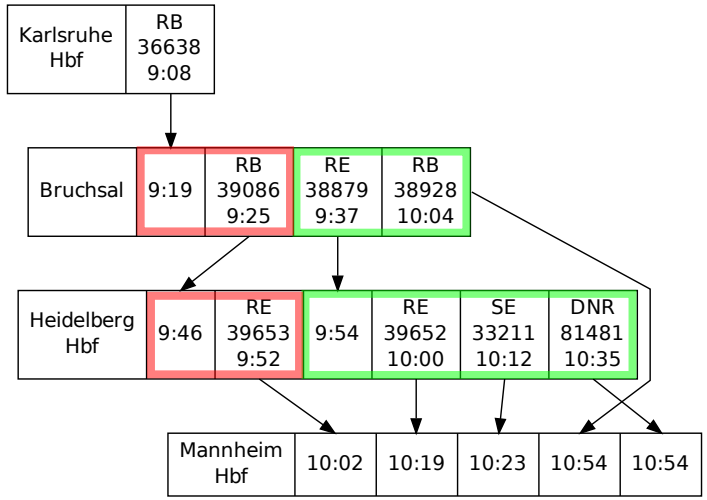
Karlsruhe Institute of Technology

Connection Scan

Julian Dibbelt, Thomas Pajor, Ben Strasser, Dorothea Wagner | 29. Juni 2015

INSTITUT FÜR THEORETISCHE INFORMATIK · ALGORITHMIK · PROF. DR. DOROTHEA WAGNER



Prüfungstermine

31. Juli

24. September

5. Oktober

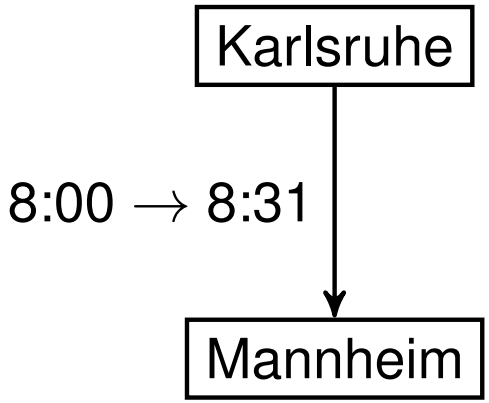
Anmeldung per E-Mail ans Sekretariat

(lilian.beckert@kit.edu)

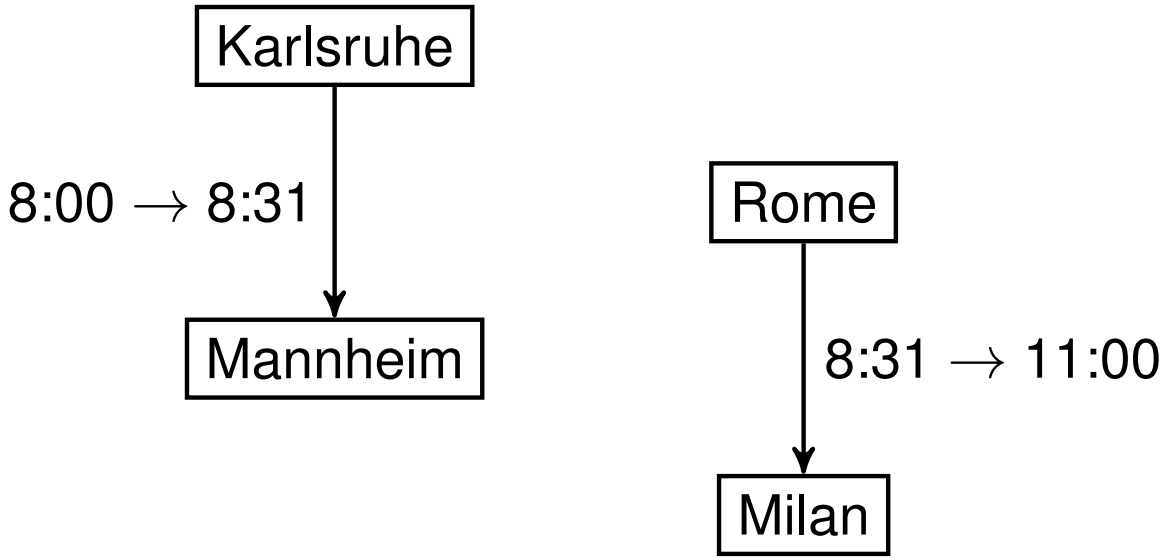
Was ist ein Fahrplan?

