1. Runde

des

42. Bundeswettbewerbs Informatik

Junioraufgabe 1: Wundertüte

Teamname: EZ

Team-ID: 00871

Bearbeitet

von

Florian Bange
Teilnahme-ID: 71639

21. November 2023

Inhaltsverzeichnis

0. Einleitung

Dieses Dokument beinhaltet meine Dokumentation der ersten Junioraufgabe[1] der ersten Runde des 42. Bundeswettbewerbs Informatik[2] aus dem Jahr 2023.

1. Modellierung und Definition des Problems

Im Folgenden wird die Aufgabe mathematisch definiert, um eine sinnvolle Loesung vorstellen zu koennen.

Im Weiteren sei $n \in \mathbb{N}$ die Anzahl der Wundertueten und $k \in \mathbb{N}$ die Anzahl der Spielsorten.
Weiter sei fuer $j \in \{1, 2, \ldots, k\}$ die Anzahl der Spiele der j-ten Spielsorte gegeben durch $a_j \in \mathbb{N}$.
Nach der Aufgabenstellung sollen nun alle $\sum_{j=1}^{k} a_j$ Spiele auf die n Wundertueten verteilt werden, sodass folgende Eigenschaften gelten:

1. Alle Spiele werden verteilt,
2. Die Anzahl der Spiele pro Wundertuete ist moeglichst gleich, und
3. Jede Spielsorte ist moeglichst gleichmaeszig auf die Wundertueten verteilt.

Im Folgenden wird die i-te Wundertuete auch Wundertuete i, sowie die j-te Spielsorte auch Spielsorte j genannt.

Definition 1: Verteilung

Eine Verteilung der Spiele auf die Wundertueten ist gegeben durch eine Matrix X mit n Zeilen und k Spalten:

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,k} \end{pmatrix}$$

Dabei ist fuer $j \in \{1, 2, \ldots, k\}$ und $i \in \{1, 2, \ldots, n\}$ die Anzahl der Spiele der Sorte j in der i-ten Wundertuete gegeben durch $x_{i,j} \in \mathbb{N}$.
Somit stellt fuer $i \in \{1, 2, \ldots, n\}$ die i-te Zeile der Matrix die Verteilung der Spiele auf die i-te Wundertuete dar:

$$X^i := (x_{i,1}, x_{i,2}, \ldots, x_{i,k})$$

Weiter wird fuer $j \in \{1, 2, \ldots, k\}$ die Verteilung der Spiele des Typen j auf alle Wundertueten dargestellt durch die j-te Spalte der Matrix:

$$X_j := \begin{pmatrix} x_{1,j} \\ x_{2,j} \\ \vdots \\ x_{n,j} \end{pmatrix}$$

sodass

$$X_j^T = (x_{1,j}, x_{2,j}, \ldots, x_{n,j})$$

Im Weiteren wird die Verteilung der Spiele auf die i-te Wundertuete auch einfach als die i-te Wundertuete bezeichnet.
Damit die Definition sinnvoll ist, muss fuer jeden Spieltypen $j \in \{1, 2, \ldots, k\}$ gelten, dass

$$\sum_{i=1}^{k} x_{i,j} \leq a_j,$$

sodass die Summe der Spiele der Spielsorte j über alle Wundertueten insgesamt kleiner oder gleich der Anzahl a_j der Spiele dieser Spielsorte ist.

[1] siehe `https://bwinf.de/fileadmin/bundeswettbewerb/42/BwInf_42_Aufgaben_WEB.pdf`
[2] Siehe `https://bwinf.de/bundeswettbewerb/42/1/`

Definition 2: Anzahl der Spiele

Gegeben sei eine Verteilung X der Spiele auf die n Wundertueten.

1. Fuer $i \in \{1,\ 2,\ \dots\ ,\ n\}$ sei
$$\|X^i\| \coloneqq \sum_{j=1}^{k} x_{i,j}$$
die Anzahl der Spiele der i-ten Wundertuete.

2. Weiter sei fuer $j \in \{1,\ 2,\ \dots\ ,\ k\}$
$$\|X_j\| \coloneqq \sum_{i=1}^{n} x_{i,j}$$
die Summe der verteilten Spiele des Spieltypen j.

