Schwer sind und vermutlich keine andere einfache Lösung haben¹. Ein Problem auf das spätestens irgendwann alle Programmierer stoßen werden ist, dass exponentiell zunehmende Probleme sehr schnell in Komplexität zunehmen. Das heißt, ihre Berechnungszeit nimmt drastisch zu. So auch in unserem Problem.

Betrachte man die Fälle die für ein Gewicht eintreten können. Das Gewicht:

- liegt auf der rechten Seite und beeinflusst die gesamte Summe negativ,
- liegt auf der linken Seite und beeinflusst die gesamte Summe positiv, oder
- ist nicht auf der Waage vorhanden und beeinflusst die gesamte Summe nicht.

Das sind insgesamt 3 Zustände je Gewicht. Aus der Lehre der Kombinatorik folgt daraus für die Anzahl² der Permutationen $P(n) = 3^n$. Das schwierigste Beispiel stellt somit *gewichtsstuecke5.txt* dar, weil dort insgesamt 23 Gewichte zugeordnet werden können. Für $n = 23$ ergibt sich eine Anzahl an Permutationen von $P(23) \approx 94 \cdot 10^9$. Eine solche Lösung ist nicht praktikabel, da es Stunden dauern könnte so viele Permutationen zu errechnen und zu prüfen.

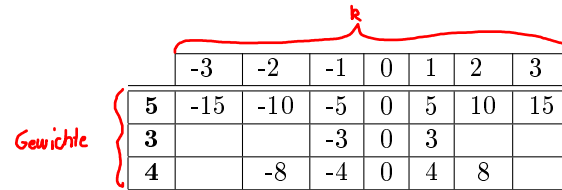
Eine effizientere Lösung erfolgt stattdessen über einen Algorithmus, der ausnutzt dass es in den Beispielen Gewichte gibt, die mehrfach vorkommen. Dadurch entstehen nämlich Permutationen die gleich sind. Doch wie filtert man diese Kombinationen raus? Eine Möglichkeit ist es, die Gewichte die mehrmals vorkommen nicht als einzelne, alleinstehende Gewichte anzusehen, sondern diese zusammenzufassen, und stattdessen als Multipl. des Grundgewichts zu sehen. z.B. Statt zwei Gewichte 5 und 5 zu haben, könnte man diese als Multipl. von 5, bis $k = 2$ darstellen. Dann hätten wir $[-10, -5, 0, 5, 10]$, statt $[-5, 0, 5]$ und $[-5, 0, 5]$. Wo wir vorher $3^2 = 9$ mögliche Permutationen hatte, haben wir jetzt nur noch $2 \cdot 2 + 1 = 5$ Permutationen, trotzdem sind alle Kombinationen abgedeckt. Überträgt man diese Idee jetzt auf mehrere verschiedene Gewichte, so hält sie weiterhin stand. Daraus folgt jetzt der Algorithmus in 2.1.

¹Siehe <https://de.wikipedia.org/wiki/Teilsummenproblem>, welches ähnlich wie dieses Problem ist

²Angemerkt sei, dass hier gleiche Permutationen nicht herausgerechnet wurden, da dies algorithmisch sowieso nicht direkt umsetzbar ist

2.1 Der Algorithmus

1. Erzeuge zu Beginn alle möglichen Vielfache jedes unterschiedlichen Gewichts w_i , mit $k \in \mathbb{N}$ von $-n_i \cdot w_i$ bis $+n_i \cdot w_i$. Die Spalte ganz links in der Tabelle gibt an, welches Gewicht w_i genutzt wurde.



	-3	-2	-1	0	1	2	3
5	-15	-10	-5	0	5	10	15
3			-3	0	3		
4		-8	-4	0	4	8	

Tabelle 1: Beispiel

2. Erzeuge alle möglichen Summen der Multipl, mit jeweils einer Zahl aus jeder Zeile, und speichere diese zusammen mit der Kombination ab, sofern sie innerhalb des Intervalls I_0 oder I_1 liegen. Man kann sich vorstellen, man würde mit dem Finger einen Pfad ziehen, von oben in der Tabelle nach unten, und dabei die Elemente summieren. Fährt man jeden Pfad ab, so hat man jede Permutation erstellt.
3. Aus den generierten Kombinationen, erschließe für jede Kombination und für jedes Multipl in der Kombination, mit welchem k das Multipl aus der jeweiligen Zeile generiert wurde. Somit können wir dann sagen, wie viele Gewichte mit der Masse w_i auf welcher Seite liegen. Z.B. wenn wir in einer Kombination das Multipl -15 haben, und wir wissen, dass $w_i = 5$, dann ist $k = \frac{-15}{5} = -3$. Ist k negativ, so liegen $|n|$ Gewichte mit der Masse w_i nach (2) auf der rechten Seite. Ist k positiv, so liegen $|k|$ Gewichte der Masse w_i auf der linken Seite. Ist $k = 0$, so liegen keine Gewichte mit der Masse w_i auf der Waage.

2.2 Laufzeitanalyse

Die genaue Laufzeit hängt stark von der Anzahl an gleichen Gewichten ab. Für die genaue Anzahl an Permutationen gilt:

$$\mathcal{O}\left(\prod_{i=0}^m (2 \cdot n_i + 1)\right)$$

Im Best-case gibt es von jedem Gewicht nur ein Stück. Aus der oben stehenden Formel folgt dann:

$$\mathcal{O}(3^m)$$

Im Worst-case gibt es von jedem Gewicht genau z Stücke. Dann folgt für die Anzahl an Permutationen:

$$\mathcal{O}((2z + 1)^m)$$

Aber warum ist der Algorithmus überhaupt schneller, als einfach alle Permutationen durchzurechnen? Mit der in 2.2 beschriebenen Methode lassen sich viele Kombinationen, die zur gleichen Masse führen wegstreichen. Durch das Gruppieren von gleichen Gewichten, kann man den außerdem den Exponenten, n deutlich verkleinern (z.B. in *gewichtsstuecke5.txt* von 23 auf 13 Gewichte). Der Exponent ist am ausschlaggebendsten, wenn es zur Laufzeit kommt.

3 Umsetzung

Die Umsetzung der Lösungsidee erfolgt in der Programmiersprache Rust. Rust ist eine multiparadigmen-Systemprogrammiersprache, die Geschwindigkeiten ähnlich wie die von C erreicht. Das Programm zur Umsetzung des Algorithmus wird in mehrere Teile gegliedert:

1. Gewichte auslesen
2. Multipl generieren
3. Alle Permutationen und die dazugehörige Summe der Multipl generieren
4. Aus den Kombinationen die Verteilung der Gewichte schließen
5. Ausgabe schreiben

3.1 Multipel generieren

Für den Algorithmus müssen erstmal die Multipel generiert werden. Nach 2.2 müssen wir alle Vielfache des Gewichtes für jedes unterschiedliche Gewicht generieren. Die Vielfache werden in ein 2D-Array *arr* geschrieben.

3.2 Permutationen der Multipel generieren

Wenn man an die Tabelle aus 2.2 denkt, ist der erste Gedanke, mit geschachtelten For-Schleifen zu arbeiten, um Permutationen aus *arr* zu generieren. Gegeben ist das 2D-Array *arr*, welches für jedes unterschiedliche Gewicht, ein Array mit allen Multipeln des Gewichts speichert.

