Aufgabe 5: Wichteln

00759 - TranshumanTechnocrator - Erik Klein

1 Aufgabe

Gegeben sind Schüler mit jeweils drei Wünschen für Geschenke: Einen Wunsch 1., 2, und einen 3. Priorität. Es ist eine Verteilung von Geschenken an Schüler gesucht, sodass die Anzahl der erfüllten Wünsche 1. Priorität maximal ist, und wenn es mehrere solche Verteilungen gibt, auch die Anzahl Wünsche 2. Prio maximal usw.

2 Lösungsidee

Das Problem ist als ein bipartiter Graph darstellbar: Die Schüler sind Kantenmenge A, die Wünsche B. Eine Kante verläuft zwischen $u \in A$, $v \in B$, wenn v ein Wunsch von u ist. Meine Lösungsidee ist, Kantenkosten gemäß Prioritäten zu vergeben, und es als Gewichtetes Bipartites Matching mit dem Min Cost Max Flow Problem und einem Edmonds-Karp-Algorithmus zu lösen.

Die Wünsche 3. Prio sind am unwichtigsten, deshalb erhalten sie die Kosten $-1 = -(|A|+1)^0$. Ein Wunsch 2. Prio soll wichtiger sein als $|A|$ Wünsche 3. Prio, deshalb gebe ich ihm die Kosten $-(|A|+1)^1$. Und ein Wunsch 1. Prio ist wichtiger als $|A|$ Wünsche 2. Prio, erhält somit die Kosten $-(|A|+1)^2$. So werden alle Wichtigkeitsrelationen berücksichtigt. Das Matching mit den minimalen Kosten in diesem Graph gibt die optimale Verteilung der Geschenke.

Es wird noch ein Source-Knoten mit Kanten zu allen $u \in A$, und ein Sink-Knoten mit Kanten von allen $v \in B$ zu ihm mit jeweils Kosten 0 eingefügt. Alle genannten Kanten sollen außerdem eine Kapazität 1 haben. In diesem Netzwerk will ich den maximalen Fluss mit den minimalen Kosten finden. Berechnet wird dieser mit einem Edmonds-Karp-Algorithmus von `cp-algorithms.com/graph/Assignment-problem-min-flow.html`.

Die Geschenke, die dann noch nicht vergeben sind, werden dann einfach den restlichen Schülern zugeordnet. In Praxis ist es ein korrektes und effizientes Verfahren und löst selbst Beispiel 7 in Sekundenschnelle!

3 Umsetzung

Die Lösungsidee wurde in C++ implementiert. Die Kosten werden wie beschrieben in eine Matrix a geladen.

```
int N; cin >> N;
vector<vector<int>> a (N, vector<int>(N));
for(int i = 0; i < N; i++) {
    int w1, w2, w3; cin >> w1 >> w2 >> w3;
    a[i][w1-1] = -(N+1)*(N+1); // kosten 1. prio
    a[i][w2-1] = -(N+1);       // 2. prio
    a[i][w3-1] = -1;           // 3. prio
}
```

Den Rest erledigt die Funktion von der genannten prominenten Website:

```
vector<int> assignment(vector<vector<int>> a) {
    int n = a.size();
    int m = n * 2 + 2;
    vector<vector<int>> f(m, vector<int>(m));
    int s = m - 2, t = m - 1;
    int cost = 0;
    while (true) {
        vector<int> dist(m, INF);
```

```
        vector<int> p(m);
        vector<bool> inq(m, false);
        queue<int> q;
        dist[s] = 0;
        p[s] = -1;
        q.push(s);
        while (!q.empty()) {
            int v = q.front();
            q.pop();
            inq[v] = false;
            if (v == s) {
                for (int i = 0; i < n; ++i) {
                    if (f[s][i] == 0) {
                        dist[i] = 0;
                        p[i] = s;
                        inq[i] = true;
                        q.push(i);
                    }
                }
            } else {
                if (v < n) {
                    for (int j = n; j < n + n; ++j) {
                        if (f[v][j] < 1 && dist[j] > dist[v] + a[v][j - n]) {
                            dist[j] = dist[v] + a[v][j - n];
                            p[j] = v;
                            if (!inq[j]) {
                                q.push(j);
                                inq[j] = true;
                            }
                        }
                    }
                } else {
                    for (int j = 0; j < n; ++j) {
                        if (f[v][j] < 0 && dist[j] > dist[v] - a[j][v - n]) {
                            dist[j] = dist[v] - a[j][v - n];
                            p[j] = v;
                            if (!inq[j]) {
                                q.push(j);
                                inq[j] = true;
                            }
                        }
                    }
                }
            }
        }

        int curcost = INF;
        for (int i = n; i < n + n; ++i) {
            if (f[i][t] == 0 && dist[i] < curcost) {
                curcost = dist[i];
                p[t] = i;
            }
        }
        if (curcost == INF)
            break;
        cost += curcost;
        for (int cur = t; cur != -1; cur = p[cur]) {
            int prev = p[cur];
            if (prev != -1)
                f[cur][prev] = -(f[prev][cur] = 1);
        }
    }
```

```
72      vector<int> answer(n);
        for (int i = 0; i < n; ++i) {
74          for (int j = 0; j < n; ++j) {
                if (f[i][j + n] == 1)
76                  answer[i] = j;
            }
78      }
        return answer;
80  }
```

Es folgt nur noch die Ausgabe der Lösung. Es war sehr einfach zu implementieren, denn wie schon Steven Skiena meint, es ist besser, schon existierende, getestete und optimierte Industrie-Implementationen zu verwenden, als komplizierte Algorithmen selbst zu versuchen, zu implementieren. Man erledigt dann nicht nur Arbeit doppelt, sondern macht Fehler und kommt nur mit sehr viel Arbeit auf das Niveau professioneller Entwickler. Deshalb sehe ich mein Vorgehen als am besten vertretbar. Skiena hat sogar seine eigene Sammlung von Links zu Industrie-Implementationen, die Hitchhiker's guide to algorithms, `algorist.com/algorist.html`.

4 Beispiele

Mein Ausgabeformat gibt N Zeilen mit jeweils zwei Zahlen, die erste davon ein Schüler, die zweite das Geschenk, das er erhält. Hier die Ausgaben für die Beispiele von der BwInf-Webseite:
wichteln1.txt
1 2
2 5
3 1
4 3
5 8
6 4
7 7
8 10
9 9
10 6
wichteln2.txt
1 4
2 5
3 6
4 1
5 2
6 3
7 7
8 8
9 9
10 10
wichteln3.txt
1 2
2 20
3 29
4 8
5 1
6 3
7 5
8 12
9 4
10 28
11 6
12 9
13 14

14 23
15 26
16 30
17 11
18 7
19 16
20 10
21 19
22 13
23 27
24 15
25 17
26 18
27 22
28 21
29 24
30 25
wichteln4.txt
1 28
2 21
3 14
4 4
5 6
6 12
7 3
8 16
9 9
10 19
11 10
12 11
13 7
14 20
15 2
16 23
17 1
18 27
19 17
20 13
21 22
22 25
23 15
24 26
25 30
26 24
27 18
28 8
29 5
30 29
wichteln5.txt
1 4
2 6
3 7
4 18
5 27
6 2
7 5
8 9
9 25

10 16
11 14
12 13
13 19
14 15
15 10
16 8
17 26
18 23
19 20
20 3
21 1
22 21
23 22
24 11
25 24
26 28
27 30
28 12
29 17
30 29

Die Ausgaben für wichteln6.txt und wichteln7.txt befinden sich im Dateiordner.