3. Dann ist die Summe $\|X\|$ der Anzahl aller Spiele der Verteilung gegeben durch
$$\|X\| \coloneqq \sum_{i=1}^{n} \|X^i\| = \sum_{i=1}^{n}\sum_{j=1}^{k} x_{i,j} = \sum_{j=1}^{k}\sum_{i=1}^{n} x_{i,j} = \sum_{j=1}^{k} \|X_j\|$$

Bemerkung 3: Formalisierung der Aufgabenstellung

Es sei eine Verteilung X der Spiele auf die Wundertueten gegeben. Nun werden die Eigenschaften der Aufgabenstellung formalisiert.

1. Alle Spiele sollen verteilt werden. Dies ist gegeben, gdw. fuer jede Spielsorte $j \in \{1,\ 2,\ \dots\ ,\ k\}$ gilt, dass
$$\|X_j\| = a_j.$$

2. Die Anzahl der Spiele pro Wundertuete soll moeglichst gleich sein. D.h., es soll Folgendes minimiert werden:
$$\max\Big\{\ |\ \|X^a\| - \|X^b\|\ |\ :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}\Big\}$$
Dabei ist $|\ \|X^a\| - \|X^b\|\ |$ die absolute Differenz der Anzahl der Spiele der Wundertueten a und b. Es soll also das Maximum der absoluten Differenz der Anzahl der Spiele aller Paare an Wundertueten $(a,\ b)$ minimiert werden.

3. Jede Spielsorte soll moeglichst gleichmaeszig auf die Wundertueten verteilt sein. D.h., fuer jede Spielsorte $j \in \{1,\ 2,\ \dots\ ,\ k\}$ soll Folgendes minimiert werden:
$$\max\Big\{\ |\ x_{a,j} - x_{b,j}\ |\ :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}\ \Big\}$$
Dabei ist $|\ x_{a,j} - x_{b,j}\ |$ die absolute Differenz der Anzahl der Spiele des Spieltypen j der Wundertueten a und b. Es soll also das Maximum der absoluten Differenz der Anzahl der Spiele der Spielsorte j aller Paare an Wundertueten $(a,\ b)$ minimiert werden.

Definition 4: Optimale Verteilung

Eine Verteilung X der Spiele auf die Wundertueten heiszt optimale Verteilung, wenn Folgendes gilt:

1. Fuer jeden Spieltypen $j \in \{1,\ 2,\ \dots\ ,\ k\}$ gilt $\|X_j\| = a_j$. D.h., die Summe aller verteilten Spiele dieses Spieltyps entspricht der Anzahl der Spiele dieses Spieltyps.

2. Fuer $n \mid \|X\|$ (n teilt $\|X\|$) gilt, dass
$$\max\Big\{\ |\ \|X^a\| - \|X^b\|\ |\ :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}\Big\} = 0.$$
Sowie fuer $n \nmid \|X\|$ (n teilt $\|X\|$ nicht) gilt, dass
$$\max\Big\{\ |\ \|X^a\| - \|X^b\|\ |\ :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}\Big\} = 1.$$

3. Fuer alle $j \in \{1,\ 2,\ \dots\ ,\ k\}$ mit $n \mid a_j$ gilt, dass

$$\max\left\{\ |\ x_{a,j} - x_{b,j}\ |\ :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}\ \right\} = 0.$$

Sowie fuer $n \nmid a_j$ gilt, dass

$$\max\left\{\ |\ x_{a,j} - x_{b,j}\ |\ :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}\ \right\} = 1.$$

Satz 5: Satz über die optimale Verteilung

Eine optimale Verteilung erfuellt die gesuchten Eigenschaften (3.1), (3.2) und (3.3) der Aufgabenstellung.

Beweis:
Gegeben sei eine optimale Verteilung X.
Wir zeigen: Die Verteilung erfuellt die gesuchten Eigenschaften.

