InformatiCup 2022 – Abfahrt!

Ausarbeitung von

Team_NULL

Johannes Böhmer (johannesboehmer0@gmail.com)

Ferdinand Pfeifer (ferdinand.pfeifer@student.uni-siegen.de)

Silas Trippler (silas.trippler@web.de)

Jens Vollmer (jens.vollmer@student.uni-siegen.de)

Inhaltsverzeichnis

1. Einleitung

Die Aufgabe beim InformatiCup 2022 ist es, ein Programm zu schreiben, das den bestmöglichen Fahrplan für ein beliebiges Schienennetzwerk berechnet. Ziel ist es, die zusammengerechnete Verspätung aller Passagiere so klein wie möglich zu halten. Als Ausgangssituation erhält man ein Schienennetzwerk mit einer Liste von Stationen, Bahnstrecken, Zügen und Passagieren.

Bei Betrachtung der Aufgabenstellung fallen mehrere Gesichtspunkte auf, die für die Entwicklung eines gut funktionierenden Programms und die Erstellung eines möglichst effizienten Fahrplans berücksichtigt werden müssen.

Die beiden zentralen Aspekte, die betrachtet werden sollten, sind dabei einerseits die summierte Verspätung aller Passagiere und die für Berechnung des Fahrplans benötigte Laufzeit. Als Verspätung zählt in diesem Fall der Unterschied zwischen der gewünschten und der tatsächlichen Ankunftszeit eines Passagiers multipliziert mit der Größe der Passagier Gruppe.

In der folgenden Ausarbeitung erläutern wir unsere erarbeiteten Lösungsansätze sowie die grobe Architektur unseres Programms.

2. Lösungsansätze

2.1 Gewichteter Graph

Das Netzwerk aus Bahnhöfen und Strecken wird durch einen gewichteten Graphen modelliert. Dabei werden die Bahnhöfe durch die Knoten und die Strecken durch die Kanten repräsentiert. Die Gewichte der Kanten entsprechen der jeweiligen Streckenlänge.

Durch dieses Grundmodell ist es sehr effizient möglich den kürzesten Weg zwischen zwei Knoten über den Algorithmus von Dijkstra zu ermitteln. Außerdem können Informationen wie benachbarte Stationen, angrenzende Strecken und deren Länge/Gewicht bestimmt werden.

Dieses Modell bildet die Basis unserer folgenden Lösungsansätze.

2.2 Zug-Passagier Bewertung

In einer Matrix wird für jedes Zug-Passagier Paar eine Bewertung angelegt. Diese stellt dar, wie gut der Zug dafür geeignet ist, den Passagier in der gewünschten Zeit an den jeweiligen Zielbahnhof zu bringen. Das Rating wird zu Beginn des Programms einmal aufgestellt. Anhand dieser Bewertung wird anschließend für jeden Zug festgelegt, welchen Passagier er als nächstes transportieren soll.

Das Problem, was hierbei jedoch auftritt ist, dass die jeweilige Zug-Passagier Bewertung nicht nur zu Anfang berechnet werden kann, da sich die Positionen der Züge und somit deren Distanzen zu den einzelnen Passagieren stetig verändert.

Das Aufstellen dieser Matrix nach jedem Simulationsschritt ist jedoch zu ineffizient.

Um dieses Problem zu beheben entwickelten wir den nächsten Ansatz.

2.3 Pfad Gruppierung

Für jeden Passagier wird der kürzeste Weg zu seinem Zielort berechnet. Die daraus resultierenden Pfade werden nach der Länge des Weges absteigend und der Ankunftszeit des Passagiers aufsteigend sortiert. Diese Pfade werden zu Beginn des Programms einmalig berechnet. Anschließend werden die Passagiere mit folgenden Kriterien gruppiert:

1. Identische Pfade
2. Teilabschnitte des Pfades

Dazu wird immer der längste noch nicht betrachtete oder bereits gruppierte Pfad als Basis verwendet. Durch die vorher angegebene Sortierung kann dafür immer der erste Pfad verwendet werden. Außerdem wird durch die Sortierung sichergestellt, dass die Passagiere, welche die früheste Ankunftszeit haben, als Erstes betrachtet werden.