Algorithmus 1 : Permutationen generieren - iterativ, nicht-dynamisch

```

1 for i = 0 To arr[0].len do
2   for j = 0 To arr[1].len do
3     for k = 0 To arr[2].len do
4       kombination ← summe der Kombination;
5       if summe ≤ max and summe ≥ min then
6         speichere die Summe und die Kombination ab;
7       end
8     end
9   end
10 end
```

Dieser pseudo-code berechnet alle Permutationen für das *Beispiel 1* aus 2.2 aus. Damit könnte man zwar alle Permutationen erstellen, allerdings ist der Code nicht dynamisch aufgebaut. Für jedes weitere, unterschiedliche Gewicht müsste man eine weitere geschachtelte For-Schleife erstellen. Es ist also unverzichtbar, hier einen Rekursiven Algorithmus zu verwenden. Der Rekursive Algorithmus ist wie folgt aufgebaut, und ist das Ebenbild der obenstehenden Funktion.

Algorithmus 2 : Permutationen generieren - rekursiv

input : Die aktuelle Tiefe *n* und die Kombination
output : Speichert alle möglichen Summen und ihre Kombinationen

```

1 fn rekursiere(n:int, kombination:[int]):
2   if n = arr.len then
3     summe ← summe der kombination;
4     if summe ≤ max and summe ≥ min then
5       speichere die Summe und die Kombination ab;
6     end
7   else
8     for i < arr[n].len do
9       aktualisiere die Kombination;
10      rekursiere mit n+1 und der neuen Kombination;
11    end
12  end
```

4 Beispiele

Die Lösungen der Beispiele sind wie folgend zu lesen. Die Masse bezeichnet die gesuchte Masse die durch eine Kombination aus Gewichten gemessen werden soll. Die Masse liegt auf der linken Seite der Waage. M_{gerundet} bezeichnet die Masse, die tatsächlich durch die Kombination in der gleichen Zeile erzeugt werden kann. Ist für M_{gerundet} ein Wert gegeben, existiert keine Kombination aus Gewichten, mit der die gesuchte Masse exakt gewogen werden kann. Angegeben ist die Kombination, die am nächsten kommt. Ist hingegen kein Wert angegeben, so kann die Masse durch die nebenstehende Kombination genau gewogen werden. Die dritte und vierte Spalte der Tabelle geben an, auf welcher Seite der Waage sich welche Gewichte befinden müssen. Der Index im Array korrespondiert in der Legende die am Anfange jedes Beispiels steht zu einem Gewicht. Die Zahl im Array an dem Index gibt an, wie viele Gewichte der jeweiligen Masse auf der Seite zu platzieren sind.³

4.1 Beispiel 0

gewichtsstuecke0.txt enthält insgesamt 15 Gewichte, davon 6 unterschiedliche. Der Algorithmus prüft insgesamt 36015 Permutationen, und bearbeitet diese in etwa 2ms.

Index	0	1	2	3	4	5
Gewicht	10g	50g	100g	500g	1000g	5000g

Kombinationen			
Masse	M_{gerundet}	links	rechts
10g		[0, 0, 1, 0, 1, 0]	[1, 2, 0, 2, 0, 0]
20g		[0, 0, 1, 0, 1, 0]	[2, 2, 0, 2, 0, 0]
30g		[0, 0, 1, 0, 1, 0]	[3, 2, 0, 2, 0, 0]
40g		[1, 0, 0, 0, 1, 0]	[0, 1, 0, 2, 0, 0]
50g		[0, 0, 0, 0, 1, 0]	[0, 1, 0, 2, 0, 0]
60g		[0, 0, 0, 0, 1, 0]	[1, 1, 0, 2, 0, 0]
70g		[0, 0, 0, 0, 1, 0]	[2, 1, 0, 2, 0, 0]
80g		[0, 0, 0, 0, 1, 0]	[3, 1, 0, 2, 0, 0]
90g		[1, 0, 0, 0, 1, 0]	[0, 2, 0, 2, 0, 0]
100g		[0, 0, 0, 0, 1, 0]	[0, 2, 0, 2, 0, 0]
110g		[0, 0, 0, 0, 1, 0]	[1, 2, 0, 2, 0, 0]
120g		[0, 0, 0, 0, 1, 0]	[2, 2, 0, 2, 0, 0]
130g		[0, 0, 0, 0, 1, 0]	[3, 2, 0, 2, 0, 0]
140g		[1, 0, 0, 0, 1, 0]	[0, 1, 1, 2, 0, 0]
150g		[0, 0, 0, 0, 1, 0]	[0, 1, 1, 2, 0, 0]
160g		[0, 0, 0, 0, 1, 0]	[1, 1, 1, 2, 0, 0]
170g		[0, 0, 0, 0, 1, 0]	[2, 1, 1, 2, 0, 0]
180g		[0, 0, 0, 0, 1, 0]	[3, 1, 1, 2, 0, 0]
190g		[1, 0, 0, 0, 1, 0]	[0, 2, 1, 2, 0, 0]
200g		[0, 0, 0, 0, 1, 0]	[0, 2, 1, 2, 0, 0]
210g		[0, 0, 0, 0, 1, 0]	[1, 2, 1, 2, 0, 0]
220g		[0, 0, 0, 0, 1, 0]	[2, 2, 1, 2, 0, 0]
230g		[0, 0, 0, 0, 1, 0]	[3, 2, 1, 2, 0, 0]
240g		[1, 0, 0, 0, 1, 0]	[0, 1, 2, 2, 0, 0]
250g		[0, 0, 0, 0, 1, 0]	[0, 1, 2, 2, 0, 0]
260g		[0, 0, 0, 0, 1, 0]	[1, 1, 2, 2, 0, 0]
270g		[0, 0, 0, 0, 1, 0]	[2, 1, 2, 2, 0, 0]
280g		[0, 0, 0, 0, 1, 0]	[3, 1, 2, 2, 0, 0]
290g		[1, 0, 0, 0, 1, 0]	[0, 2, 2, 2, 0, 0]
300g		[0, 0, 0, 0, 1, 0]	[0, 2, 2, 2, 0, 0]
310g		[0, 0, 0, 0, 1, 0]	[1, 2, 2, 2, 0, 0]
320g		[0, 0, 0, 0, 1, 0]	[2, 2, 2, 2, 0, 0]
330g		[0, 0, 0, 0, 1, 0]	[3, 2, 2, 2, 0, 0]
340g		[1, 0, 0, 0, 1, 0]	[0, 1, 3, 2, 0, 0]
350g		[0, 0, 0, 0, 1, 0]	[0, 1, 3, 2, 0, 0]
360g		[0, 0, 0, 0, 1, 0]	[1, 1, 3, 2, 0, 0]

³Diese Darstellung wurde genutzt um Platz zu sparen, eine übersichtlicheres Format finden Sie unter den Beispielen bei "results"

