Aufgabe 5: Wichteln

Team-ID: 01144

Team: Legends

Bearbeiter/-innen dieser Aufgabe:
Christoph Waffler

23. November 2020

Lösungsidee 1
Umsetzung 4
Beispiele 6
wichteln1.txt 6
wichteln2.txt 7
wichteln3.txt 8
wichteln4.txt 9
wichteln5.txt 10
wichteln6.txt 11
wichteln7.txt 11
Quellcode 12

Lösungsidee

Ziel dieser Aufgabe ist es, für Pirmins Klasse eine bestmögliche Verteilung der Wichtelgeschenke zu erzielen. Jeder der Schüler hat einen Erst-, Zweit- und Drittwunsch bei der Auswahl der Geschenke.

Es gilt die Anzahl der Erstwünsche zu maximieren. Bei gleicher Anzahl der Erstwünsche werden die Zweitwünsche maximiert usw.

Ein wichtiger Schritt, um das Problem zu lösen, ist die Wünsche anders zu speichern:
Für jedes Geschenk wird jeweils eine Menge mit Bewerbern für das Geschenk als Erstwunsch (*M1*), Zweitwunsch (*M2*) und Drittwunsch (*M3*) erstellt.
Ein Bewerber ist eine Person, die potenziell das Geschenk erhalten könnte, um die Anzahl der Erst-, Zweit- bzw. Drittwünsche zu erhöhen.

So kann die Menge *M2* für das Geschenk Nr. 3 beispielsweise die Personen Nr. 1 und 2 enthalten. Das bedeutet, dass Person Nr. 1 und 2 denselben Zweitwunsch – das Geschenk Nr. 3 – besitzen.

Anschließend wird folgendermaßen vorgegangen:

1. Alle Geschenke, die mindestens einen Bewerber als Erstwunsch haben, d.h. die Menge *M1* enthält mindestens ein Element, werden sicher als Erstwunsch verschenkt werden.
   Das bedeutet im Umkehrschluss, dass diese Geschenke nicht als Zweit- oder Drittwunsch verschenkt werden können, wodurch die Mengen *M2* und *M3* dieser Geschenke gelöscht werden.

2. Prüfe für jedes Geschenk, ob sie exakt einen Bewerber als Erstwunsch besitzen, d.h. die Menge *M1* dieses Geschenks enthält genau ein Element.
   Ist dies der Fall, so erhält diese Person das Geschenk und *M1* wird gelöscht.
   Außerdem wird die Person aus allen anderen Mengen gelöscht, da sie ein Geschenk erhalten und somit kein potenzieller Bewerber mehr ist.

3. Nun werden jene Geschenke überprüft, die mehrere Bewerber besitzen, d.h. *M1* dieses Geschenks enthält mehr als ein Element. Von *M1* wird jene Person ermittelt, die optimal für das Geschenk geeignet ist.

   a. Für jede Person aus *M1* werden die Anzahl der Konkurrenten beim Zweitwunsch ermittelt: Falls diese Menge *M2* vorhanden ist, in der sich die Person befindet, wird die Anzahl der Elemente (= Personen) ermittelt.

   b. Falls die aktuelle Person theoretisch keinen Zweitwunsch erfüllt bekommen kann, werden die Anzahl der Elemente (= Personen) von *M3* ermittelt, in der sich die aktuelle Person befindet, falls eine solche Menge vorhanden ist.

   c. Kann der Person theoretisch keinen Zweit- oder Drittwunsch erfüllt werden, so wird ihr der Erstwunsch erfüllt.

   Die geeignetste Person wird wie folgt ausgewählt:
   - Kein Zweit- oder Drittwunsch erfüllbar → Person bekommt Erstwunsch
   - Anzahl Konkurrenten Zweitwunsch oder Drittwunsch am größten → Person bekommt Erstwunsch
   - alle Personen gleiche Anzahl an Konkurrenten → beliebige Person bekommt Erstwunsch

   Jene Person, die das Geschenk erhält, wird aus allen Mengen entfernt, in denen die Person als Element gelistet ist.