Nun muss der bestmögliche Zug bestimmt werden. Diese werden zur Betrachtung in die folgende drei Klassen unterteilt und entsprechend priorisiert:

1. Züge an der Station des Passagiers
2. Unplatzierte Züge
3. Alle restlichen Züge im Schienennetzwerk

Die zuvor angegeben Priorisierung betrachtet die Anfahrtszeit der Züge zu dem Passagier, als zweiten Faktor wird die Größe des Zuges betrachtet, damit dieser einerseits möglichst wenig Kapazität verschwendet und andererseits in der Lage ist die Gruppe zu transportieren. Die Anfahrtszeit setzt sich zusammen aus der Distanz zu dem Passagier geteilt durch die Geschwindigkeit des Zuges.

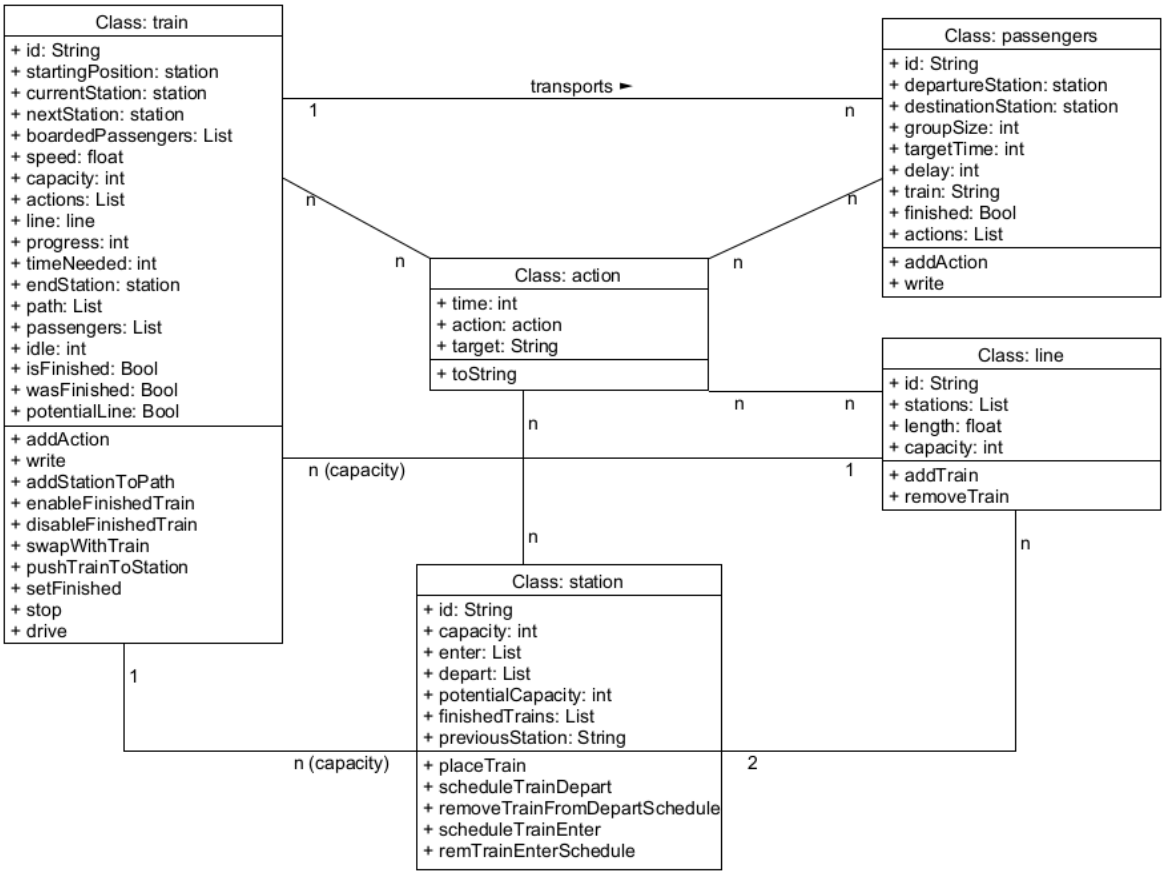
Alle Passagiere, die der gewählte Zug transportieren kann, werden im folgenden nicht mehr betrachtet.