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
370g		[0, 0, 0, 0, 1, 0]	[2, 1, 3, 2, 0, 0]
380g		[0, 0, 0, 0, 1, 0]	[3, 1, 3, 2, 0, 0]
390g		[1, 0, 0, 0, 1, 0]	[0, 2, 3, 2, 0, 0]
400g		[0, 0, 0, 0, 1, 0]	[0, 2, 3, 2, 0, 0]
410g		[0, 0, 0, 0, 1, 0]	[1, 2, 3, 2, 0, 0]
420g		[0, 0, 0, 0, 1, 0]	[2, 2, 3, 2, 0, 0]
430g		[0, 0, 0, 0, 1, 0]	[3, 2, 3, 2, 0, 0]
440g		[1, 0, 1, 0, 1, 0]	[0, 1, 0, 3, 0, 0]
450g		[0, 0, 1, 0, 1, 0]	[0, 1, 0, 3, 0, 0]
460g		[0, 0, 1, 0, 1, 0]	[1, 1, 0, 3, 0, 0]
470g		[0, 0, 1, 0, 1, 0]	[2, 1, 0, 3, 0, 0]
480g		[0, 0, 1, 0, 1, 0]	[3, 1, 0, 3, 0, 0]
490g		[1, 0, 1, 0, 1, 0]	[0, 2, 0, 3, 0, 0]
500g		[0, 0, 1, 0, 1, 0]	[0, 2, 0, 3, 0, 0]
9500g		[0, 0, 1, 0, 0, 0]	[0, 2, 0, 3, 3, 1]
9510g		[0, 0, 1, 0, 0, 0]	[1, 2, 0, 3, 3, 1]
9520g		[0, 0, 1, 0, 0, 0]	[2, 2, 0, 3, 3, 1]
9530g		[0, 0, 1, 0, 0, 0]	[3, 2, 0, 3, 3, 1]
9540g		[1, 0, 0, 0, 0, 0]	[0, 1, 0, 3, 3, 1]
9550g		[0, 0, 0, 0, 0, 0]	[0, 1, 0, 3, 3, 1]
9560g		[0, 0, 0, 0, 0, 0]	[1, 1, 0, 3, 3, 1]
9570g		[0, 0, 0, 0, 0, 0]	[2, 1, 0, 3, 3, 1]
9580g		[0, 0, 0, 0, 0, 0]	[3, 1, 0, 3, 3, 1]
9590g		[1, 0, 0, 0, 0, 0]	[0, 2, 0, 3, 3, 1]
9600g		[0, 0, 0, 0, 0, 0]	[0, 2, 0, 3, 3, 1]
9610g		[0, 0, 0, 0, 0, 0]	[1, 2, 0, 3, 3, 1]
9620g		[0, 0, 0, 0, 0, 0]	[2, 2, 0, 3, 3, 1]
9630g		[0, 0, 0, 0, 0, 0]	[3, 2, 0, 3, 3, 1]
9640g		[1, 0, 0, 0, 0, 0]	[0, 1, 1, 3, 3, 1]
9650g		[0, 0, 0, 0, 0, 0]	[0, 1, 1, 3, 3, 1]
9660g		[0, 0, 0, 0, 0, 0]	[1, 1, 1, 3, 3, 1]
9670g		[0, 0, 0, 0, 0, 0]	[2, 1, 1, 3, 3, 1]
9680g		[0, 0, 0, 0, 0, 0]	[3, 1, 1, 3, 3, 1]
9690g		[1, 0, 0, 0, 0, 0]	[0, 2, 1, 3, 3, 1]
9700g		[0, 0, 0, 0, 0, 0]	[0, 2, 1, 3, 3, 1]
9710g		[0, 0, 0, 0, 0, 0]	[1, 2, 1, 3, 3, 1]
9720g		[0, 0, 0, 0, 0, 0]	[2, 2, 1, 3, 3, 1]
9730g		[0, 0, 0, 0, 0, 0]	[3, 2, 1, 3, 3, 1]
9740g		[1, 0, 0, 0, 0, 0]	[0, 1, 2, 3, 3, 1]
9750g		[0, 0, 0, 0, 0, 0]	[0, 1, 2, 3, 3, 1]
9760g		[0, 0, 0, 0, 0, 0]	[1, 1, 2, 3, 3, 1]
9770g		[0, 0, 0, 0, 0, 0]	[2, 1, 2, 3, 3, 1]
9780g		[0, 0, 0, 0, 0, 0]	[3, 1, 2, 3, 3, 1]
9790g		[1, 0, 0, 0, 0, 0]	[0, 2, 2, 3, 3, 1]
9800g		[0, 0, 0, 0, 0, 0]	[0, 2, 2, 3, 3, 1]
9810g		[0, 0, 0, 0, 0, 0]	[1, 2, 2, 3, 3, 1]
9820g		[0, 0, 0, 0, 0, 0]	[2, 2, 2, 3, 3, 1]
9830g		[0, 0, 0, 0, 0, 0]	[3, 2, 2, 3, 3, 1]
9840g		[1, 0, 0, 0, 0, 0]	[0, 1, 3, 3, 3, 1]
9850g		[0, 0, 0, 0, 0, 0]	[0, 1, 3, 3, 3, 1]
9860g		[0, 0, 0, 0, 0, 0]	[1, 1, 3, 3, 3, 1]
9870g		[0, 0, 0, 0, 0, 0]	[2, 1, 3, 3, 3, 1]
9880g		[0, 0, 0, 0, 0, 0]	[3, 1, 3, 3, 3, 1]
9890g		[1, 0, 0, 0, 0, 0]	[0, 2, 3, 3, 3, 1]
9900g		[0, 0, 0, 0, 0, 0]	[0, 2, 3, 3, 3, 1]
9910g		[0, 0, 0, 0, 0, 0]	[1, 2, 3, 3, 3, 1]
9920g		[0, 0, 0, 0, 0, 0]	[2, 2, 3, 3, 3, 1]
9930g		[0, 0, 0, 0, 0, 0]	[3, 2, 3, 3, 3, 1]
9940g	9930g	[0, 0, 0, 0, 0, 0]	[3, 2, 3, 3, 3, 1]
9950g	9930g	[0, 0, 0, 0, 0, 0]	[3, 2, 3, 3, 3, 1]
9960g	9930g	[0, 0, 0, 0, 0, 0]	[3, 2, 3, 3, 3, 1]
9970g	9930g	[0, 0, 0, 0, 0, 0]	[3, 2, 3, 3, 3, 1]

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
9980g	9930g	[0, 0, 0, 0, 0, 0]	[3, 2, 3, 3, 3, 1]
9990g	9930g	[0, 0, 0, 0, 0, 0]	[3, 2, 3, 3, 3, 1]
10000g	9930g	[0, 0, 0, 0, 0, 0]	[3, 2, 3, 3, 3, 1]

4.2 Beispiel 1

gewichtsstuecke1.txt enthält insgesamt 17 Gewichte, davon 4 unterschiedliche. Der Algorithmus prüft insgesamt 7007 Permutationen, und bearbeitet diese in etwa 500µs.