Karlsruhe

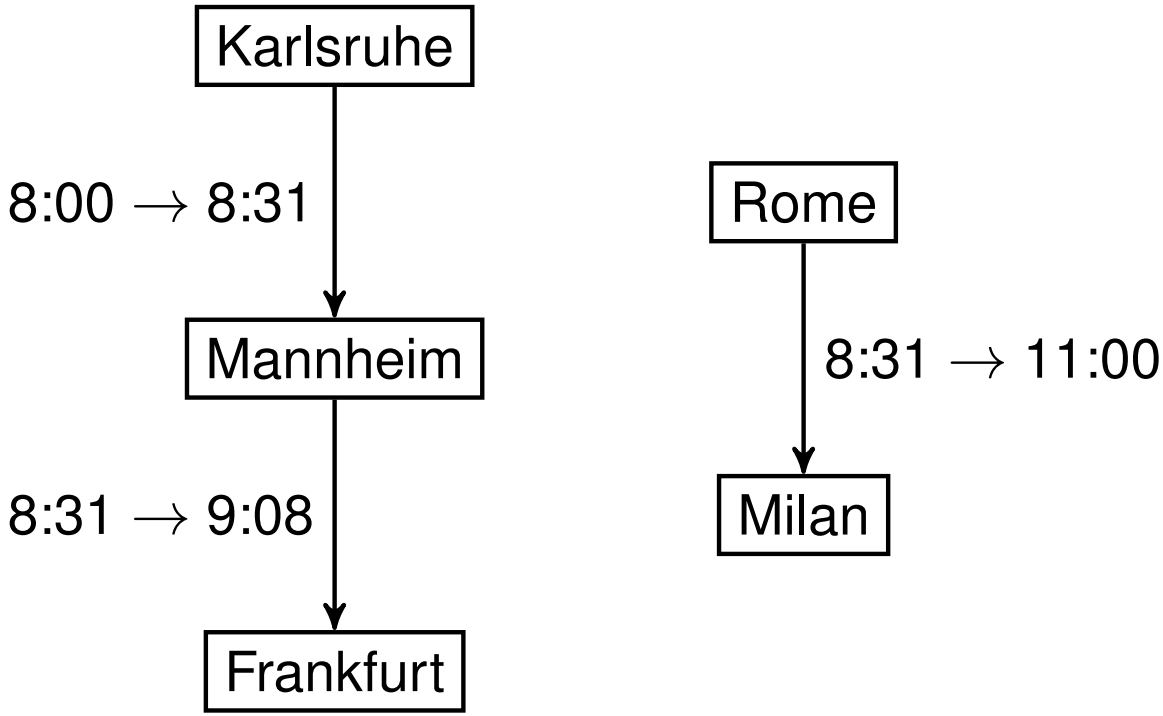
Was ist ein Fahrplan?



Was ist ein Fahrplan?