4. Schritt 1 wird nun für die Zweitwünsche angewendet:
   Alle Geschenke, die mindestens einen Bewerber als Zweitwunsch haben, d.h. die Menge *M2* der Geschenke enthalten mindestens ein Element, werden sicher als Zweitwusch verschenkt werden.
   Das bedeutet im Umkehrschluss, dass diese Geschenke nicht als Drittwunsch verschenkt werden können, wodurch die Menge *M3* dieser Geschenke gelöscht werden.

5. Prüfe für jedes Geschenk, ob sie exakt einen Bewerber als Zweitwunsch besitzen, d.h. die Menge *M2* dieses Geschenks enthält genau ein Element.
   Ist dies der Fall, so erhält diese Person das Geschenk und *M2* wird gelöscht.
   Ebenfalls wird diese Person aus allen anderen Mengen gelöscht.

6. Schritt 3 wird nun für die Zweitwünsche angewendet:
   Wähle diejenige Person als Empfänger des Geschenks aus, dessen Drittwunsch nicht erfüllt werden kann oder dessen Anzahl an Konkurrenten für den Drittwunsch am größten ist. Sind alle Personen gleich geeignet, wähle irgendeine Person aus.
   Lösche die ausgewählte Person ebenfalls aus allen anderen Mengen.

7. Prüfe für alle Geschenke, ob sie noch eine Menge *M3* besitzen und weise das Geschenk einem beliebigen Element (Person) aus *M3* zu.

8. Alle Personen, die noch kein Geschenk erhalten haben, werden die übrigen, noch nicht verwendeten Geschenke erhalten.

Die Verteilung der Geschenke ist nun vollständig und wird ausgegeben.

Umsetzung

Die Lösungsidee wird in C++ implementiert. (Hinweis zum Ausführen des Programms: die Programmierung erfolgte mit dem OSX Betriebssystem von Apple).

Zu Beginn des Programms wurde die Methode *delete_element_vvi(vvi& arr, int e)* implementiert, mit der das Element *e* aus allen „Untervectoren“ vom Vector<Vector<int>> *arr* gelöscht werden. Diese Methode wird später verwendet werden.

Die Main-Methode:

Mit *cin* erfolgt das Einlesen der Daten im gegebenen BwInf-Format. Von allen Zahlen (außer der ersten Zahl *n*) wird 1 abgezogen, damit die Indices später zusammenpassen, da Arrays mit 0 beginnen und im BwInf-Format beginnt die Nummerierung mit 1.
Alle Wünsche der *n* Personen werden im Vector<Vector<int>> *arr* gespeichert. Der Erstwunsch der Person *i* in *arr[i][0],* der Zweitwunsch in *arr[i][1]* und der Drittwunsch in *arr[i][2].*

Anschließend werden die 3 Vector<Vector<int>> *erste_geschenke*, *zweite_geschenke* und *dritte_geschenke* deklariert, in denen für jedes Geschenk eine Liste mit Bewerbern gespeichert wird. So enthält z.B. *erste_geschenke[2]* einen Vector mit allen Personen, die sich das Geschenk Nr. 2 als Erstwunsch wünschen.

Im Vector<int> *verteilung* mit der Größe *n* wird die finale Verteilung der Geschenke gespeichert, z.B. enthält *verteilung[1]* das Geschenk, das Person Nr. 1 erhält.
Dieser wird mit *-1* für jedes Element initialisiert.

Nun wird mithilfe einer for-Schleife Schritt 1 der Lösungsidee durchgeführt und für jedes Geschenk *i* überprüft, ob *erste_geschenke[i]* mindestens ein Element enthält. Ist dies der Fall, so wird jeweils der Vector<int> in *zweite_geschenke[i]* und *dritte_geschenke[i]* gelöscht.

Anschließend folgt Schritt 2 der Lösungsidee: Für jedes Geschenk *i* wird überprüft, ob der Vector in *erste_geschenke[i]* ein Element enthält. Ist dies der Fall, so wird mit *person = erste_geschenke[i][0]* die Nr. der Person ermittelt und mit *verteilung[person] = i* das Geschenk der Person festgelegt.
Daraufhin wird die Methode *delete_element_vvi* aufgerufen um *person* jeweils aus dem Vector *zweite_geschenke* und *dritte_geschenke* zu löschen.