Index	0	1	2	3
Gewicht	42g	127g	371g	2000g

Kombinationen			
Masse	M _{gerundet}	links	rechts
10g		[0, 0, 1, 0]	[0, 3, 0, 0]
20g	19g	[3, 0, 5, 0]	[0, 0, 0, 1]
30g	29g	[3, 0, 6, 0]	[0, 3, 0, 1]
40g	42g	[0, 0, 0, 0]	[1, 0, 0, 0]
50g	51g	[2, 3, 4, 0]	[0, 0, 0, 1]
60g		[0, 1, 5, 0]	[1, 0, 0, 1]
70g		[0, 0, 6, 0]	[1, 2, 0, 1]
80g	84g	[0, 0, 0, 0]	[2, 0, 0, 0]
90g	93g	[1, 3, 4, 0]	[0, 0, 0, 1]
100g		[3, 0, 0, 1]	[0, 0, 6, 0]
110g		[3, 0, 0, 1]	[0, 3, 5, 0]
120g	118g	[3, 1, 0, 0]	[0, 0, 1, 0]
130g	128g	[3, 0, 0, 0]	[0, 2, 0, 0]
140g	141g	[0, 1, 0, 1]	[1, 0, 6, 0]
150g	151g	[0, 0, 0, 1]	[1, 2, 5, 0]
160g		[2, 1, 0, 0]	[0, 0, 1, 0]
170g		[2, 0, 0, 0]	[0, 2, 0, 0]
180g	178g	[2, 2, 4, 0]	[0, 0, 0, 1]
190g	188g	[2, 0, 5, 0]	[0, 1, 0, 1]
200g	201g	[0, 2, 0, 0]	[2, 0, 1, 0]
210g	211g	[0, 0, 0, 0]	[2, 1, 0, 0]
220g		[1, 2, 4, 0]	[0, 0, 0, 1]
230g		[1, 0, 5, 0]	[0, 1, 0, 1]
240g	239g	[0, 0, 6, 0]	[2, 3, 0, 1]
250g	253g	[0, 0, 0, 0]	[3, 1, 0, 0]
260g	261g	[0, 3, 4, 0]	[3, 0, 0, 1]
270g	271g	[0, 0, 5, 0]	[3, 0, 0, 1]
280g	281g	[0, 0, 6, 0]	[3, 3, 0, 1]
290g	287g	[2, 0, 0, 0]	[0, 0, 1, 0]
300g	297g	[2, 0, 0, 0]	[0, 3, 0, 0]
310g		[0, 0, 0, 1]	[2, 0, 6, 0]
320g		[0, 0, 0, 1]	[2, 3, 5, 0]
330g	329g	[1, 0, 0, 0]	[0, 0, 1, 0]
340g	339g	[1, 0, 0, 0]	[0, 3, 0, 0]
350g	352g	[0, 0, 0, 1]	[3, 0, 6, 0]
360g	361g	[0, 3, 0, 0]	[0, 0, 2, 0]
370g		[0, 1, 0, 0]	[3, 0, 1, 0]
380g		[0, 0, 0, 0]	[3, 2, 0, 0]
390g		[3, 0, 4, 0]	[0, 0, 0, 1]
400g		[3, 0, 5, 0]	[0, 3, 0, 1]
410g	413g	[0, 0, 0, 0]	[1, 0, 1, 0]
420g	422g	[2, 3, 3, 0]	[0, 0, 0, 1]
430g	431g	[0, 1, 4, 0]	[1, 0, 0, 1]
440g	441g	[0, 0, 5, 0]	[1, 2, 0, 1]

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
450g	446g	[1, 2, 0, 0]	[0, 0, 2, 0]
460g	464g	[1, 3, 3, 0]	[0, 0, 0, 1]
470g	473g	[0, 1, 4, 0]	[2, 0, 0, 1]
480g		[0, 0, 0, 1]	[0, 2, 6, 0]
490g	489g	[3, 1, 0, 0]	[0, 0, 2, 0]
500g	499g	[3, 0, 0, 0]	[0, 2, 1, 0]
9500g	9501g	[0, 2, 1, 0]	[3, 0, 0, 5]
9510g	9511g	[0, 0, 2, 0]	[3, 1, 0, 5]
9520g	9517g	[2, 2, 0, 0]	[0, 0, 5, 4]
9530g	9527g	[2, 0, 0, 0]	[0, 1, 4, 4]
9540g	9544g	[0, 1, 1, 0]	[1, 0, 0, 5]
9550g	9554g	[0, 0, 2, 0]	[1, 2, 0, 5]
9560g	9559g	[1, 2, 0, 0]	[0, 0, 5, 4]
9570g	9569g	[1, 0, 0, 0]	[0, 1, 4, 4]
9580g	9578g	[0, 0, 0, 0]	[2, 3, 3, 4]
9590g	9587g	[1, 0, 1, 0]	[0, 0, 0, 5]
9600g		[0, 3, 0, 0]	[3, 0, 5, 4]
9610g		[0, 0, 0, 0]	[3, 0, 4, 4]
9620g		[0, 0, 0, 0]	[3, 3, 3, 4]
9630g		[3, 0, 1, 0]	[0, 1, 0, 5]
9640g	9639g	[0, 0, 2, 0]	[0, 3, 0, 5]
9650g	9653g	[0, 0, 0, 0]	[1, 1, 4, 4]
9660g	9661g	[0, 3, 0, 0]	[1, 0, 0, 5]
9670g	9671g	[0, 0, 1, 0]	[1, 0, 0, 5]
9680g	9681g	[0, 0, 2, 0]	[1, 3, 0, 5]
9690g	9686g	[1, 1, 0, 0]	[0, 0, 5, 4]
9700g	9703g	[0, 3, 0, 0]	[2, 0, 0, 5]
9710g	9713g	[0, 0, 1, 0]	[2, 0, 0, 5]
9720g	9719g	[3, 3, 0, 0]	[0, 0, 6, 4]
9730g	9729g	[3, 0, 0, 0]	[0, 0, 5, 4]
9740g	9739g	[3, 0, 0, 0]	[0, 3, 4, 4]
9750g	9747g	[3, 1, 0, 0]	[0, 0, 0, 5]
9760g	9761g	[2, 3, 0, 0]	[0, 0, 6, 4]
9770g		[0, 1, 0, 0]	[1, 0, 5, 4]
9780g		[0, 0, 0, 0]	[1, 2, 4, 4]
9790g	9789g	[2, 1, 0, 0]	[0, 0, 0, 5]
9800g	9799g	[2, 0, 1, 0]	[0, 2, 0, 5]
9810g	9812g	[0, 1, 0, 0]	[2, 0, 5, 4]
9820g	9822g	[0, 0, 0, 0]	[2, 2, 4, 4]
9830g		[0, 2, 0, 0]	[2, 0, 0, 5]
9840g		[0, 0, 1, 0]	[2, 1, 0, 5]
9850g	9854g	[0, 1, 0, 0]	[3, 0, 5, 4]
9860g	9864g	[0, 0, 0, 0]	[3, 2, 4, 4]
9870g	9872g	[0, 2, 0, 0]	[3, 0, 0, 5]
9880g	9882g	[0, 0, 1, 0]	[3, 1, 0, 5]
9890g	9888g	[2, 2, 0, 0]	[0, 0, 6, 4]
9900g	9898g	[2, 0, 0, 0]	[0, 1, 5, 4]
9910g	9907g	[0, 0, 0, 0]	[1, 3, 4, 4]
9920g	9916g	[2, 0, 0, 0]	[0, 0, 0, 5]
9930g		[1, 2, 0, 0]	[0, 0, 6, 4]
9940g		[1, 0, 0, 0]	[0, 1, 5, 4]
9950g	9949g	[0, 0, 0, 0]	[2, 3, 4, 4]
9960g	9958g	[1, 0, 0, 0]	[0, 0, 0, 5]
9970g	9971g	[0, 3, 0, 0]	[3, 0, 6, 4]
9980g	9981g	[0, 0, 0, 0]	[3, 0, 5, 4]
9990g		[0, 3, 0, 0]	[0, 0, 1, 5]
10000g		[0, 0, 0, 0]	[0, 0, 0, 5]

4.3 Beispiel 2

gewichtsstuecke2.txt enthält insgesamt 10 Gewichte, davon 10 unterschiedliche. Der Algorithmus prüft insgesamt 59049 Permutationen, und bearbeitet diese in etwa 3.2ms.

