Abfahrt!

Ab dem Jahr 2021 soll im Bahnverkehr schrittweise der Deutschlandtakt[1] eingeführt werden. Ziel dieses Konzepts ist ein abgestimmter Taktfahrplan für den Schienenpersonenverkehr in Deutschland. Pünktlich mit dem Zug von Stadt zu Stadt, auch auf dem Land bequem umsteigen ohne lange Wartezeiten - diese Vision soll in Deutschland Realität werden.

Auch der informatiCup widmet sich in diesem Jahr der Aufgabe, einen Fahrplan für ein Schienennetzwerk zu erstellen. Das Ziel ist eine Software für die Berechnung von optimalen Fahrplänen, die die Gesamtverspätung aller Fahrgäste minimiert und somit insgesamt die Zufriedenheit der Kunden mit dem Schienenverkehr verbessert.

Diese so einfach formulierte Aufgabe enthält eine Menge weiterer Herausforderungen, die Ihr zunächst genau analysieren und für die Ihr schließlich eine Lösung entwickeln sollt. Tausende Passagiere möchten täglich von A nach B transportiert werden - und das möglichst schnell. Um diese Passagiere zu transportieren steht jedoch nur eine begrenzte Anzahl an Zügen zur Verfügung. Ein riesiges Schienennetzwerk verbindet alle Städte miteinander. Ein reibungsloser Ablauf aller Schienenfahrzeuge muss gewährleistet werden, um die Ressourcen optimal zu nutzen. Vor diesem logistischen Optimierungsproblem stehen nicht nur Eisenbahnkonzerne, sondern auch viele andere Logistikunternehmen, die Personen oder Waren transportieren. Der algorithmische Kern der diesjährigen informatiCup-Aufgabe ist damit im wirtschaftlichen Kontext tagtäglich relevant und optimale Lösungen sind praktisch einsetzbar.

[1] https://www.deutschlandtakt.de

Das Modell

Das Bahnnetz wird in dieser Aufgabe als rundenbasiertes Modell dargestellt. Jede Runde entspricht einer (fiktiven) Zeiteinheit. In dem Modell gibt es vier Elemente, die im weiteren detailliert beschrieben werden: Bahnhöfe, Strecken, Züge und Passagiere.

Eure Aufgabe ist es, einen Zugfahrplan und einen Passagierfahrplan zu erstellen, so dass alle Passagiere mit möglichst wenig Verspätung an ihrem Ziel ankommen. Das Format des Fahrplans findet Ihr in dem Abschnitt "Eingabe & Ausgabe".

Bahnhof

An einem Bahnhof können Passagiere ein- und aussteigen sowie eine beliebige Zeit verweilen. Ein Bahnhof hat eine maximale Zugkapazität, allerdings keine Kapazität an Passagieren. Folgende Eigenschaften besitzt ein Bahnhof.

- **Name:** Eine eindeutige Bezeichnung mit der jeder Bahnhof identifiziert werden kann.
- **Kapazität:** Die maximale Anzahl an Zügen die gleichzeitig in einer Runde an einem Bahnhof stehen können.

Strecke

Eine Strecke ist die Verbindung zwischen zwei Bahnhöfen. Sie kann eine bestimmte Menge an Zügen aufnehmen, wobei die genaue Position der Züge auf der Strecke für die Simulation irrelevant ist. Dadurch ist es in unserem einfachen Modell zum Beispiel erlaubt, dass alle Züge auf der Strecke gleichzeitig abfahren und parallel fahren, auch wenn das nicht realistisch ist. Es wird garantiert, dass jede Stadt direkt oder transitiv über Strecken verbunden ist.

- **ID:** Eine eindeutige Bezeichnung mit der jede Strecke identifiziert werden kann.
- **Ende:** Der Name der Bahnhöfe, die die Strecke verbindet. Es gibt immer exakt zwei Enden. Die Reihenfolge der Enden ist nicht relevant, eine Strecke kann immer von beiden Seiten angefahren werden.
- **Kapazität:** Die maximale Anzahl der Züge, die in einer Runde gleichzeitig auf einer Strecke sein können.
- **Länge:** Die Länge der Strecke als Dezimalzahl.

Zug

Züge bewegen sich auf Strecken zwischen Bahnhöfen und können dadurch Passagiere transportieren. Dabei ist zu beachten, dass bei einem Zug in der Abfahrts- und Ankunftsrunde keine Passagiere einsteigen oder aussteigen können.

Wenn ein Zug auf einer Strecke ist, so kommt er in der Runde an, in der *Geschwindigkeit x Zeiteinheiten auf Strecke* größer ist als die Länge der Strecke. Dabei kann es vorkommen, dass ein Zug in derselben Runde an einem Bahnhof ankommt, in der er von einem anderen losgefahren ist.

