Fahrplanauskunft

Kürzeste Verbindungen

- **Gegeben:**
  - Eine Zugfahrplan
  - Eine Menge von *Anfragen*: Eine *Anfrage* ist ein Tripel $(S, T, time)$, wobei S den Startbahnhof, T den Zielbahhof und $time$ die früheste Abfahrtszeit angibt.
- **Gesucht:** Für jede Anfrage $(S, T, time)$ eine Zugverbindung von S nach T mit minimaler Reisedauer, d.h. die Zeit zwischen der frühesten Abfahrtszeit $time$ und der Ankunft in T ist minimal unter allen Verbindungen.

Ein (vereinfachter) Zugfahrplan

Ein **Fahrplan** besteht aus

- einer Menge von **Bahnhöfen** und
- einer Menge von **Ereignissen**: die *Abfahrt* oder die *Ankunft* eines Zuges an einem Bahnhof.
  Ein **Ereignis** ist ein Vektor $e = (S, e', leaving, time)$, dabei seien
  - S: der Bahnhof an dem das Ereignis geschieht
  - e': Die mit e verbundene Ankunft am nächsten Bahnhof, falls e eine Abfahrt ist, sonst die mit e verbundene Abfahrt
  - $leaving$: wahr, falls das Ereignis eine Abfahrt ist
  - $time$: die Uhrzeit

Das Datenformat des Fahrplans

Der Fahrplan liegt in Form einer Textdatei vor. Diese besteht aus einer Folge von Zeilen, die eine fest vorgegeben Struktur aufweisen:

- Die erste Zeile enthält zwei ganze Zahlen, die durch ein Leerzeichen getrennt sind.
  - Die erste Zahl n_1 ist der grösste Index eines Bahnhofes, wobei alle Bahnhöfe von 0 bis n_1 durchnummeriert sind. Es gibt also $n_1 + 1$ Bahnhöfe.
  - Die zweite Zahl n_2 ist der grösste Index eines Ereignisses, wobei alle Ereignisse von 0 bis n_2 durchnummeriert sind. Es gibt also $n_2 + 1$ Ereignisse.
- Dann folgt für jeden Bahnhof eine Folge von Zeilen, ein sogenannter *Bahnhofsblock*.

- ein *Bahnhofsblock* besteht aus folgenden Zeilen:
  1. Dem Namen des Bahnhofes
  2. Die EVA Nummer des Bahnhofes
  3. Diese Zeile besteht aus vier Zahlen:
     - dem Index des Bahnhofes (ganzzahlig),
     - der X-Koordinate des Bahnhofes (rationale Zahl),
     - der Y-Koordinate des Bahnhofes (rationale Zahl) und
     - einem ganzzahligen Wert, welcher der Wichtigkeit des Bahnhofes entspricht
  4. Diese Zeile besteht aus drei Zahlen, nämlich aus
     - der Anzahl der Ereignisse insgesamt,
     - der Anzahl der Abfahrten und
     - der Anzahl der Ankünfte an diesem Bahnhof.

- *Bahnhofsblock* (Fortsetzung)
  - Dann folgt für jedes Ereignis $e = (S, e', leaving, time)$ dieses Bahnhofs eine Zeile: Diese bestehen jeweils aus vier Zahlen:
    - dem Index des Bahhofes T von e'
    - der Index von e' innerhalb des Bahnhofs T
    - im Fall eines Abfahrtsereignis 1, sonst 0
    - der Uhrzeit des Ereignisses in Minuten nach Mitternacht

    Die Ereignisse sind dabei innerhalb eines Bahnhofes von 0 bis Anzahl der Ereignisse durchnummeriert (darauf bezieht sich die zweite Zahl), und nach der Uhrzeit sortiert.

Das Datenformat - Beispiel

```
6960 931746
Hannover Hbf
8000152
0 0 0 5000
1145 574 571
1662 177 0 0
263 0 1 6
1658 0 1 8
1660 1 0 9
1660 2 1 9

...

Konstanz
8003400
2089 -43.5 -519.8 200
104 52 52
459 1 0 14
2090 2 1 30
2090 1 0 35
2090 4 1 305
2090 6 1 328
2090 9 0 345
459 12 1 347
```

Das Datenformat der Anfragen

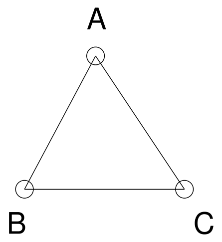
Die Anfragen liegen in einer Textdatei vor, in der jede Zeile aus drei ganzen Zahlen besteht, die eine Anfrage $(S, T, time)$ darstellen:

- der Index des Startbahnhofes S,
- der Index des Zielbahnhofes T, und
- die früheste Abfahrtzeit $time$ in Minuten nach Mitternacht.