Index	0	1	2	3	4	5	6	7	8	9
Gewicht	10g	20g	40g	80g	160g	320g	640g	1280g	2560g	5120g

Kombinationen			
Masse	M _{gerundet}	links	rechts
10g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
20g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
30g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
40g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
50g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 0, 0, 0, 0, 0]
60g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 0, 0, 0, 0, 0, 0]
70g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
80g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
90g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 0, 0, 0, 0, 0]
100g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 0, 0, 0, 0, 0]
110g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 1, 0, 0, 0, 0, 0, 0]
120g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 0, 0, 0, 0, 0, 0]
130g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 0, 0, 0, 0, 0, 0]
140g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 1, 0, 0, 0, 0, 0, 0]
150g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
160g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
170g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 0, 0, 0, 0, 0]
180g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 1, 0, 0, 0, 0, 0]
190g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 1, 0, 0, 0, 0, 0]
200g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 1, 0, 0, 0, 0, 0]
210g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 1, 0, 0, 0, 0, 0]
220g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 1, 0, 0, 0, 0, 0]
230g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 1, 0, 0, 0, 0, 0]
240g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 1, 0, 0, 0, 0, 0]
250g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 0, 0, 0, 0, 0]
260g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 1, 0, 0, 0, 0, 0]
270g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 1, 1, 0, 0, 0, 0, 0]
280g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 1, 0, 0, 0, 0, 0]
290g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 1, 0, 0, 0, 0, 0]
300g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 1, 1, 0, 0, 0, 0, 0]
310g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
320g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
330g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 1, 0, 0, 0, 0]
340g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 1, 0, 0, 0, 0]
350g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 0, 1, 0, 0, 0, 0]
360g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 0, 0, 0, 0]
370g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 1, 0, 0, 0, 0]
380g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 0, 1, 0, 0, 0, 0]
390g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 0, 1, 0, 0, 0, 0]
400g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 0, 0, 0, 0]
410g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 1, 0, 0, 0, 0]
420g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 1, 0, 0, 0, 0]
430g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 1, 0, 1, 0, 0, 0, 0]
440g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 0, 1, 0, 0, 0, 0]
450g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 0, 1, 0, 0, 0, 0]
460g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 1, 0, 1, 0, 0, 0, 0]
470g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 0, 1, 0, 0, 0, 0]
480g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 1, 0, 0, 0, 0]
490g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 1, 0, 0, 0, 0]
500g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 1, 1, 0, 0, 0, 0]
9500g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 1, 1, 0, 1, 1, 1]
9510g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 1, 1, 0, 1, 1, 1]
9520g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 1, 1, 0, 1, 1, 1]

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
9530g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 1, 0, 1, 1, 1]
9540g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 1, 1, 0, 1, 1, 1]
9550g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 1, 1, 1, 0, 1, 1, 1]
9560g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 1, 1, 0, 1, 1, 1]
9570g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 1, 1, 0, 1, 1, 1]
9580g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 1, 1, 1, 0, 1, 1, 1]
9590g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 1, 1, 0, 1, 1, 1]
9600g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 1, 1, 1]
9610g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 1, 1, 1, 1]
9620g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 1, 1, 1, 1]
9630g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 0, 0, 1, 1, 1, 1]
9640g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 1, 1, 1, 1]
9650g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 0, 1, 1, 1, 1]
9660g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 0, 0, 1, 1, 1, 1]
9670g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 0, 0, 1, 1, 1, 1]
9680g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 1, 1, 1, 1]
9690g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 0, 1, 1, 1, 1]
9700g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 0, 1, 1, 1, 1]
9710g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 1, 0, 0, 1, 1, 1, 1]
9720g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 0, 0, 1, 1, 1, 1]
9730g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 0, 0, 1, 1, 1, 1]
9740g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 1, 0, 0, 1, 1, 1, 1]
9750g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 0, 0, 1, 1, 1, 1]
9760g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 1, 1, 1, 1]
9770g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 0, 1, 1, 1, 1]
9780g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 1, 0, 1, 1, 1, 1]
9790g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 1, 0, 1, 1, 1, 1]
9800g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 1, 0, 1, 1, 1, 1]
9810g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 1, 0, 1, 1, 1, 1]
9820g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 1, 0, 1, 1, 1, 1]
9830g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 1, 0, 1, 1, 1, 1]
9840g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 1, 0, 1, 1, 1, 1]
9850g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 0, 1, 1, 1, 1]
9860g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 1, 0, 1, 1, 1, 1]
9870g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 1, 1, 0, 1, 1, 1, 1]
9880g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 1, 0, 1, 1, 1, 1]
9890g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 1, 0, 1, 1, 1, 1]
9900g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 1, 1, 0, 1, 1, 1, 1]
9910g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 1, 0, 1, 1, 1, 1]
9920g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
9930g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 1, 1, 1, 1, 1]
9940g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 1, 1, 1, 1, 1]
9950g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
9960g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1, 1, 1, 1]
9970g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 1, 1, 1, 1, 1]
9980g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 0, 1, 1, 1, 1, 1]
9990g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 0, 1, 1, 1, 1, 1]
10000g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1, 1, 1, 1]

4.4 Beispiel 3

Das einfachste Beispiel *gewichtsstuecke3.txt* enthält insgesamt 7 Gewichte, davon 7 unterschiedliche. Der Algorithmus prüft insgesamt 2187 Permutationen, und bearbeitet diese in etwa 200µs.

Index	0	1	2	3	4	5	6
Gewicht	10g	30g	90g	270g	810g	2430g	7290g

Masse	Kombinationen		
	M _{gerundet}	links	rechts
10g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0]
20g		[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0]
30g		[0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0]
40g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 0, 0, 0]
50g		[1, 1, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0]
60g		[0, 1, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0]
70g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 0, 0]
80g		[1, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0]
90g		[0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0]
100g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 0, 0]
110g		[1, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 0, 0, 0]
120g		[0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 0, 0, 0]
130g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 0, 0, 0]
140g		[1, 1, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0]
150g		[0, 1, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0]
160g		[0, 1, 1, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 0, 0]
170g		[1, 0, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0]
180g		[0, 0, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0]
190g		[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 0, 0]
200g		[1, 0, 1, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 0, 0]
210g		[0, 0, 1, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 0, 0]
220g		[0, 0, 1, 0, 0, 0, 0]	[1, 1, 0, 1, 0, 0, 0]
230g		[1, 1, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0]
240g		[0, 1, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0]
250g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 0, 0]
260g		[1, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0]
270g		[0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 0, 0]
280g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 0, 0]
290g		[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 0, 0]
300g		[0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 0, 0]
310g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 1, 0, 0, 0]
320g		[1, 1, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 0, 0, 0]
330g		[0, 1, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 0, 0, 0]
340g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 0, 0, 0]
350g		[1, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 0, 0, 0]
360g		[0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 0, 0, 0]
370g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 0, 0, 0]
380g		[1, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 1, 0, 0, 0]
390g		[0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 1, 0, 0, 0]
400g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 0, 0, 0]
410g		[1, 1, 1, 1, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0]
420g		[0, 1, 1, 1, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0]
430g		[0, 1, 1, 1, 0, 0, 0]	[1, 0, 0, 0, 1, 0, 0]
440g		[1, 0, 1, 1, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0]
450g		[0, 0, 1, 1, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0]
460g		[0, 0, 1, 1, 0, 0, 0]	[1, 0, 0, 0, 1, 0, 0]
470g		[1, 0, 1, 1, 0, 0, 0]	[0, 1, 0, 0, 1, 0, 0]
480g		[0, 0, 1, 1, 0, 0, 0]	[0, 1, 0, 0, 1, 0, 0]
490g		[0, 0, 1, 1, 0, 0, 0]	[1, 1, 0, 0, 1, 0, 0]
500g		[1, 1, 0, 1, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0]
9500g		[1, 1, 0, 1, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1]
9510g		[0, 1, 0, 1, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1]
9520g		[0, 1, 0, 1, 0, 0, 0]	[1, 0, 1, 0, 0, 1, 1]
9530g		[1, 0, 0, 1, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1]
9540g		[0, 0, 0, 1, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1]
9550g		[0, 0, 0, 1, 0, 0, 0]	[1, 0, 1, 0, 0, 1, 1]
9560g		[1, 0, 0, 1, 0, 0, 0]	[0, 1, 1, 0, 0, 1, 1]
9570g		[0, 0, 0, 1, 0, 0, 0]	[0, 1, 1, 0, 0, 1, 1]
9580g		[0, 0, 0, 1, 0, 0, 0]	[1, 1, 1, 0, 0, 1, 1]
9590g		[1, 1, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1]
9600g		[0, 1, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1]