Jetzt wird Schritt 3 der Lösungsidee ausgeführt: Falls der Vector in *erste_geschenk[i]* mehrere Elemente enthält, wird für jede Person *p* in *erste_geschenke[i]* die Anzahl der Konkurrenten um den Zweitwunsch ermittelt, was mit einer weiteren for-Schleife umgesetzt wird. Die *integer*-Variablen *max_value* und *max_person_idx* enthalten die aktuelle maximale Anzahl an Konkurrenten und die dazugehörige Nummer dieser Person.
Da *max_value* mit *-1* initialisiert wurde, kann anschließend überprüft werden, ob sich der Wert verändert hat. Ist er immer noch *-1*, so wird die Anzahl der Konkurrenten um den Drittwunsch bestimmt.
Ist *max_value* daraufhin immer noch *-1*, so kann der Person kein Zweit- oder Drittwunsch erfüllt werden und *max_value* wird auf *INF* gesetzt, um den Erstwunsch zu erhalten.

Ist diese Prozedur für alle *p* aus *erste_geschenke[i]* fertig, so kann der Wert *i* (Nr. des Geschenks) derjenigen Person *p_max* im Vector *verteilung* zugewiesen werden, die den höchsten *max_value* hat.
Anschließend wird der Vector in *erste_geschenke[i]* gelöscht.
Dann wird mithilfe der Methode *delete_element_vvi p_max* jeweils aus dem Vector *zweite_geschenke* und *dritte_geschenke* gelöscht.

Daraufhin folgt Schritt 4, indem für alle Geschenke *i* überprüft werden, ob deren Vectoren *zweite_geschenke[i]* mindestens ein Element enthalten. Ist dies der Fall so wird der Vector in *dritte_geschenke[i]* gelöscht.

Nun wird Schritt 5 der Lösungsidee ausgeführt, wodurch alle Geschenke die nur einen Bewerber haben diesen Bewerbern zugewiesen werden. *Analog zur Implementierung für Schritt 2.*

Die Implementierung für Schritt 6 der Lösungsidee ist ebenfalls analog zur Implementierung für Schritt 3.

Zur Implementierung von Schritt 7 der Lösungsidee werden nun mit einer for-Schleife alle Geschenke überprüft, ob der Vector für das aktuelle Geschenk *i* in *dritte_geschenke[i]* mindestens ein Element enthält. Ist dies der Fall, so erhält die erste Person *e* in *dritte_geschenke[i]* mit *verteilung[e] = i* das Geschenk.

Anschließend werden in einem Stack *s* alle Geschenke gespeichert, die noch nicht verschenkt wurden. Die Elemente in *s* werden nun allen Personen im Vector *verteilung* zugewiesen.

Daraufhin hat nun jede Person ein Geschenk und es wird für jede Person *i* mit *verteilung[i]* das entsprechende Geschenk über *cout* ausgegeben.

Zum Schluss des Programms wird die Anzahl der erfüllten Erst-, Zweit- und Drittwünsche ermittelt und neben der Anzahl der nichterfüllten Wünsche ebenfalls über *cout* ausgegeben.

Beispiele

wichteln1.txt

```
>> Eingabe der Wünsche im BwInf-Format
10
 2 10  6
 2  7  3
 4  7  1
 3  4  9
 3  7  9
 4  3  2
 7  6  2
10  2  4
 9  8  1
 4  9  6


> Ausgabe der Verteilung:
Schüler Nr. -> Geschenk Nr.
1 -> 6
2 -> 2
3 -> 1
4 -> 3
5 -> 8
6 -> 4
7 -> 7
8 -> 10
9 -> 9
10 -> 5
Anzahl an erfüllten 1. Wünschen: 6
Anzahl an erfüllten 2. Wünschen: 0
Anzahl an erfüllten 3. Wünschen: 2
Anzahl an Geschenk, die nicht gewünscht waren: 2
```

wichteln2.txt

>> Eingabe der Wünsche im BwInf-Format
10
4 6 5
5 4 6
6 4 5
6 4 5
5 4 6
4 6 5
4 5 6
5 4 6
6 5 4
4 5 6