Fahrplangraph

- *Knoten:* Ein Knoten für jedes Ereignis
- *Kanten:* (v, w) ist eine gerichtete Kante, falls
  1. v die *Abfahrt* eines Zuges, und
     w die nächste Ankunft dieses Zuges ist ($\rightarrow$ *Zugverbindung*), oder
  2. w der direkte Nachfolger von v bzgl. der Uhrzeit
     an *einem Bahnhof* ist ($\rightarrow$ *Aufenthalt*), oder
  3. v das letzte Ereignis und w das erste Ereignis an
     einem Bahnhof ist ($\rightarrow$ *Aufenthalt über Mitternacht*).
- Länge einer Kante (v, w):
  Die Differenz $time(w) - time(v)$ der Uhrzeiten der Ereignisse modulo 1440, in Minuten zwischen 0 und 1440.

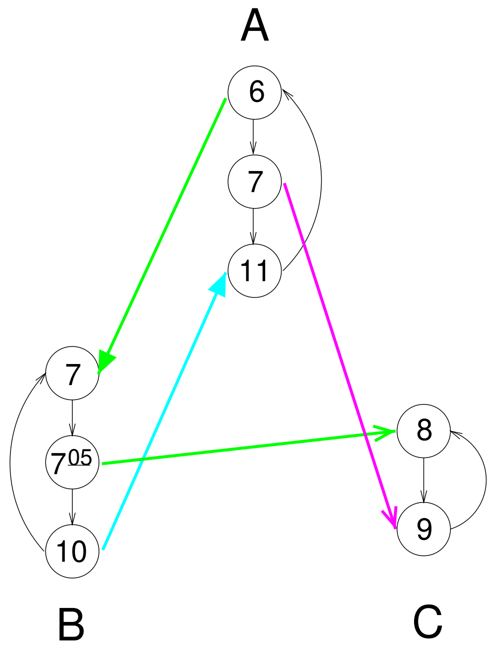
Fahrplangraph: Einfaches Beispiel



Train I		
leaving	A	6
arriving	B	7
leaving	B	7:05
arriving	C	8

Train II		
leaving	B	10
arriving	A	11

Train III		
leaving	A	7
arriving	C	9



Kürzeste Verbindung $\leftrightarrow$ Kürzester Weg

- Sei $(S, T, time)$ ein Anfrage
- s sei das Ereignis in S, mit
  - i) $time(s) > time$
  - ii) $time(s) - time$ minimal

  (falls es kein solches Ereignis mit $time(s) > time$ gibt sei s das Ereignis mit $time(s)$ minimal)
- Dann ist ein **kürzester Weg** im Fahrplangraph von s zu einem Ereignis in T eine **kürzeste Verbindung** bezüglich der Anfrage $(S, T, time)$.

Dijkstra's Algorithmus: Kürzeste Wege in (gerichteten) Graphen

Gegeben: Gerichteter Graph $G = (V, E)$, Startknoten $s \in V$, Endknoten $T \subset V$, Kantenlängen $l(e)$, Priorityqueue Q.

1. for all $v \in V$: $d(v) \leftarrow \infty$
2. $d(s) \leftarrow 0$ and insert(Q, s)
3. while not empty(Q) do
4. $v \leftarrow$ extract-minimum(Q)
5.   if $v \in T$ then **terminate** the algorithm
6.   for all outgoing edges $e = (v, w)$ of v do
7.   if $d(w) > d(v) + l(e)$ do
8. $d(w) \leftarrow d(v) + l(e)$
9.   insert(Q, w)/decreaseKey(Q,w)

JDSL - Data Structures Library in Java

- Interfaces
  - jdsl.graph.api.**InspectableGraph**
    $\rightarrow$ EdgeIterator *incidentEdges*(Vertex, EdgeDirection.IN)
  - jdsl.graph.api.**Vertex**
    $\rightarrow$ Object *element*()
  - jdsl.graph.api.**Edge**
    $\rightarrow$ Object *element*()
- Class
  - jdsl.graph.ref.**IncidenceListGraph**
    $\rightarrow$ Vertex *insertVertex*(Object)
    $\rightarrow$ Edge *insertDirectedEdge*(Vertex, Vertex, Object)

Vorschlag zur Vorgehensweise

- Fahrplandaten einlesen:
  → Klasse "Station"
  → Klasse "Event"
  → Package `java.io` verwenden: siehe `FileReader, LineNumberReader, StringReader, StreamTokenizer`
- Graph aufbauen:
  → Zuerst Knoten (mit Objekt "Event") und Aufenthalt-Kanten, dann die Zugverbindungskanten einfügen
- Dijkstra implementieren:
  → Aufbauend auf InspectableGraph/Vertex/Edge
  → Möglichst nicht spezialisieren auf Fahrplangraph
  → Abbruchkriterium von Instanz einer abstrakten Klasse/Interface prüfen lassen, die dem Algorithmus mit übergeben wird

Pfade & Links

- Verzeichnis: `gruppenname`/dijkstra
- Fahrplan- und Anfragedaten
  `~willhalm/lehre/inf4_W00/fahrplan`
- JDSL-Installation
  `/usr/local/lib/jdsl/jdsl-2.0.1-developer/lib/jdsl.jar`
  `/usr/local/lib/jdsl/jdsl-2.0.1-developer/doc/index.html`
- JDSL Page: `http://www.cs.brown.edu/cgc/jdsl/`