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
9610g		[0, 1, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 1, 1]
9620g		[1, 0, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1]
9630g		[0, 0, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1]
9640g		[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 1, 1]
9650g		[1, 0, 1, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 1, 1]
9660g		[0, 0, 1, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 1, 1]
9670g		[0, 0, 1, 0, 0, 0, 0]	[1, 1, 0, 0, 0, 1, 1]
9680g		[1, 1, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1]
9690g		[0, 1, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1]
9700g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 1, 1]
9710g		[1, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1]
9720g		[0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 1]
9730g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 1, 1]
9740g		[1, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 1, 1]
9750g		[0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 1, 1]
9760g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 0, 1, 1]
9770g		[1, 1, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1]
9780g		[0, 1, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1]
9790g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 1, 1]
9800g		[1, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1]
9810g		[0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 1, 1]
9820g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 1, 1]
9830g		[1, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 0, 1, 1]
9840g		[0, 0, 0, 0, 0, 0, 0]	[0, 1, 1, 0, 0, 1, 1]
9850g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 0, 1, 1]
9860g		[1, 1, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1]
9870g		[0, 1, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1]
9880g		[0, 1, 1, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 1, 1]
9890g		[1, 0, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1]
9900g		[0, 0, 1, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1]
9910g		[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 1, 1]
9920g		[1, 0, 1, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 1, 1]
9930g		[0, 0, 1, 0, 0, 0, 0]	[0, 1, 0, 1, 0, 1, 1]
9940g		[0, 0, 1, 0, 0, 0, 0]	[1, 1, 0, 1, 0, 1, 1]
9950g		[1, 1, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1]
9960g		[0, 1, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1]
9970g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 1, 1]
9980g		[1, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1]
9990g		[0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 1]
10000g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 1, 1]

4.5 Beispiel 4

gewichtsstuecke4.txt enthält insgesamt 13 Gewichte, davon 7 unterschiedliche. Der Algorithmus prüft insgesamt 27783 Permutationen, und bearbeitet diese in etwa 1.7ms.

Index	0	1	2	3	4	5	6
Gewicht	5g	21g	29g	259g	287g	399g	2993g

Kombinationen			
Masse	M _{gerundet}	links	rechts
10g		[0, 0, 2, 0, 0, 0, 0]	[1, 3, 0, 0, 0, 0, 0]
20g		[0, 2, 0, 1, 0, 0, 0]	[1, 0, 1, 0, 1, 0, 0]
30g		[0, 0, 3, 0, 1, 0, 0]	[1, 0, 0, 0, 0, 1, 0]
40g		[0, 0, 0, 0, 1, 0, 0]	[1, 3, 0, 1, 0, 0, 0]
50g		[0, 2, 0, 0, 0, 0, 0]	[1, 0, 3, 0, 0, 0, 0]
60g		[0, 3, 1, 0, 0, 1, 0]	[1, 0, 0, 1, 1, 0, 0]
70g		[0, 1, 0, 1, 0, 0, 0]	[1, 0, 2, 0, 1, 0, 0]

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
80g		[0, 0, 2, 0, 1, 0, 0]	[1, 1, 0, 0, 0, 1, 0]
90g		[0, 1, 1, 1, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 0]
100g		[0, 0, 3, 1, 0, 0, 0]	[1, 2, 0, 0, 0, 1, 0]
110g		[0, 2, 0, 0, 0, 1, 0]	[1, 0, 0, 1, 1, 0, 0]
120g		[0, 0, 0, 1, 0, 0, 0]	[1, 0, 3, 0, 1, 0, 0]
130g		[0, 0, 1, 0, 1, 0, 0]	[1, 2, 0, 0, 0, 1, 0]
140g		[0, 3, 0, 1, 0, 0, 0]	[1, 0, 2, 0, 0, 1, 0]
150g		[0, 0, 2, 1, 0, 0, 0]	[1, 3, 0, 0, 0, 1, 0]
160g		[0, 1, 0, 0, 0, 1, 0]	[1, 0, 1, 1, 1, 0, 0]
170g		[0, 0, 0, 0, 1, 0, 0]	[0, 0, 2, 0, 0, 1, 0]
180g		[0, 0, 0, 0, 1, 0, 0]	[1, 3, 0, 0, 0, 1, 0]
190g		[0, 2, 0, 1, 0, 0, 0]	[1, 0, 3, 0, 0, 1, 0]
200g		[0, 3, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 0, 0]
210g		[0, 0, 0, 0, 0, 1, 0]	[1, 0, 2, 1, 1, 0, 0]
220g		[0, 0, 0, 0, 1, 0, 0]	[0, 1, 3, 0, 0, 1, 0]
230g		[0, 3, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 0, 0, 0]
240g		[0, 0, 3, 0, 0, 0, 0]	[1, 3, 0, 1, 0, 0, 0]
250g		[0, 2, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 0, 0]
260g		[0, 0, 0, 0, 0, 1, 0]	[1, 1, 3, 1, 1, 0, 0]
270g		[0, 2, 3, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 0]
280g		[0, 2, 0, 0, 0, 0, 0]	[1, 0, 2, 1, 0, 0, 0]
290g		[0, 0, 0, 1, 0, 0, 0]	[0, 3, 3, 0, 0, 1, 0]
300g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 1, 0, 0]
310g		[0, 0, 3, 0, 1, 0, 0]	[1, 1, 0, 1, 0, 1, 0]
320g		[0, 1, 2, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1, 0]
330g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 3, 1, 0, 0, 0]
340g		[0, 3, 1, 1, 0, 0, 0]	[1, 0, 0, 0, 1, 1, 0]
350g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 2, 0, 1, 0, 0]
360g		[0, 0, 2, 0, 1, 0, 0]	[1, 2, 0, 1, 0, 1, 0]
370g		[0, 3, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 1, 0]
380g		[0, 0, 3, 0, 0, 0, 0]	[1, 3, 0, 0, 0, 1, 0]
390g		[0, 2, 0, 1, 0, 0, 0]	[1, 0, 0, 0, 1, 1, 0]
400g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 3, 0, 1, 0, 0]
410g		[0, 0, 1, 0, 1, 0, 0]	[1, 3, 0, 1, 0, 1, 0]
420g		[0, 2, 0, 0, 0, 0, 0]	[1, 0, 2, 0, 0, 1, 0]
430g		[0, 3, 2, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 0, 0]
440g		[0, 1, 0, 1, 0, 0, 0]	[1, 0, 1, 0, 1, 1, 0]
450g		[0, 0, 0, 0, 1, 0, 0]	[0, 1, 2, 1, 0, 1, 0]
460g		[1, 1, 0, 0, 0, 0, 0]	[0, 0, 3, 0, 0, 1, 0]
470g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 3, 0, 0, 1, 0]
480g		[0, 2, 1, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 0, 0]
490g		[0, 0, 0, 1, 0, 0, 0]	[1, 0, 2, 0, 1, 1, 0]
500g		[0, 0, 0, 0, 1, 0, 0]	[0, 2, 3, 1, 0, 1, 0]
9500g		[0, 0, 0, 0, 1, 0, 0]	[0, 3, 3, 1, 0, 1, 3]
9510g		[1, 3, 3, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 1, 3]
9520g		[0, 3, 3, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 1, 3]
9530g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 0, 3]
9540g		[0, 0, 0, 1, 0, 0, 0]	[1, 2, 3, 0, 1, 1, 3]
9550g		[0, 3, 1, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 1, 3]
9560g		[1, 2, 2, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 1, 3]
9570g		[0, 2, 2, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 1, 3]
9580g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 1, 0, 3]
9590g		[1, 2, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 0, 1, 3]
9600g		[0, 2, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 0, 1, 3]
9610g		[1, 1, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 1, 3]
9620g		[0, 1, 1, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 1, 3]
9630g		[0, 0, 0, 0, 0, 0, 0]	[1, 2, 2, 1, 1, 0, 3]
9640g		[1, 1, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 0, 1, 3]
9650g		[0, 1, 0, 0, 0, 0, 0]	[1, 0, 1, 1, 0, 1, 3]
9660g		[0, 3, 0, 0, 0, 0, 0]	[0, 0, 2, 0, 1, 1, 3]
9670g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 1, 3]
9680g		[0, 0, 0, 0, 0, 0, 0]	[1, 3, 3, 1, 1, 0, 3]