1. Die Eigenschaft (3.1) ist per Definition erfuellt (vgl. (4.1)).
2. Die Eigenschaft (3.2) fordert, dass $\max(\Omega)$ minimiert wird, wobei

$$\Omega := \left\{\ |\ \|X^a\| - \|X^b\|\ |\ :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}\right\}$$

Dabei gilt fuer alle $a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}$, dass $|\ \|X^a\| - \|X^b\|\ | \geq 0$. Somit gilt fuer alle $x \in \Omega$, dass $x \geq 0$. Daraus folgt, dass im allgemeinen $\max(\Omega) \geq 0$ gilt. Es werden zwei Faelle betrachtet:

(a) $n \mid \|X\|$.
Da eine optimale Verteilung gegeben ist, folgt nach (4.2) $\max(\Omega) = 0$. Wegen $\max(\Omega) \geq 0$ wird $\max(\Omega)$ minimiert.

(b) $n \nmid \|X\|$.
Da eine optimale Verteilung gegeben ist, folgt nach (4.2) $\max(\Omega) = 1$.
Wir zeigen: $\max(\Omega) \neq 0$. Daraus folgt sofort, dass $\max(\Omega)$ minimiert wird.
Angenommen, es waere $\max(\Omega) = 0$. Dann gilt fuer alle $a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}$, dass

$$\begin{aligned} &|\ \|X^a\| - \|X^b\|\ | = 0 \\ \iff &\|X^a\| - \|X^b\| = 0 \\ \iff &\|X^a\| = \|X^b\| \end{aligned}$$

Somit existiert ein $l \in \mathbb{N}$ mit $\|X^i\| = l$ fuer alle $i \in \{1,\ 2,\ \dots\ ,\ n\}$. Dann gilt weiter:

$$\|X\| = \sum_{i=1}^{n} \|X^i\| = \sum_{i=1}^{n} l = n \cdot l$$

Daraus folgt: $n \mid \|X\|$. Somit besteht ein Widerspruch zu $n \nmid \|X\|$. Somit war die Annahme, dass $\max(\Omega) = 0$ falsch und $\max(\Omega)$ wird durch die optimale Verteilung minimiert.

3. Die Eigenschaft (3.3) fordert, dass fuer jeden Spieltypen $j \in \{1,\ 2,\ \dots\ ,\ k\}$ $\max(\Lambda_j)$ minimiert wird, wobei

$$\Lambda_j := \left\{\ |\ x_{a,j} - x_{b,j}\ |\ :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}\ \right\}.$$

Erneut gilt fuer alle $x \in \Lambda_j$ $x \geq 0$, da fuer alle $a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}$ gilt, dass $|\ x_{a,j} - x_{b,j}\ | \geq 0$, sodass im allgemeinen $\max(\Lambda_j) \geq 0$ gilt.
Wieder betrachten wir zwei Faelle:

(a) $n \mid a_j$.
Da eine optimale Verteilung gegeben ist, folgt nach (4.3) $\max(\Lambda_j) = 0$. Wegen $\max(\Lambda_j) \geq 0$, wird $\max(\Lambda_j)$ minimiert.

(b) $n \nmid a_j$.
Da eine optimale Verteilung gegeben ist, folgt nach (4.3) $\max(\Lambda_j) = 1$.

Wir zeigen: $\max(\Lambda_j) \neq 0$. Daraus folgt sofort, dass $\max(\Lambda_j)$ minimiert wird.
Angenommen, es waere $\max(\Lambda_j) = 0$. Dann gilt fuer alle $a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}$, dass

$$\begin{aligned} &\mid x_{a,j} - x_{b,j} \mid = 0 \\ \iff &x_{a,j} - x_{b,j} = 0 \\ \iff &x_{a,j} = x_{b,j} \end{aligned}$$

Somit existiert ein $p \in \mathbb{N}$ mit $x_{i,j} = p$ fuer alle $i \in \{1,\ 2,\ \dots\ ,\ n\}$. Dann gilt weiter:

$$\|X_j\| = \sum_{i=1}^{n} x_{i,j} = \sum_{i=1}^{n} p = n \cdot p$$

Daraus folgt: $n \mid \|X_j\|$. Somit besteht ein Widerspruch zu $n \nmid \|X_j\|$. Somit war die Annahme, dass $\max(\Lambda_j) = 0$ falsch und $\max(\Lambda_j)$ wird durch die optimale Verteilung minimiert.

Satz 6: Existenz einer optimalen Verteilung

Eine optimale Verteilung nach Definition 4 existiert.