- **ID:** Eine eindeutige Bezeichnung mit der jeder Zug identifiziert werden kann.
- **Startposition:** Name des Bahnhofs, an der ein Zug am Anfang der Simulation steht. Einige Züge können am Anfang frei platziert werden, diese haben die Startposition `'*'`.
- **Kapazität:** Anzahl der Passagiere die in einer Runde gleichzeitig in einem Zug sein können.
- **Geschwindigkeit:** Länge der Strecke die sich ein Zug auf einer Strecke in einer Runde bewegen kann als Dezimalzahl.

Der Zugfahrplan, den Ihr berechnet, enthält die Abfahrtszeiten aller Züge. Zudem könnt Ihr allen Zügen mit der Startposition `'*'` am Anfang eine Startposition zuordnen.

Passagiere

Passagiere bewegen sich in Zügen von einem Bahnhof zu einem anderen. Sie haben eine Zielzeit. Kommen Passagiere später als die Zielzeit an ihrem Bahnhof an, so sind die verspätet. Das Ziel der Simulation ist es, diese Verspätung zu minimieren.

Zur Vereinfachung wird nicht jeder einzelne Passagier simuliert, sondern Passagiere werden in Gruppen zusammengefasst (zum Beispiel eine Familie oder eine Schulklasse, die während der ganzen Fahrt zusammen bleibt). Gruppen werden als eine Einheit gesehen, die sich während der Simulation nicht trennt.

- **ID:** Eine eindeutige Bezeichnung mit der jede Passagiergruppe identifiziert werden kann.
- **Startbahnhof:** Name des Bahnhofs an dem eine Passagiergruppe am Anfang der Simulation steht.
- **Zielbahnhof:** Name des Bahnhofs an dem eine Passagiergruppe ankommen soll.
- **Gruppengröße:** Anzahl der Personen in der Gruppe, relevant für die Kapazität von Zügen.
- **Zielzeit:** Zeitpunkt in der Simulation zu dem eine Gruppe am Zielbahnhof sein möchte. Kommt eine Gruppe später als diese Zeiteinheit an so hat sie Verspätung.

In Eurem Passagierfahrplan sollt Ihr festhalten, in welchen Zug Passagiere wann einsteigen und wann wieder aussteigen sollen. Ein Passagier kann nicht in einen Zug einsteigen, der in dieser Runde abfährt oder angekommen ist!

Ablauf einer Runde

In jeder Runde wird die momentane Zeiteinheit um eins erhöht. Danach werden alle Aktionen (Züge fahren ab und kommen an, Passagiere steigen ein und aus) parallel in beliebiger Reihenfolge ausgeführt. Das bedeutet, dass *während* einer Runde die Kapazitäten kurzzeitig überschritten werden dürfen, solange *am Ende* jeder Runde sämtliche Kapazitäten eingehalten werden. Falls am Ende einer Runde irgendeine der Bedingungen oder Kapazitäten nicht erfüllt oder überschritten ist, gilt der Fahrplan insgesamt als ungültig.

Die Simulation wird solange ausgeführt bis die letzte Handlung in einem der Fahrpläne ausgeführt wurde.

In dem GitHub Repository des diesjährigen Wettbewerbs[2] bieten wir Euch ein Tool an, mit dem sich basierend auf einer Eingabe- und Ausgabedatei die Simulation ausführen lässt.

[2] https://github.com/informatiCup/informatiCup2022

Eingabe & Ausgabe

Die Eingabe erfolgt über die Standardeingabe, die Ausgabe über die Standardausgabe. Folgende Formate werden dafür genutzt.

Eingabe

Jede einzelne Aufgabe wird eurer Software als Eingabe über die die Standardeingabe übergegeben. Das Format der Eingabe ist wie folgt definiert.

```
# Kommentar
# Leere Zeilen werden ignoriert
# str: ([a-z]|[A-Z]|[0-9]|_)+
# dec: Dezimalzahl mit beliebig vielen Nachkommastellen
# int: Ganzzahl mit beliebig vielen Stellen >= 0
# Werte-Trenner: 0x20 (SPACE)
# Zeilen-Trenner: 0x0A (NEWLINE)

# Bahnhöfe: str(ID) int(Kapazität)
[Stations]
S1 2
S2 2
S3 2

# Strecken: str(ID) str(Anfang) str(Ende) dec(Länge) int(Kapazität)
[Lines]
L1 S2 S3 3.14 1
L2 S2 S1 4 1

# Züge: str(ID) str(Startbahnhof)/* dec(Geschwindigkeit) int(Kapazität)
[Trains]
T1 S2 5.5 30
T2 * 0.999 50

# Passagiere: str(ID) str(Startbahnhof) str(Zielbahnhof)
# int(Gruppengröße) int(Ankunftszeit)
[Passengers]
P1 S2 S3 3 3
P2 S2 S1 10 3
```