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
9690g	9731g	[1, 0, 0, 0, 0, 0, 0]	[0, 0, 2, 1, 0, 1, 3]
9700g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 2, 1, 0, 1, 3]
9710g		[0, 2, 0, 0, 0, 0, 0]	[0, 0, 3, 0, 1, 1, 3]
9720g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 0, 1, 1, 3]
9730g		[0, 1, 0, 0, 0, 0, 0]	[0, 0, 3, 0, 1, 1, 3]
9740g		[1, 0, 0, 0, 0, 0, 0]	[0, 1, 3, 1, 0, 1, 3]
9750g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 3, 1, 0, 1, 3]
9760g		[1, 0, 0, 0, 0, 0, 0]	[0, 2, 2, 0, 1, 1, 3]
9770g	9781g	[0, 0, 0, 0, 0, 0, 0]	[1, 2, 2, 0, 1, 1, 3]
9780g		[1, 0, 0, 0, 0, 0, 0]	[0, 3, 2, 0, 1, 1, 3]
9790g		[1, 2, 3, 0, 0, 0, 0]	[0, 0, 0, 1, 1, 1, 3]
9800g		[0, 2, 3, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 1, 3]
9810g	9829g	[1, 0, 0, 0, 0, 0, 0]	[0, 3, 3, 0, 1, 1, 3]
9820g		[0, 0, 0, 0, 0, 0, 0]	[1, 3, 3, 0, 1, 1, 3]
9830g		[0, 2, 2, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 1, 3]
9840g		[1, 1, 2, 0, 0, 0, 0]	[0, 0, 0, 1, 1, 1, 3]
9850g		[0, 1, 2, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 1, 3]
9860g	9861g	[0, 3, 0, 0, 0, 0, 0]	[0, 0, 0, 1, 1, 1, 3]
9870g	9871g	[0, 0, 2, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 1, 3]
9880g	9879g	[0, 1, 1, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 1, 3]
9890g	9911g	[0, 3, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 1, 1, 3]
9900g		[0, 0, 1, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 1, 3]
9910g		[0, 2, 0, 0, 0, 0, 0]	[0, 0, 1, 1, 1, 1, 3]
9920g		[0, 0, 1, 0, 0, 0, 0]	[1, 1, 0, 1, 1, 1, 3]
9930g		[0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 1, 1, 1, 3]
9940g		[0, 2, 0, 0, 0, 0, 0]	[0, 0, 2, 1, 1, 1, 3]
9950g		[0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 1, 1, 1, 3]
9960g	9961g	[0, 1, 0, 0, 0, 0, 0]	[0, 0, 2, 1, 1, 1, 3]
9970g	9971g	[0, 0, 0, 0, 0, 0, 0]	[1, 2, 0, 1, 1, 1, 3]
9980g	9979g	[0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 1, 1, 3]
9990g	9990g	[0, 1, 0, 0, 0, 0, 0]	[0, 0, 3, 1, 1, 1, 3]
10000g		[0, 0, 0, 0, 0, 0, 0]	[1, 2, 1, 1, 1, 1, 3]

4.6 Beispiel 5

Das schwierigste Beispiel *gewichtsstuecke5.txt* enthält insgesamt 23 Gewichte, davon 13 unterschiedliche. Der Algorithmus prüft insgesamt 101269035 Permutationen, und bearbeitet diese in etwa 1.1s.

Index	0	1	2	3	4	5	6
Gewicht	11g	99480g	99511g	299836g	599761g	4497786g	1499171g
Index	7	8	9	10	11	12	
Gewicht	10499654g	41999427g	94499810g	283501867g	661499326g	1984505261g	

Kombinationen			
Masse	M _{gerundet}	links	rechts
10g	11g	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
20g	31g	[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
30g		[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
40g	42g	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
50g	47g	[1, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
60g	58g	[0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
70g	69g	[0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
80g	78g	[1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
90g	89g	[0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
100g	109g	[0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
110g		[1, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0]
120g	131g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0]
130g		[0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0]

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
140g	139g	[0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 1, 0, 2, 0, 2, 1, 4, 1, 0, 0, 0, 1]
150g		[0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 1, 0, 2, 0, 2, 1, 4, 1, 0, 0, 0, 1]
160g	159g	[1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 0, 2, 0, 2, 1, 4, 1, 0, 0, 0, 1]
170g		[0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 0, 2, 0, 2, 1, 4, 1, 0, 0, 0, 1]
180g	181g	[0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 0, 2, 0, 2, 1, 4, 1, 0, 0, 0, 1]
190g		[1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 1, 2, 0, 2, 1, 4, 1, 0, 0, 0, 1]
200g	201g	[0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 1, 2, 0, 2, 1, 4, 1, 0, 0, 0, 1]
210g	212g	[0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 2, 0, 2, 1, 4, 1, 0, 0, 0, 1]
220g	217g	[1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 1, 0, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
230g	228g	[0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 1, 0, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
240g	239g	[0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 1, 0, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
250g	248g	[1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 0, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
260g	259g	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 0, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
270g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 0, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
280g	279g	[1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 1, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
290g		[0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 1, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
300g	301g	[0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 0, 2, 1, 4, 1, 0, 0, 0, 1]
310g	306g	[1, 0, 1, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 1, 0, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
320g	317g	[0, 0, 1, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 1, 0, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
330g	328g	[0, 0, 1, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 1, 0, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
340g	337g	[1, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 0, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
350g	348g	[0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 0, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
360g	359g	[0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 0, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
370g	368g	[1, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
380g	379g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[0, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
390g		[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
400g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
410g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
420g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
430g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
440g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
450g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
460g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
470g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
480g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
490g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
500g	390g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 1, 0, 3, 0]	[1, 0, 1, 0, 1, 2, 1, 4, 1, 0, 0, 0, 1]
9500g	9503g	[0, 1, 0, 0, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
9510g	9508g	[1, 0, 1, 2, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
9520g	9519g	[0, 0, 1, 2, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
9530g		[0, 0, 1, 2, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
9540g	9539g	[1, 0, 0, 2, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
9550g		[0, 0, 0, 2, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
9560g	9561g	[0, 0, 0, 2, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
9570g		[1, 1, 0, 2, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
9580g	9581g	[0, 1, 0, 2, 0, 2, 1, 4, 1, 2, 0, 0, 0]	[0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]
9590g	9589g	[1, 0, 1, 0, 1, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9600g		[0, 0, 1, 0, 1, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9610g	9611g	[0, 0, 1, 0, 1, 0, 0, 0, 0, 3, 0, 3, 0]	[1, 1, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9620g		[1, 0, 0, 0, 1, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9630g	9631g	[0, 0, 0, 0, 1, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9640g	9642g	[0, 0, 0, 0, 1, 0, 0, 0, 0, 3, 0, 3, 0]	[1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9650g	9651g	[1, 1, 0, 0, 1, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9660g	9662g	[0, 1, 0, 0, 1, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9670g	9668g	[0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 1, 3, 0]	[0, 0, 0, 2, 0, 0, 0, 1, 0, 3, 0, 0, 1]
9680g	9679g	[0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 0, 2, 0, 0, 0, 1, 0, 3, 0, 0, 1]
9690g	9689g	[0, 0, 1, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9700g		[0, 0, 1, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9710g		[0, 1, 0, 0, 1, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 2, 0, 0, 0, 1, 0, 3, 0, 0, 1]
9720g		[0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9730g	9731g	[0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9740g		[1, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]