> Ausgabe der Verteilung:
Schüler Nr. -> Geschenk Nr.
1 -> 4
2 -> 5
3 -> 6
4 -> 10
5 -> 9
6 -> 8
7 -> 7
8 -> 3
9 -> 2
10 -> 1
Anzahl an erfüllten 1. Wünschen: 3
Anzahl an erfüllten 2. Wünschen: 0
Anzahl an erfüllten 3. Wünschen: 0
Anzahl an Geschenk, die nicht gewünscht waren: 7

wichteln3.txt

(Eingabe von wichteln3.txt)

> Ausgabe der Verteilung:
Schüler Nr. -> Geschenk Nr.
1 -> 2
2 -> 20
3 -> 29
4 -> 8
5 -> 25
6 -> 3
7 -> 24
8 -> 12
9 -> 4
10 -> 28
11 -> 22
12 -> 18
13 -> 14
14 -> 23
15 -> 26
16 -> 30
17 -> 9
18 -> 7
19 -> 16
20 -> 10
21 -> 19
22 -> 13
23 -> 27
24 -> 15
25 -> 17
26 -> 11
27 -> 5
28 -> 21
29 -> 6
30 -> 1
Anzahl an erfüllten 1. Wünschen: 15
Anzahl an erfüllten 2. Wünschen: 6
Anzahl an erfüllten 3. Wünschen: 1
Anzahl an Geschenk, die nicht gewünscht waren: 8

wichteln4.txt

(Eingabe von wichteln4.txt)

> Ausgabe der Verteilung:
Schüler Nr. -> Geschenk Nr.
1 -> 28
2 -> 21
3 -> 14
4 -> 26
5 -> 6
6 -> 5
7 -> 24
8 -> 16
9 -> 25
10 -> 19
11 -> 23
12 -> 12
13 -> 7
14 -> 20
15 -> 2
16 -> 11
17 -> 1
18 -> 27
19 -> 17
20 -> 13
21 -> 22
22 -> 10
23 -> 15
24 -> 9
25 -> 30
26 -> 4
27 -> 18
28 -> 8
29 -> 3
30 -> 29
Anzahl an erfüllten 1. Wünschen: 15
Anzahl an erfüllten 2. Wünschen: 3
Anzahl an erfüllten 3. Wünschen: 3
Anzahl an Geschenk, die nicht gewünscht waren: 9

wichteln5.txt

(Eingabe von wichteln5.txt)

> Ausgabe der Verteilung:
Schüler Nr. -> Geschenk Nr.
1 -> 25
2 -> 6
3 -> 7
4 -> 19
5 -> 27
6 -> 2
7 -> 4
8 -> 18
9 -> 29
10 -> 28
11 -> 14
12 -> 13
13 -> 24
14 -> 15
15 -> 10
16 -> 8
17 -> 26
18 -> 23
19 -> 22
20 -> 3
21 -> 1
22 -> 21
23 -> 20
24 -> 11
25 -> 16
26 -> 9
27 -> 30
28 -> 12
29 -> 17
30 -> 5
Anzahl an erfüllten 1. Wünschen: 13
Anzahl an erfüllten 2. Wünschen: 1
Anzahl an erfüllten 3. Wünschen: 7
Anzahl an Geschenk, die nicht gewünscht waren: 9

wichteln6.txt

(Gesamte Ein- und Ausgabe der Kommandozeile siehe Textdatei ‚wichteln6-in-out.txt')
...

Anzahl an erfüllten 1. Wünschen: 37
Anzahl an erfüllten 2. Wünschen: 2
Anzahl an erfüllten 3. Wünschen: 21
Anzahl an Geschenk, die nicht gewünscht waren: 30

wichteln7.txt

(Gesamte Ein- und Ausgabe der Kommandozeile siehe Textdatei ‚wichteln7-in-out.txt')
...

