Übung 3 – Graphen:

Fabian Mahdi, Dániel Láz

Verwendeter Algorithmus: Dijkstra

Funktionsbeschreibung:

1.: Schritt:

Die Kosten der Startstation werden auf 0 gesetzt. Um zu zeigen, dass sie die erste Station in unserem Pfad ist, bekommt sie sich selbst als Vorgänger.

```
curStation->setWeight(0);
curStation->setPredecessor(curStation);
```

Solange nicht alle Stationen besucht wurden, wiederhole Folgendes:

```
while(!allStationsVisited())
{
```

2.: Schritt:

Für jede Verbindung (Kante) der aktuellen Station (in der ersten Iteration entspricht „curStation“ der Startstation) wiederhole Folgendes:

1. Kontrolliere, ob die Station bereist besucht wurde.
2. Falls sie noch nicht besucht wurde: berechne die Kosten zum Erreichen der Station
3. Falls diese Kosten geringer sind als die aktuellen Kosten zum Erreichen der Station (Initialisierungswert der Kosten entspricht „logisch unendlich“): Ersetzte Kosten und Vorgänger durch die neuen Kosten und die aktuell besuchte Station

```
//get Connections of current station
std::vector<Station> connections = curStation->getConnections();

//iterate through all connections and calculate the needed cost
for(int i = 0; i < int(connections.size()); ++i)
{
    Station* curConnection = getStationFromName(connections[i].getName());

    //only calculate for unvisited stations
    if(!curConnection->wasVisited())
    {
        int totalCost = curStation->getWeight() + connections[i].getWeight();

        if(curConnection->getWeight() > totalCost)
        {
            curConnection->setWeight(totalCost);
            curConnection->setPredecessor(curStation);
            std::string lineUsed = connections[i].getLine()[0];
            curConnection->setLineUsed(lineUsed);
        }
    }
}
```

3.: Schritt:

Markiere die aktuelle Station als besucht.

```
curStation->isVisited();
```

4.: Schritt:

Iteriere durch alle unbesuchten Stationen und speichere die mit den niedrigsten Kosten in „curStation“ ab.

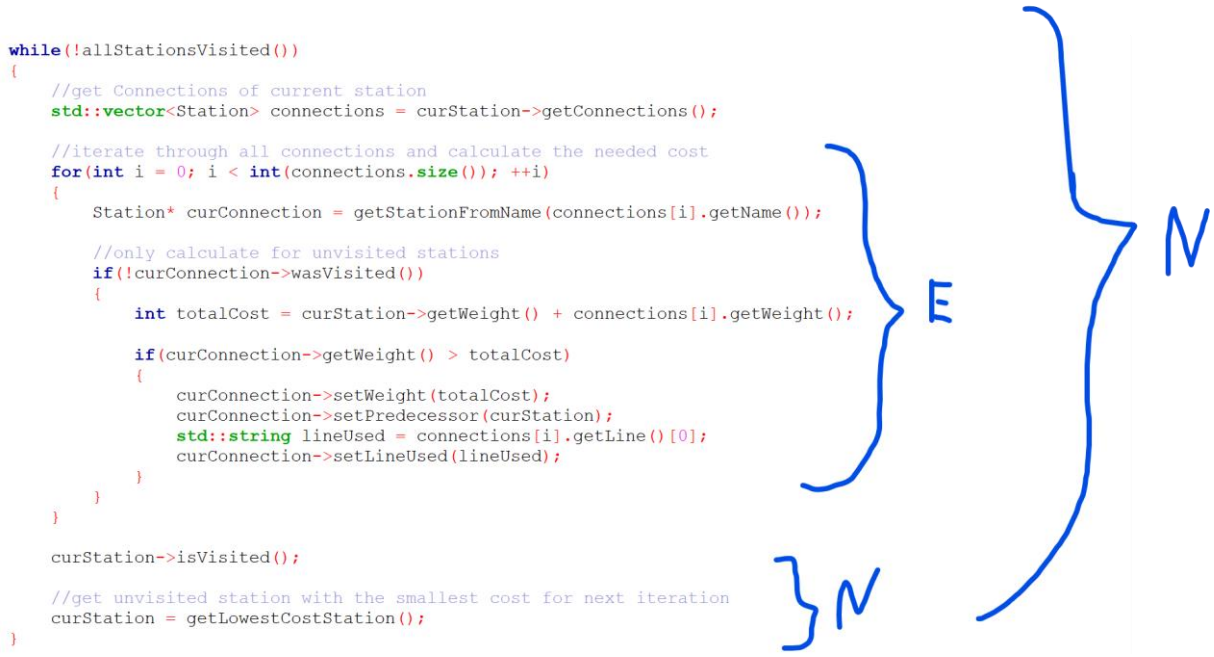
```
//get unvisited station with the smallest cost for next iteration
curStation = getLowestCostStation();
```

Schritt 2 – 4 werden wiederholt, bis alle Stationen besucht wurden.

Aufwandsabschätzung:

N = Anzahl aller Stationen

E = Anzahl aller Verbindungen einer Station



Die komplette Schleife wird N mal durchgeführt O(N).

Pro Iteration werden alle Verbindungen durchgegangen (O(E)) und die unbesuchte Station mit den niedrigsten Kosten wird erarbeitet (Diese Funktion hat bei uns einen Aufwand von O(N));

Komplexität von: O(N* (N + E))

Da E < V sein muss beträgt die Komplexität: O(N*N) = **O(N^2)**

Zeitmessung Dijkstra Algorithmus:

Komplette Angabe (U-Bahn + Bus): ca. 5.6 – 6.1 ms

Nur U-Bahnen: 0.15 – 0.21 ms