Ausgabe

Die berechneten Zugfahrpläne und Passagierfahrpläne müssen über die Standardausgabe ausgegeben werden. Dabei muss das Format exakt wie folgt sein.

```
# Kommentar
# Leere Zeilen werden ignoriert
# str: ([a-z]|[A-Z]|[0-9]|_)+
# dec: Dezimalzahl mit beliebig vielen Nachkommastellen
# int: Ganzzahl mit beliebig vielen Stellen >= 0
# Werte-Trenner: 0x20 (SPACE)
# Zeilen-Trenner: 0x0A (NEWLINE)
# Invariante: Zeitpunkt >= 1

# ID des Akteurs (Zug, Passagier)
[Train:T1]
# Start: 0 Start str(ID des Startbahnhofs)
# Start muss angegeben werden, GENAU DANN WENN der Zug in der
# Eingabe * als Startbahnhof hat UND der Zug verwendet wird.
# Start darf nicht angegeben werden, GENAU DANN WENN der Zug
# bereits einen Startbahnhof hat.
# Abfahren: int(Zeitpunkt) Depart str(ID der Strecke)
# Zeitpunkte müssen pro Zug sortiert sein
2 Depart L1

[Train:T2]
0 Start S2
2 Depart L2

[Passenger:P1]
# Zeitpunkte müssen pro Passagier sortiert sein
# Einsteigen: int(Zeitpunkt) Board str(ID des Zuges)
1 Board T1
# Aussteigen: int(Zeitpunkt) Detrain
3 Detrain

[Passenger:P2]
1 Board T2
7 Detrain
```

Ein Passagier kann nicht in einen Zug einsteigen, der in dieser Runde abfährt oder angekommen ist!

Bewertung

Die Bewertung der Lösung erfolgt anhand verschiedener Kriterien: theoretischer Ansatz, Softwarearchitektur und -qualität, Benutzerführung, Qualität der Lösung und Präsentation im Finale.

Bitte achtet bei Eurer Abgabe darauf, dass wirklich alle Aspekte (ausgenommen Präsentation) bearbeitet wurden. Eure Abgabe muss eine theoretische Ausarbeitung sowie ein Handbuch enthalten.


Nutzt die Checkliste um sicher zu gehen, dass Eure Abgabe wirklich vollständig ist! Unvollständige Abgaben werden nicht begutachtet.

Teilnahme und Quelltextverwaltung

Die Anmeldung zum Wettbewerb findet in diesem Jahr zum ersten Mal nicht per Email, sondern über das neue Online-Portal *Teammates*[3] für Teilnehmer am informatiCup statt. Auf diesem Online-Portal könnt Ihr Euch als Teilnehmer registrieren, zu einem gemeinsamen Team einladen und dieses Team schließlich zum Wettbewerb anmelden.

Außerdem kann in Teammates jedes Mitglied eines Teams einen öffentlichen SSH-Schlüssel eintragen und damit über SSH auf das Git Repository des Teams zugreifen. Dieses Git Repository wird automatisch jedem angemeldeten Team durch ein neues Continuous Integration (CI) Tool für den informatiCup zur Verfügung gestellt. Eine technische Anleitung dazu findet Ihr im GitHub Repository des diesjährigen Wettbewerbs[4].


Die Abgabe Eurer Lösung erfolgt schließlich durch einen Push in dieses Team-Repository. Theoretische Ausarbeitung und ggf. Handbuch müssen als PDF im Team-Repository vorliegen.

Auch vor der finalen Abgabe solltet Ihr Eure Lösung regelmäßig in das Team-Repository pushen, damit eure Software automatisch getestet wird. Die Ergebnisse dieser Softwaretests könnt Ihr fortlaufend im Teammates-Portal einsehen.

Als Lösungseinreichung wird ein Dockerfile, mit dem ein lauffähiges Docker Image auf einem Referenzsystem (Linux) erstellt werden kann, erwartet. Das Docker Image muss alleine und ohne weitere manuelle Schritte mit einem Aufruf von `docker build` erstellt werden können.


Abgaben, die das CI nicht bauen oder starten kann, werden zurückgewiesen. Bitte nutzt das CI deshalb unbedingt regelmäßig, um Eure Lösungen zu validieren.

Die FAQs zum laufenden Wettbewerb sowie Beispieleingaben und -ausgaben findet Ihr in dem GitHub Repository des diesjährigen Wettbewerbs.