Anzahl an erfüllten 1. Wünschen: 541
Anzahl an erfüllten 2. Wünschen: 88
Anzahl an erfüllten 3. Wünschen: 69
Anzahl an Geschenk, die nicht gewünscht waren: 302

Quellcode

(Anmerkung: Es wurde der gesamte Quellcode eingefügt.)

```
#include <bits/stdc++.h>

using namespace std;

#define vi vector<int>
#define vvi vector<vi>
#define vs vector<string>
#define vb vector<bool>

const int INF = numeric_limits<int>::max()/2;

// Löschen des gegebenen Elements aus allen Untervectoren von arr
vvi delete_element_vvi(vvi& arr, int e) {
    for (int i = 0; i < arr.size(); ++i) {
        auto it = find(arr[i].begin(), arr[i].end(), e);
        if (it != arr[i].end()) {
            arr[i].erase(it);
        }
    }
    return arr;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);

    cout << ">> Eingabe der Wünsche im BwInf-Format\n";

    // Einlesen der Daten
```

```
int n;
cin >> n;

// Wünsche der n Personen werden gespeichert
// Erstwunsch in arr[i][0]
// Zweitwunsch in arr[i][1]
// Drittwunsch in arr[i][2]
vvi arr(n, vi(3));

// Jeder eingelesene Index wird um 1 reduziert,
// da Arrays bei 0 starten
for (int i = 0; i < n; ++i) {
    int first, sec, third;
    cin >> first >> sec >> third;
    arr[i][0] = first - 1;
    arr[i][1] = sec - 1;
    arr[i][2] = third - 1;
}

// Für die drei verschiedenen Wünsche gibt es für jedes Geschenk
// eine Liste mit Bewerbern, die sich das Geschenk wünschen
// z.B. erste_geschenke[2] ist ein Vector mit allen Bewerbern für das Geschenk Nr. 2
vvi erste_geschenke(n, vi());
vvi zweite_geschenke(n, vi());
vvi dritte_geschenke(n, vi());

// Für jedes Geschenk werden die potentiellen Empfänger gespeichert
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < 3; ++j) {
        if (j == 0) erste_geschenke[arr[i][j]].push_back(i);
        else if (j == 1) zweite_geschenke[arr[i][j]].push_back(i);
```

```
            else if (j == 2) dritte_geschenke[arr[i][j]].push_back(i);
        }
    }

    // in verteilung wird die finale Verteilung der Geschenke gespeichert
    // verteilung[2] beinhaltet das Geschenk, das Person Nr. 2 erhält
    vi verteilung(n, -1);

    // Alle Geschenke mit mindestens einen Bewerber als Erstwunsch werden sicher als Erstwunsch vergeben werden
    // Geschenk kann somit nicht als Zweit- oder Drittwunsch verwendet werden
    for (int i = 0; i < n; ++i) {
        if (erste_geschenke[i].size() >= 1) {
            zweite_geschenke[i].clear();
            dritte_geschenke[i].clear();
        }
    }

    // Zuweisung aller Geschenke, die nur einen Bewerber haben
    // hier: Erstwunsch
    for (int i = 0; i < n; ++i) {

        // Falls das Geschenk nur einen Bewerber hat wird es sofort zugewiesen
        if (erste_geschenke[i].size() == 1) {
            int person = erste_geschenke[i][0];
            // Die Person bekommt das Geschenk sicher
            verteilung[person] = i;

            // Liste für das aktuelle Geschenk wird gelöscht, da es vergeben wurde
            erste_geschenke[i].clear();
```

```
            // Person ist kein potenzieller Bewerber für andere Geschenke mehr
            zweite_geschenke = delete_element_vvi(zweite_geschenke, person);
            dritte_geschenke = delete_element_vvi(dritte_geschenke, person);

        }
    }

    // Ermitteln der optimalsten Empfänger für alle Geschenke
    // hier: Erstwunsch
    for (int i = 0; i < n; ++i) {
        // Geschenk hat mehrere Bewerber
        if (erste_geschenke[i].size() > 1) {
            // Diejenige Person mit den meisten Konkurrenten beim Zweitwunsch erhalten ihren
Erstwunsch
            int max_value_ges = -1;
            int max_person_idx_ges;

            // Iteration über alle unterschiedliche Bewerber p