Im nächsten Schritt, wird die ungefähr für den Transport der Gruppe benötigte Zeit abgeschätzt. Diese ergibt sich aus der Pfadlänge geteilt durch die Geschwindigkeit des Zuges. Hinzu kommt eine kurze Verzögerung für jede Station, an der Passagiere ein- oder aussteigen. Diese Zeitschätzung wird dem Zug zugewiesen, damit bei den weiteren Gruppen berücksichtigt wird, dass dieser eine gewisse Zeit unterwegs ist.

Dieser Vorgang wird so lange wiederholt, bis alle Passagiere behandelt wurden.

3. Softwarearchitektur

3.1 UML-Diagramm



3.2 Coding Conventions

Der Programmcode entspricht den Python Coding Conventions nach folgendem Standard:

https://www.python.org/dev/peps/pep-0008/.

3.3 Verwendete Softwarepakete

NumPy

NumPy ist eine Programmbibliothek für die Programmiersprache Python, die eine einfache Handhabung von Vektoren, Matrizen oder generell großen mehrdimensionalen Arrays ermöglicht. Neben den Datenstrukturen bietet NumPy auch effizient implementierte Funktionen für numerische Berechnungen an. [1]

NetworkX

NetworkX ist eine freie Python-Bibliothek auf dem Gebiet der Graphentheorie und Netzwerke. Aufgrund der Verwendung einer reinen Python-Datenstruktur ist NetworkX ein recht effizientes, sehr skalierbares, hochportables Framework für die Analyse von sozialen und anderen Netzwerken.[2]

[1] Wikipedia (2022): https://de.wikipedia.org/wiki/NumPy (15.01.2022).
[2] Wikipedia (2022): https://de.wikipedia.org/wiki/NetworkX (15.01.2022).

3.4 Architektur

Das Programm besteht aus folgenden Klassen:

- Action (action.py)
- Line (line.py)
- Passengers (passengers.py)
- Station (station.py)
- Train (train.py)

Diese befinden sich in einem dafür angelegten Ordner “classes”.

Die Beziehung der oben genannten Klassen wird durch das in 3.1 angegebene UML-Diagramm beschrieben. Die Instanzierung, Verwaltung und Organisation der Objekte der zuvor genannten Klassen übernimmt die Datei “main.py”. Diese beinhaltet auch die Hauptfunktionen des Lösungsansatzes “Pfad Gruppierung”.

3.5 Softwaretesting

Zum Testen der Software haben wir einerseits das für den Wettbewerb zur Verfügung gestellte Testsystem verwendet. Hierbei wird das Programm automatisch auf einem Server mit der Anforderungsbeschreibung entsprechenden Eingaben getestet und überprüft, ob das Programm einen fehlerfreien Fahrplan berechnet.

Des Weiteren haben wir unser eigenes Testsystem entwickelt, das zufällige Inputs für das Programm generiert. Dieses ermöglichte es uns, die Laufzeit und die Qualität der Lösung zu verbessern.

4. Benutzerhandbuch

Um das Programm auf Ihrem Rechner zu installieren, klonen Sie bitte zuerst das gegebene GitHub-Repository auf Ihren Rechner.