Beweis:
Fuer die n Wundertueten und k Spieltypen mit a_j Spielen des Spieltyps j fuer $j \in \{1,\ 2,\ \dots\ ,\ k\}$ wird nun ein Verfahren vorgestellt, dass eine Verteilung X konstruiert, sodass X eine optimale Verteilung nach Definition 4 ist.
Zu Beginn sei

$$X \coloneqq \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

sodass fuer $i \in \{1,\ 2,\ \dots\ ,\ n\}$ und fuer $j \in \{1,\ 2,\ \dots\ ,\ k\}$ gilt, dass $x_{i,j} = 0$.
Zunaechst sei $s_1 \coloneqq 1$. Nun wird nacheinander fuer jeden Spieltyp $j = 1,\ 2,\ \dots\ ,\ k$ die Matrix in der j-ten Spalte veraendert. Dies entspricht einer Verteilung der Spiele des Typs j auf die n Wundertueten.
Fuer den j-ten Durchlauf seien $g_j,\ r_j \in \mathbb{N}$ mit $0 \leq r_j < n$ und $a_j = g_j \cdot n + r_j$. Dann sind g_j und r_j eindeutig bestimmt und es gilt $g_j = \lfloor \frac{a_j}{n} \rfloor$ und $r_j = a_j \bmod n$. Weiter geschehen zwei Aenderungen der Matrix:

1. Die Eintraege $x_{1,j},\ x_{2,j},\ \dots\ ,\ x_{n,j}$ der j-ten Spalte der Matrix werden je auf g_j gesetzt. Dies entspricht einer Verteilung mit g_j Spielen des Typs j pro Wundertuete.

2. Die restlichen r_j Spiele des Typs j werden wie folgt auf die n Wundertueten (Zeilen der Matrix) verteilt: Der Eintrag $x_{s_j,j}$ der Zeile s_j und Spalte j wird um eins erhoeht und s_j wird um eins erhoeht. Ist $s_j = n$ wird s_j auf 1 gesetzt. Dies wird r_j mal wiederholt.
   D.h., die restlichen r_j Spiele des Typs j werden mit je einem Spiel auf r_j Wundertueten, angefangen bei s_j verteilt, wobei bei der ersten Wundertuete weitergemacht wird, sofern man die letzte erreicht hat. Anschhlieszend wird s_{j+1} als der letzte Wert von s_j definiert.

Dann wird mit dem naechsten Durchlauf fortgefahren, oder das Verfahren ist beendet.

Dieses Verfahren erzeugt in der Tat eine optimale Verteilung:
Nun werden die Teile der Definition einer optimalen Verteilung nach Definition 4 nachgewiesen.

1. Teil (4.1). Es soll gelten, dass fuer jeden Spieltypen $j \in \{1,\ 2,\ \dots\ ,\ k\}$ gilt, dass $\|X_j\| = a_j$.
   Das vorgestellte Verfahren veraendert die Spalte j ausschlieszlich im j-ten Durchlauf. Es sei $a_j = g_j \cdot n + r_j$ fuer $g_j,\ r_j \in \mathbb{N}$ mit $0 \leq r_j < n$ (g_j und r_j sind eindeutig!). Das Verfahren setzt zunaechst jeden der n Eintraege der Spalte j auf g_j. Weiter werden genau r_j Eintraege der Spalte um 1 erhoeht. Somit gilt fuer die Summe $\|X_j\|$ der Eintraege der j-ten Spalte:

$$\|X_j\| = g_j \cdot n + r_j = a_j$$

2. Teil (4.2). Es sei

$$\Omega \coloneqq \Big\{ \mid \|X^a\| - \|X^b\| \mid :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\} \Big\}$$

   Fuer diesen Teil wird eine Fallunterscheidung getroffen:

(a) Fuer $n \mid \|X\|$ soll gelten, dass $\max(\Omega) = 0$.
Es gelte $n \mid \|X\|$. Es gilt

$$\|X\| = \sum_{j=1}^{k} \|X_j\| = \sum_{j=1}^{k} g_j \cdot n + r_j = \sum_{j=1}^{k} g_j \cdot n + \sum_{j=1}^{k} r_j = n \cdot \sum_{j=1}^{k} g_j + \sum_{j=1}^{k} r_j$$

Somit folgt:

$$n \text{ teilt } n \cdot \sum_{j=1}^{k} g_j + \sum_{j=1}^{k} r_j, \text{ sodass auch } n \text{ teilt } \sum_{j=1}^{k} r_j$$