            for (int p : erste_geschenke[i]) {

                // mit max_value werden die Anzahl der Konkurrenten beim Zweitwunsch oder ggf.
Drittwunsch gespeichert
                int max_value = -1;
                int max_person_idx;

                // Zweitwünsche werden überprüft
                for (int j = 0; j < n;++j) {
                    if (find(zweite_geschenke[j].begin(), zweite_geschenke[j].end(), p) !=
zweite_geschenke[j].end()) {
```

```
                if (zweite_geschenke[j].size() > max_value) {
                    max_value = zweite_geschenke[j].size();
                    max_person_idx = p;

                    // Abbruch, da man sich nur ein Geschenk wünschen kann
                    break;
                }
            }
        }

        // Person kann keinen Zweitwunsch erfüllt bekommen --> Überprüfung des
Drittwunsches
        if (max_value == -1) {
            for (int j = 0; j < n;++j) {
                if (find(dritte_geschenke[j].begin(), dritte_geschenke[j].end(), p) !=
dritte_geschenke[j].end()) {
                    if (dritte_geschenke[j].size() > 0) {
                        max_value = dritte_geschenke[j].size();
                        max_person_idx = p;

                        // Abbruch, da man sich nur ein Geschenk wünschen kann
                        break;
                    }
                }
            }
        }

        // Person kann auch keinen Drittwunsch erfüllt bekommen
        // --> Person bekommt Erstwunsch
        if (max_value == -1) {
            max_value = INF;
```

```
                max_person_idx = p;
            }

            // Aktualisierung des optimalsten Empfängers
            if (max_value > max_value_ges) {
                max_value_ges = max_value;
                max_person_idx_ges = max_person_idx;
            }
        }

        // falls alle Personen gleich geeignet sind, wird die erste Person ausgewählt
        if (max_value_ges == -1) {
            max_person_idx_ges = erste_geschenke[i][0];
        }

        // optimalste Person erhält den Erstwunsch
        verteilung[max_person_idx_ges] = i;

        // restliche potentielle Empfänger werden entfernt
        erste_geschenke[i].clear();

        // Person ist kein potenzieller Empfänger für andere Geschenke mehr
        zweite_geschenke = delete_element_vvi(zweite_geschenke, max_person_idx_ges);
        dritte_geschenke = delete_element_vvi(dritte_geschenke, max_person_idx_ges);
    }
}


// Alle Geschenke mit mindestens einem Bewerber als Zweitwunsch werden sicher als
Zweitwunsch vergeben werden
```

```
    // Geschenk kann somit nicht als Drittwunsch verwendet werden
    for (int i = 0; i < n; ++i) {
        if (zweite_geschenke[i].size() >= 1) {
            dritte_geschenke[i].clear();
        }
    }


    // Zuweisung aller Geschenke, die nur einen Bewerber haben
    // hier: Zweitwunsch
    for (int i = 0; i < n; ++i) {
        // Falls das Geschenk nur einen Bewerber hat, wird es sofort zugewiesen
        if (zweite_geschenke[i].size() == 1) {
            int person = zweite_geschenke[i][0];
            // person bekommt das Geschenk sicher
            verteilung[person] = i;

            // Liste für das aktuelle Geschenk wird gelöscht, da es vergeben wurde
            zweite_geschenke[i].clear();

            // person ist kein potenzieller Bewerber für andere Geschenke mehr
            dritte_geschenke = delete_element_vvi(dritte_geschenke, person);

        }
    }

    // Ermitteln der optimalsten Empfänger für alle Geschenke
    // hier: Zweitwunsch
    for (int i = 0; i < n; ++i) {
        // Geschenk hat mehrere Bewerber