---

[3] https://teams.informaticup.de/
[4] https://github.com/informatiCup/informatiCup2022

Checkliste der Bewertungskriterien

Theoretischer Ansatz

Der theoretische Ansatz muss in einer Ausarbeitung, die zusammen mit der Implementierung eingereicht wird, dargestellt werden. Bewertet werden sowohl der Inhalt als auch die Form.

Theoretische Ausarbeitung

Die theoretische Ausarbeitung soll die verwendete Theorie beschreiben.

- ❏ **Hintergrund:** Der Hintergrund der Lösung. Welche theoretischen Ansätze wurden verwendet? Warum wurden diese verwendet?
- ❏ **Auswertung:** Wie gut ist die Qualität der Lösung? Nach welchen (wissenschaftlichen) Kriterien wurde bewertet? Warum wurden diese Kriterien verwendet?
- ❏ **Diskussion:** Wie „gut“ (auch außerhalb der reinen Qualität) ist die Lösung? Was für Probleme gibt es noch? Wie lässt sich die Lösung praktisch einsetzen?
- ❏ **Quellen:** Wurden (wissenschaftliche) Quellen richtig und angemessen verwendet?

Formalien

Eine gute Form ist entscheidend für die Lesbarkeit einer Ausarbeitung. Beachte deshalb neben dem reinen Inhalt der Ausarbeitung auch einige Formalien.

- ❏ **Rechtschreibung:** Rechtschreibung und Grammatik sind korrekt.
- ❏ **Lesefluss:** Es gibt einen Lesefluss in der Ausarbeitung.
- ❏ **Layout:** Das Dokument hat ein einheitliches Layout. Dieses kann frei gewählt werden, darf aber nicht den Lesefluss stören.
- ❏ **Zitate:** Es wird richtig und einheitlich zitiert.
- ❏ **Quellenangaben:** Quellen sind richtig und einheitlich angegeben.

Softwarearchitektur und -qualität

Da eine etablierte Softwarearchitektur nur mit hohem Aufwand zu ändern ist, sollte sie besonders gründlich durchdacht und ausführlich begründet werden. Ausgewählte Aspekte der Softwarequalität sind für die Bewertung von besonderer Bedeutung. Gerne dürfen auch hier nicht genannte Aspekte aus den sehr weiten Feldern Softwarearchitektur und -qualität beleuchtet werden.

- **Architektur:** Beschreibung der Komponenten und deren Beziehungen
- **Software testing:** Begründetes Konzept, Umsetzung
- **Coding conventions:** Begründetes Konzept, Umsetzung
- **Wartbarkeit:** Mit welchem Aufwand kann das System angepasst werden?

Benutzerführung

Damit eine Software akzeptiert wird, muss diese eine passende Benutzerführung besitzen. Dazu gehört neben dem Interface selber auch ein Handbuch (in dem Benutzer durch die Funktionen der Software geführt werden) und eine Bauanleitung zum Erstellen der Anwendung.

- **Bauanleitung:** Erwartet wird eine Dockerfile, mit dem ein lauffähiges Docker Image auf einem Referenzsystem (Linux) erstellt werden kann. Bitte stellt sicher, dass das Docker Image allein mit Hilfe von `docker build` erstellt werden kann, manuelle Schritte außerhalb von `docker build` werden nicht getätigt.
- **Handbuch:** Eine kurze Anleitung, wie die Software und eventuelle Erweiterungen auszuführen und zu bedienen sind.
- **Interface:** Die Benutzung der Software ist einfach gestaltet. Dies kann durch ein Shell-Interface oder durch eine grafische Oberfläche geschehen. Eine grafische Oberfläche ist nicht verpflichtend.

Qualität der Lösung

Die Qualität der Lösung wird anhand der Gesamtverspätung aller Passagiere ermittelt.

- **Minimale Gesamtverspätung:** Wie hoch ist die Verspätung aller Passagiere nach dem berechneten Plan?
- **Rechenzeit:** Wie lange benötigt Eure Software, um die Lösung zu berechnen?
- **Bonus:** Zusätzlich kann eine einzige Aufgabe als Eingabedatei eingereicht werden, bei der Ihr glaubt, dass Eure Lösung besonders gut funktioniert. Dieser Punkt ist nicht verpflichtend.

Präsentation

Im informatiCup-Finale werden die besten Lösungen vor einer Fachjury präsentiert.

- ❏ **Foliendesign:** Sind die Folien ansprechend? Lenken die Folien nicht vom Inhalt der Präsentation ab?
- ❏ **Vortragsstil:** Weckt der Vortragsstil Interesse an der Präsentation?
- ❏ **Verständlichkeit:** Ist der Vortrag verständlich? Wird der Hintergrund der Lösung in einem angemessenen Tempo erklärt? Wird nötiges Vorwissen geschaffen?
- ❏ **Reaktion auf Nachfragen:** Können Nachfragen beantwortet werden?