Es sei $a \in \{1,\ 2,\ \dots\ ,\ n\}$ beliebig. Dann wird $\|X^a\|$ in jedem Durchgang j um p_j erhoeht (erste Aenderung). Somit ergibt sich die Summe

$$\sum_{j=1}^{k} p_j$$

fuer jede Zeile. Sonst wird $\|X^a\|$ ausschlieszlich durch die zweite Aenderung veraendert. Undzwar werden bei dieser die Werte $\|X^i\|$ fuer die folgenden Zeilen i jeweils um 1 erhoeht:

$$1,\ 2,\ \dots\ ,\ r_1,\ r_1+1,\ r_1+2,\ \dots\ ,\ r_1+r_2,\ r_1+r_2+1,\ \dots\ ,\ \sum_{j=1}^{k} r_j$$

dabei wird jede der Zeilen Modulo n und plus 1 gerechnet.
Da die Anzahl dieser Zeilen durch n teilbar ist, ist jede Zeile gleich oft enthalten.
Somit gilt fuer beliebige $a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}$

$$\begin{aligned} & \|X^a\| = \|X^b\| \\ \Longleftrightarrow\ & \|X^a\| - \|X^b\| = 0 \\ \Longleftrightarrow\ & \mid \|X^a\| - \|X^b\| \mid = 0. \end{aligned}$$

Daraus folgt sofort $\max(\Omega) = 0$.

(b) Fuer $n \nmid \|X\|$ soll gelten, dass $\max(\Omega) = 1$.
Dieser Teil des Beweises ist analog zum Vorherigen und trivial. Somit wird er dem Leser als Uebungsaufgabe ueberlassen.

3. Teil (4.3). Fuer $j \in \{1,\ 2,\ \dots\ ,\ k\}$ sei

$$\Lambda_j := \left\{ \mid x_{a,j} - x_{b,j} \mid :\ a,\ b \in \{1,\ 2,\ \dots\ ,\ n\} \right\}.$$

Fuer diesen Teil wird ebenfalls eine Fallunterscheidung getroffen:

(a) Fuer $n \mid a_j$ soll gelten, dass $\max(\Lambda_j) = 0$.
Es gelte $n \mid a_j$. Dann gilt $a_j = n \cdot p = n \cdot p + 0$ fuer ein eindeutiges $p \in \mathbb{N}$. Somit ist fuer Durchgang j des Verfahrens $g_j = p = \frac{a_j}{n} \in \mathbb{N}$ und $r_j = 0$. Somit gilt nach Durchlauf j, dass $x_{i,j} = p$ fuer alle $i \in \{1,\ 2,\ \dots\ ,\ n\}$, da $r = 0$.
Daraus folgt fuer beliebige $a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}$:

$$\begin{aligned} & x_{a,j} = x_{b,j} \\ \Longleftrightarrow\ & x_{a,j} - x_{b,j} = 0 \\ \Longleftrightarrow\ & \mid x_{a,j} - x_{b,j} \mid = 0 \end{aligned}$$

Also ist $\max(\Lambda_j) = 0$.

(b) Fuer $n \nmid a_j$ soll gelten, dass $\max(\Lambda_j) = 1$.
Es gelte $n \nmid a_j$. Dann gilt $a_j = n \cdot p + q$ fuer $p,\ q \in \mathbb{N}$ mit $0 < q < n$. Dabei sind p und q eindeutig und insbesondere $q > 0$. Im Durchlauf j ist $g_j = p$ und $r_j = q$. Somit gilt nach Durchlauf j des Verfahrens, dass genau $n - q$ Eintraege der Spalte j gleich p und genau q Eintraege gleich $p+1$ sind.
Dann gilt fuer beliebige $a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}$, dass $x_{a,j},\ x_{b,j} \in \{p,\ p+1\}$. Dann ist $\mid x_{a,j} - x_{b,j} \mid \leq 1$, sodass $\max(\Lambda_j) \leq 1$.
Weiter existiert wegen $q > 0$ mind. ein $a \in \{1,\ 2,\ \dots\ ,\ n\}$ mit $x_{a,j} = p+1$ und es existiert wegen $q < n$ mind. ein $b \in \{1,\ 2,\ \dots\ ,\ n\}$ mit $x_{b,j} = p$. Somit existieren $a,\ b \in \{1,\ 2,\ \dots\ ,\ n\}$ mit $\mid x_{a,j} - x_{b,j} \mid = 1$, sodass $\max(\Lambda_j) \geq 1$.
Also ist $\max(\Lambda_j) = 1$.