        if (zweite_geschenke[i].size() >= 1) {
```

```
        // Jene Person mit den meisten Konkurrenten beim Drittwunsch erhält ihren
Zweitwunsch
        int max_value_ges = -1;
        int max_person_idx_ges;

        // Iteration über alle unterschiedliche Bewerber p
        for (int p : zweite_geschenke[i]) {

            // mit max_value werden die Anzahl der Konkurrenten beim Drittwunsch gespeichert
            int max_value = -1;
            int max_person_idx;

            // Drittwünsche werden überprüft
            for (int j = 0; j < n; ++j) {
                if (find(dritte_geschenke[j].begin(), dritte_geschenke[j].end(), p) !=
dritte_geschenke[j].end()) {
                    if (dritte_geschenke[j].size() > max_value) {
                        max_value = dritte_geschenke[j].size();
                        max_person_idx = p;

                        // Abbruch, da man sich nur ein Geschenk wünschen kann
                        break;
                    }
                }
            }

            // Person kann keinen Drittwunsch erfüllt bekommen
            // --> Person bekommt Zweitwunsch
            if (max_value == -1) {
                max_value = INF;
                max_person_idx = p;
```

```
            }

            // Aktualisieren des optimalsten Empfängers
            if (max_value > max_value_ges) {
                max_value_ges = max_value;
                max_person_idx_ges = max_person_idx;
            }
        }

        // falls alle Personen gleich geeignet sind, wird die erste Person ausgewählt
        if (max_value_ges == -1) {
            max_person_idx_ges = zweite_geschenke[i][0];
        }

        verteilung[max_person_idx_ges] = i;

        // restliche potenzielle Empfänger werden entfernt
        zweite_geschenke[i].clear();

        // Person ist kein potenzieller Empfänger für andere Geschenke mehr
        dritte_geschenke = delete_element_vvi(dritte_geschenke, max_person_idx_ges);
    }
}

// Zuweisung aller Geschenke beim Drittwunsch, egal wer das Geschenk bekommt
for (int i = 0; i < n; ++i) {
    if (dritte_geschenke[i].empty()) continue; // Geschenk hat keinen gewünschten Empfänger
    int person = dritte_geschenke[i][0];
    verteilung[person] = i;
    dritte_geschenke[i].clear();
}
```

```
// Nicht verwendete Geschenke werden ermittelt
vb used(n, false);
for (int i = 0; i < n; ++i) {
    if (verteilung[i] < 0) continue; // continue, falls verteilung[i] = -1 ,d.h. noch nicht vergeben
    used[verteilung[i]] = true;
}

// nicht verwendete Geschenke werden im Stack gespeichert
stack<int> s;
for (int i = 0; i < n; ++i) {
    if (!used[i]) s.push(i);
}

// restliche Personen bekommen die restlichen Geschenke vom stack
for (int i = 0; i < n; ++i) {
    if (verteilung[i] < 0) {
        verteilung[i] = s.top();
        s.pop();
    }
}

// Ausgabe
cout << "\n\n> Ausgabe der Verteilung: \n";
cout << "Schüler Nr. -> Geschenk Nr. \n";

for (int i = 0; i < n; ++i) {
    cout << i+1 << " -> " << verteilung[i]+1 << "\n";
}
```

```
    // Auswerten der Verteilung
    // Ermitteln der Anzahl der erfüllten Erst-, Zweit- und Drittwünsche
    int s1=0;
    int s2=0;
    int s3=0;
    int rest=0;
    for (int i = 0; i < n; ++i) {
        if (verteilung[i] == arr[i][0]) s1++;
        else if (verteilung[i] == arr[i][1]) s2++;
        else if (verteilung[i] == arr[i][2]) s3++;
        else rest++;
    }
    cout << "Anzahl an erfüllten 1. Wünschen: " << s1 << "\n";
    cout << "Anzahl an erfüllten 2. Wünschen: " << s2 << "\n";
    cout << "Anzahl an erfüllten 3. Wünschen: " << s3 << "\n";
    cout << "Anzahl an Geschenk, die nicht gewünscht waren: "<<rest;

    return 0;
}
```