Öffnen Sie nun im Stammverzeichnis des Repositories das Terminal und führen Sie folgenden Befehl aus:

```
docker build -t [Name]
```

Nun können Sie das Programm mit folgendem Befehl starten:

```
docker run -i [Name]
```

Die Eingabe der Fahrplandaten geschieht über den Standardinput. Beispiel für Linux:

```
cat input.txt > docker run -i [Name]
```

5. Auswertung

Um die Qualität und die Leistungsfähigkeit unseres Programms zu quantifizieren, haben wir zuerst Faktoren festlegen, anhand derer wir diese bewerten können. Die Qualität bemisst sich in unserem Fall an der zusammengerechneten Verspätung aller Passagiere, die Leistungsfähigkeit an der Laufzeit des Programms.

In der folgenden Tabelle werden die Ergebnisse bezüglich der angegebenen Testdateien hinsichtlich Gesamtverspätung und der dafür benötigten Laufzeit dargestellt:

Dateiname	**Gesamtverspätung**	**Laufzeit in Sekunden**
Input CI	0	0.010
Input 1	39	0.022
Input 2	0	0.004
Input 3	0	121.000
Input 4	40	0.003
Input 5	0	0.005
Input 6	0	0.004

Die Input Files stammen aus dem Bahnsimulator und der CI. Die detaillierten Input-Files finden Sie im gegebenen GitHub-Repository.

Aufgrund von mangelnden Daten bezüglich der durchschnittlichen Gesamtverspätung und Laufzeit auf diesen oder anderen Testdateien, ist es nicht möglich die Qualität unserer Lösung einzuordnen.

6. Diskussion

6.1 Argumente

Im Folgenden stellen wir die Vorteile und Nachteile unseres final gewählten Lösungsansatzes dar. Anschließend ziehen wir auf Basis dieser in Abschnitt 7 ein Gesamtfazit.

Alle Wege der Passagiere und deren Gruppierungen werden zu Beginn des Programms berechnet. Dadurch müssen im Optimalfall keine Änderungen an den Pfaden vorgenommen werden, während die Züge fahren. Dies sorgt für eine sehr niedrige und konstante Laufzeit.

Die Gruppierung der Passagiere sorgt für eine möglichst geringe Streckenauslastung, da die Passagiere durch diese Gruppierung auf so wenig Züge wie möglich pro Strecke verteilt sind.

Ein weiterer Vorteil der Wegberechnung zu Beginn des Programms ist, dass diese eine, dem Passagieraufkommen angepasste, optimale Verteilung der unplatzierten Züge ermöglicht.

Trotz der genannten Vorteile, welche die Berechnung zu Beginn mit sich bringt, entsteht dadurch eine potenzielle Kollisionsgefahr der Zugrouten bei geringen Stations- und/oder Streckenkapazitäten.

Folgende Fälle sind dabei zu beachten:

- Die maximale Kapazität der Station ist erreicht und kein Zug plant die Station zu verlassen. Dies kann durch das Verschieben eines blockierenden Zuges, welcher alle ihm zugewiesenen Passagiere abgeliefert hat, in eine angrenzende Station gelöst werden.
- Bei Stationen deren Kapazität eins ist und deren Verbindung untereinander eine maximale Kapazität von eins hat. Dieser Fall kann durch eine Ausnahmebehandlung behoben werden, falls es einen alternativen Weg zu der Zielstation gibt wird diese dem Zug zugewiesen.
- Wenn jedoch kein alternativer Weg vorhanden ist, ist die Situation mit dem gewählten Ansatz nicht mehr zu lösen.

6.2 Fazit

Aufgrund der oben genannten Vor- und Nachteile des Lösungsansatzes ist dieser insgesamt als sehr erfolgreich zu bewerten. In geringer Laufzeit werden Lösungen berechnet und diese sind, bis auf den letzten Fall der Kollisionen, valide und erzielen eine geringe Gesamtverspätung.