2. Lösungsvorschlag

Im Folgenden wird auf das Loesungsverfahren fur das zuvor definierte und analysierte Problem eingegangen. Dabei folgt der folgende Algorithmus direkt aus Satz 6.

Algorithmus 7:

Pseudocodealgorithmus zur Loesung des Problems.

Algorithmus 1 : Generate X

Input : $n,\ k \in \mathbb{N}$ und $(a_1,\ a_2,\ \dots\ ,\ a_k) \in \mathbb{N}^k$
Output : Die Matrix X

$X \leftarrow 0_{n,k}$ `// Matrix mit` n `Zeilen und` k `Spalten und den Eintraegen` $x_{i,j}$
$s \leftarrow 0$ `// Erste Zeile im naechsten Durchlauf`
for $j \leftarrow 1$ **to** k **do**
    $g \leftarrow \lfloor \frac{a_j}{n} \rfloor$
    $r \leftarrow a_j \bmod n$
    for $i \leftarrow 1$ **to** n **do**
        $x_{i,j} \leftarrow x_{i,j} + g$
    end for
    for $i \leftarrow 1$ **to** r **do**
        $x_{s,j} \leftarrow x_{s,j} + 1$
        $s \leftarrow (s + 1) \bmod n$
    end for
end for
return X

Korrektheit:

Dieser Algorithmus ist genau das in Satz 6 vorgestellt Verfahren, um eine optimale Loesung nach Definition 4 zu erhalten. Wie in Satz 5 gezeigt erfuellt eine solche Verteilung die gesuchten Eigenschaften der Aufgabenstellung. Somit ist die Korrektheit des Algorithmus gezeigt.

3. Analyse der Zeit- und Platzkomplexitaet der Lösungen und Vergleich der Lösungen

Nun wird die Zeit- und Platzkomplexitaet des Algorithmus 7 analysiert. Dazu wird angenommen, dass Ganzzahlen einen konstanten Platzverbrauch ($\mathcal{O}(1)$) und Operationen auf ihnen, wie Addition, Division und Modulo eine konstante Laufzeit ($\mathcal{O}(1)$) haben.

Platzkomplexitaet:
Da die Eingabe neben zwei Ganzzahlen ein k-Tupel entgegen nimmt, und bis auf die Matrix X mit n Zeilen und k Spalten, ansonsten nur weitere Ganzzahlen verwendet werden, ergibt sich die Platzkomplexitaet

$$\mathcal{O}(k) + \mathcal{O}(n \cdot k) = \mathcal{O}(n \cdot k).$$

Zeitkomplexitaet:
Die ersten beiden Zeilen des Algorithmus laufen in konstanter Zeit. Die Zeilen vier bis zwölf werden genau k mal durchlaufen. Dabei haben die Zeilen 4, 5, 7, 10 und 11 eine konstante Laufzeit, wobei die Zeile 7 genau n mal, und die Zeilen 10 und 11 genau r mal durchlaufen werden. Somit betraegt die Laufzeitkomplexitaet jedes Durchgangs je

$$\mathcal{O}(1) + \mathcal{O}(n) + \mathcal{O}(r) = \mathcal{O}(n) + \mathcal{O}(r).$$

Wegen $0 \leq r < n$ folgt

$$\mathcal{O}(n) + \mathcal{O}(r) = \mathcal{O}(n) + \mathcal{O}(n) = \mathcal{O}(n)$$

Somit ergibt sich die Laufzeitkomplexitaet

$$\mathcal{O}(1) + k \cdot \mathcal{O}(n) = \mathcal{O}(n \cdot k)$$

4. Implementierung

Nun wird auf die Implementierung der Loesungsideen eingegangen. Fuer die Implementierung des Algorithmus 7 wurde die Sprache C++ verwendet.
Das Programm bietet die Moeglichkeit eine Eingabe- und eine Ausgabendatei anzugeben. Aus der Eingabedatei werden anschlieszend die Eingaben eingelesen und in zwei Integern, sowie einem Vektor an integern gespeichert. Weiter wird der bereits vorgestellt Algorithmus verwendet, um die Matrix X zu berechnen. Diese wird durch einen zweidimensionalen Vektor dargestellt, der Integer enthaelt. Zu letzt wird der Vektor zeilenweise (d.h., Wundertueten-weise) ausgegeben.