Weiterführung der Kombinationen			
Masse	M _{gerundet}	links	rechts
9750g	9751g	[0, 1, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9760g	9762g	[0, 1, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1]
9770g	9768g	[0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 0, 0, 0, 0, 0, 1, 0, 3, 0, 0, 1]
9780g	9778g	[0, 0, 1, 2, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1]
9790g	9789g	[0, 0, 1, 2, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1]
9800g	9799g	[0, 1, 0, 0, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 0, 0, 0, 1, 0, 3, 0, 0, 1]
9810g	9809g	[0, 0, 0, 2, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1]
9820g		[0, 0, 0, 2, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1]
9830g	9829g	[1, 1, 0, 2, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1]
9840g		[0, 1, 0, 2, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1]
9850g	9851g	[0, 1, 0, 2, 0, 0, 0, 0, 0, 3, 0, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1]
9860g	9857g	[0, 0, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 0, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9870g	9866g	[1, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[0, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9880g	9877g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[0, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9890g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9900g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9910g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9920g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9930g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9940g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9950g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9960g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9970g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9980g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
9990g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]
10000g	9888g	[0, 1, 0, 2, 0, 2, 1, 0, 0, 0, 1, 3, 0]	[1, 0, 1, 0, 1, 0, 0, 1, 0, 3, 0, 0, 1]

5 Quellcode

```

1  const MAX_GEWICHT: i64 = 10000;
2  const MIN_GEWICHT: i64 = 0;
3
4  static mut MAX: i64 = i64::MAX;
5
6  type SolutionSlice = (i64, Vec<i64>, Vec<i64>);
7
8  fn main() {
9      let args: Vec<String> = env::args().collect();
10     let input_file = args
11         .get(1)
12         .expect("Please provide the path to an input file");
13     let output_file = args
14         .get(2)
15         .expect("Please provide the path of the output file");
16
17     let solution = solve(input_file);
18     solution.write_to_file(output_file, true);
19 }
20
21 fn solve(input_file_path: &str) -> Solution {
22     let file = File::open(input_file_path).expect("Failed to read file");
23
24     println!("\n+ Solving for {}", input_file_path);
25
26     let arr = read_weights(&file);
27
28     let mut sums: HashMap<i64, Vec<i64>> = HashMap::new();
29     let mut counters = vec![0; arr.len()];
30     let start = SystemTime::now();
31     recurse(&mut sums, &arr, 0, &mut counters);
32
33     println!(
34         "+ Successfully ran algorithm.\n\t> Finished in {:?}\n\t> Total sums: {}",
35         start.elapsed().unwrap_or_default(),
36         sums.len()
37     );
38
39     unsafe {
40         MAX = i64::MAX;
41     }
42
43     let start = SystemTime::now();
44     let solution = Solution::create(&sums, &arr);
45     println!(
46         "+ Successfully created solution.\n\t> Finished in {:?}",
47         start.elapsed().unwrap_or_default()
48     );
49     return solution;
50 }
51
52 fn recurse(
53     sums: &mut HashMap<i64, Vec<i64>>,
54     arr: &Vec<Vec<i64>>,
55     n: usize,
56     counters: &mut Vec<i64>,
57 ) {
58     if n == arr.len() {
59         let mut sum: i64 = 0;
60         for i in 0..arr.len() {
61             let ints = arr.get(i).unwrap();

```

```

62         sum += ints[counters[i] as usize];
63     }
64
65     unsafe {
66         if sum <= MAX_GEWICHT && sum >= MIN_GEWICHT {
67             sums.insert(sum, counters.clone());
68         } else if sum < MAX && sum > MAX_GEWICHT {
69             sums.remove(&MAX);
70             MAX = sum;
71             sums.insert(sum, counters.clone());
72         }
73     }
74     return;
75 }
76 for i in 0..arr.get(n).expect("No arr at index n").len() {
77     counters[n] = i as i64;
78     recurse(sums, arr, n + 1, counters);
79 }
80 }
81
82 fn generate_weight_perms(count: i64, weight: i64) -> Vec<i64> {
83     let mut weights = Vec::new();
84     for i in (-count)..(count + 1) {
85         weights.push(i * weight);
86     }
87     return weights;
88 }
89
90 struct Solution {
91     values: Vec<SolutionSlice>,
92 }
93
94 impl Solution {
95     fn create(sums: &HashMap<i64, Vec<i64>>, arr: &Vec<Vec<i64>>>) -> Self {
96         let weights = {
97             let mut vec = vec![];
98             for local in arr {
99                 vec.push(local[local.len() - 1] / ((local.len() - 1) >> 1) as i64)
100             }
101             vec
102         };
103
104         let mut sorted: Vec<_> = sums.iter().collect();
105         sorted.sort();
106
107         let mut output = vec![];
108
109         for entry in sorted {
110             let sum = entry.0;
111             let counters = entry.1;
112
113             let mut left = vec![];
114             let mut right = vec![];
115
116             for i in 0..weights.len() {
117                 let weight = weights[i];
118                 let multiple = arr[i][counters[i] as usize];
119                 let abs_count = (multiple / weight).abs();
120
121                 if multiple.is_negative() {
122                     for _ in 0..abs_count {
123                         left.push(weight)
124                     }

```

```
125         } else if multiple.is_positive() {
126             for _ in 0..abs_count {
127                 right.push(weight)
128             }
129         }
130     }
131
132     output.push((*sum, left, right));
133 }
134 Self { values: output }
135 }
136
137 fn get_closest_combination(&self, weight: i64) -> SolutionSlice {
138     if weight < MIN_GEWICHT {
139         return self.values[0].clone();
140     }
141
142     let res = self.values.binary_search_by(|x| x.0.cmp(&(weight as i64)));
143
144     return match res {
145         Ok(index) => self.values[index].clone(),
146         Err(index) => {
147             if index == self.values.len()
148                 || (self.values[index].0 - weight as i64).abs()
149                 > (self.values[index - 1].0 - weight as i64).abs()
150             {
151                 self.values[index - 1].clone()
152             } else {
153                 self.values[index].clone()
154             }
155         }
156     };
157 }
158 }
```