5. Beispiele

Nun wird auf die Beispieleingaben der BwInf Website eingegangen. Die Ausgabedatei sind im Ordner mit dem Pfad „Junioraufgabe-1/Loesungen“ zu finden.
Dabei wird in jeder Zeile der aktuelle Rucksack i, gefolgt von der Verteilung der Spiele auf diesen Rucksack angegeben. Dazu werden durch Leerzeichen getrennt die Werte $x_{i,1}, x_{i,2}, \ldots, x_{i,k}$ angegeben.

5.1. Beispiel 0 - „wundertuete0.txt“

Das Beispiel der Aufgabenstellung ergibt folgende Ausgaben:

1. Rucksack: 2 1 1
2. Rucksack: 1 2 0
3. Rucksack: 1 1 1

5.2. Beispiel 1 - „wundertuete1.txt“

Das 1. Beispiel der BwInf Website ergibt folgende Ausgabe:

1. Rueksack: 3 1 2
2. Rueksack: 3 1 2
3. Rueksack: 3 1 2
4. Rueksack: 3 1 2
5. Rueksack: 3 1 2
6. Rueksack: 3 1 2

5.3. Beispiel 2 - „wundertuete2.txt“

Das 2. Beispiel der BwInf Website ergibt folgende Ausgabe:

1. Wundertuete: 2 1 0 0
2. Wundertuete: 1 1 1 0
3. Wundertuete: 1 1 1 0
4. Wundertuete: 1 1 1 0
5. Wundertuete: 1 1 0 1
6. Wundertuete: 1 1 0 1
7. Wundertuete: 1 1 0 1
8. Wundertuete: 1 1 0 1
9. Wundertuete: 1 1 0 1

5.4. Beispiel 3 - „wundertuete3.txt“

Das 3. Beispiel der BwInf Website ergibt folgende Ausgabe:

1. Wundertuete: 1 1 0 0 0
2. Wundertuete: 1 1 0 0 0
3. Wundertuete: 0 1 1 0 0
4. Wundertuete: 0 1 1 0 0

5. Wundertuete: 0 1 1 0 0
6. Wundertuete: 0 1 1 0 0
7. Wundertuete: 0 1 1 0 0
8. Wundertuete: 0 1 1 0 0
9. Wundertuete: 0 1 0 1 0
10. Wundertuete: 0 1 0 1 0
11. Wundertuete: 0 1 0 0 1

5.5. Beispiel 4 - „wundertuete4.txt“

Das 4. Beispiel der BwInf Website ergibt folgende Ausgabe:

1. Wundertuete: 2 6 3 5 31 5
2. Wundertuete: 2 6 3 5 30 6
3. Wundertuete: 2 6 3 5 30 6
4. Wundertuete: 2 6 2 6 30 6
5. Wundertuete: 1 7 2 6 30 6
6. Wundertuete: 1 7 2 6 30 5
7. Wundertuete: 1 7 2 6 30 5
8. Wundertuete: 1 7 2 6 30 5
9. Wundertuete: 1 7 2 6 30 5
10. Wundertuete: 1 7 2 6 30 5
11. Wundertuete: 1 7 2 6 30 5
12. Wundertuete: 1 7 2 6 30 5
13. Wundertuete: 1 7 2 6 30 5
14. Wundertuete: 1 7 2 6 30 5
15. Wundertuete: 1 7 2 6 30 5
16. Wundertuete: 1 7 2 5 31 5
17. Wundertuete: 1 6 3 5 31 5

5.6. Beispiel 5 - „wundertuete5.txt“

Das 5. Beispiel der BwInf Website ergibt folgende Ausgabe:

1. Wundertuete: 1 0 2 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1
2. Wundertuete: 1 0 2 1 2 0 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1
3. Wundertuete: 1 0 2 1 2 0 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1
4. Wundertuete: 1 0 2 1 2 0 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1
5. Wundertuete: 1 0 2 1 2 0 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1
6. Wundertuete: 1 0 2 1 2 0 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1
7. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1
8. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1

9. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1
10. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 2 1
11. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
12. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
13. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
14. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
15. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
16. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
17. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
18. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
19. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
20. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
21. Wundertuete: 1 0 2 1 1 1 2 0 1 1 1 1 0 1 2 1 1 1 1 2 0 1 2
22. Wundertuete: 1 0 2 1 1 1 2 0 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
23. Wundertuete: 1 0 2 1 1 1 2 0 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
24. Wundertuete: 1 0 2 1 1 1 2 0 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
25. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
26. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
27. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
28. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
29. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
30. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
31. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
32. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
33. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 1 1 2 1 1 1 2 0 1 2
34. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 0 2 2 1 1 1 2 0 1 2
35. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 0 2 2 1 1 1 2 0 1 1
36. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 0 2 2 1 1 1 2 0 1 1
37. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 1 0 0 2 2 1 1 1 2 0 1 1
38. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 2 0 1 1
39. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 2 0 1 1
40. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 2 0 1 1
41. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 2 0 1 1
42. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 2 0 1 1
43. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1
44. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1
45. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1

46. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1
47. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1
48. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1
49. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1
50. Wundertuete: 1 0 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1
51. Wundertuete: 0 1 2 1 1 1 1 1 1 0 2 0 1 0 2 2 1 1 1 1 1 1 1
52. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
53. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
54. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
55. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
56. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
57. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
58. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
59. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
60. Wundertuete: 0 1 2 1 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
61. Wundertuete: 0 1 1 2 1 1 1 1 1 0 1 1 1 0 2 2 1 1 1 1 1 1 1
62. Wundertuete: 0 1 1 2 1 1 1 1 1 0 1 1 0 1 2 2 1 1 1 1 1 1 1
63. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
64. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
65. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
66. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
67. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
68. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
69. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
70. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
71. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 1 1 1
72. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 0 2 1
73. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 0 2 1
74. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 0 2 1
75. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 0 2 1
76. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 0 2 1
77. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 1 1 1 1 0 2 1
78. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 1 1 0 2 1
79. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 1 1 0 2 1
80. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 1 1 0 2 1
81. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 1 1 0 2 1
82. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 0 2 0 2 1

83. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 0 2 0 2 1
84. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 0 2 0 2 1
85. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 0 2 0 2 1
86. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 2 0 2 0 2 1
87. Wundertuete: 0 1 1 2 1 1 1 1 0 1 1 1 0 1 2 2 0 1 1 2 0 2 1
88. Wundertuete: 0 1 1 1 2 1 1 1 0 1 1 1 0 1 2 2 0 1 1 2 0 2 1
89. Wundertuete: 0 1 1 1 2 1 1 1 0 1 1 1 0 1 2 2 0 1 1 2 0 2 1
90. Wundertuete: 0 1 1 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1
91. Wundertuete: 0 1 1 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1
92. Wundertuete: 0 0 2 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1
93. Wundertuete: 0 0 2 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1
94. Wundertuete: 0 0 2 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1
95. Wundertuete: 0 0 2 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1
96. Wundertuete: 0 0 2 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1
97. Wundertuete: 0 0 2 1 2 0 2 0 1 1 1 1 0 1 2 2 0 1 1 2 0 2 1

6. Quellcode

Nun folgt der wichtigste Teil der Loesung implementiert in C++.

```cpp
// Calculate the result - Per bag: Number of games per type
vector<vector<int>> result(bags, vector<int>(amount_types, 0));
int index = 0; // Keep track of the current bag

// For each type: Calculate number of games per bag + rest
for (int i = 0; i < amount_types; i++) {
    // Calculate the number of games each bag can get of current type using
    // the total number of games of current type. + Add that to each bag
    int number_games = types[i];
    int games_per_bag = number_games / bags; // Integer division!

  for (int j = 0; j < bags; j++) {
        result[j][i] += games_per_bag;
    }

    // Now, add the left over games of current type
    int rest = number_games % bags;
    for (int j = 0; j < rest; j++) {
        result[index][i] += 1;
        index = (index + 1) % bags;
    }
}
```