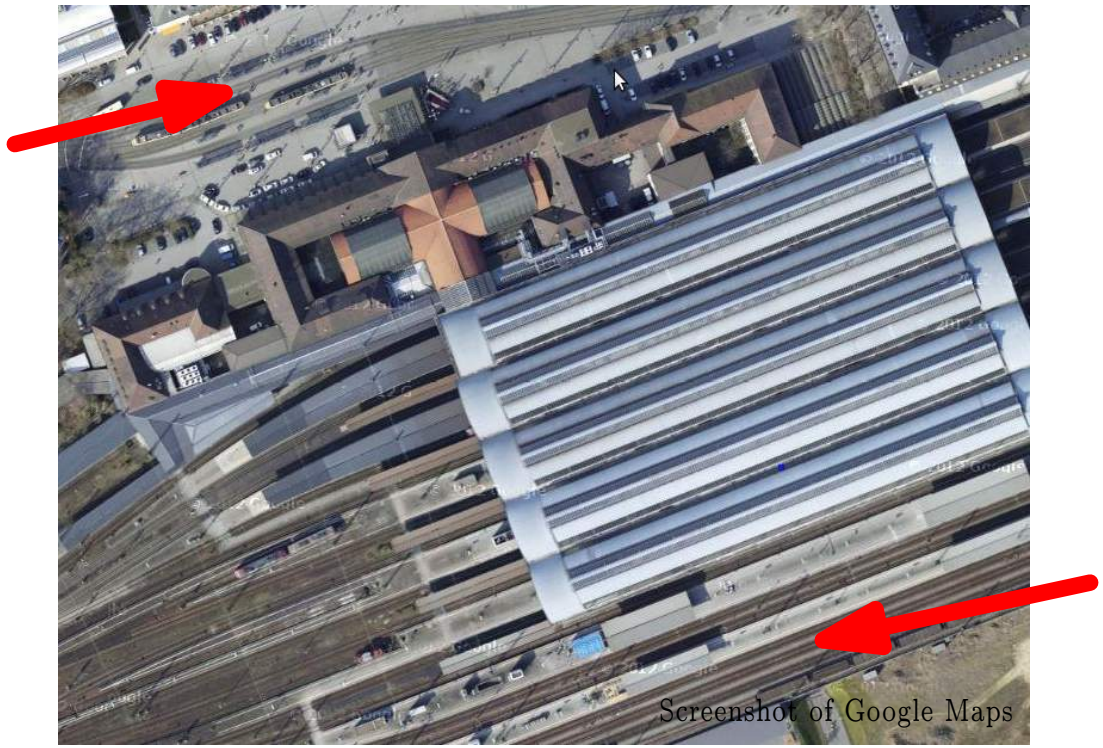
Was ist ein Fahrplan?



Was ist ein Fahrplan?

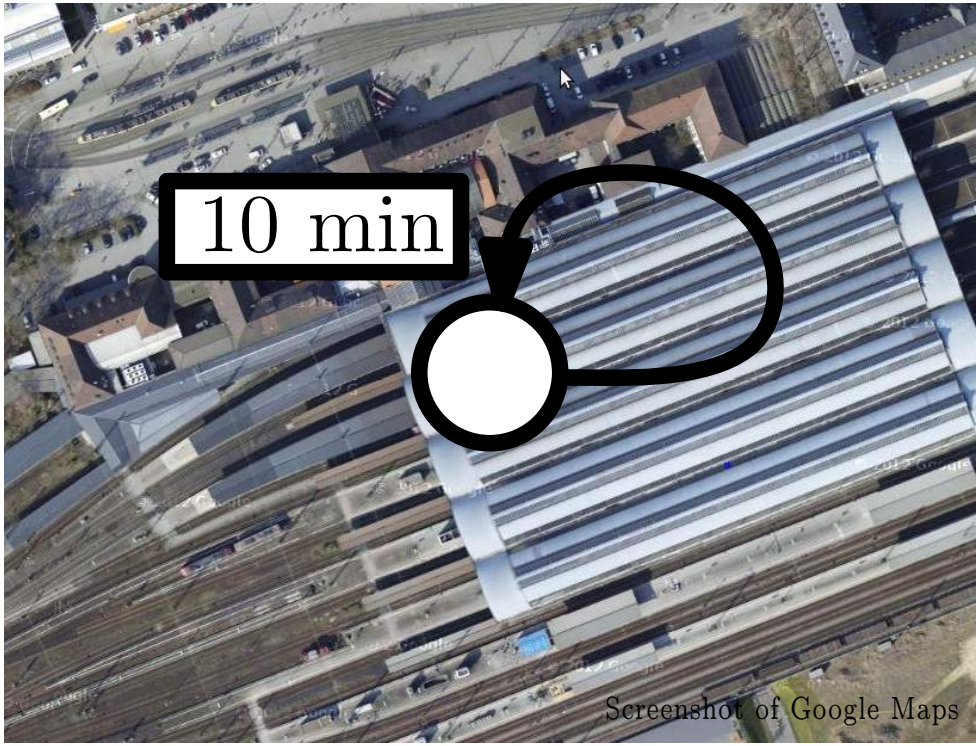
Ein Bahnhof

Was ist ein Fahrplan?



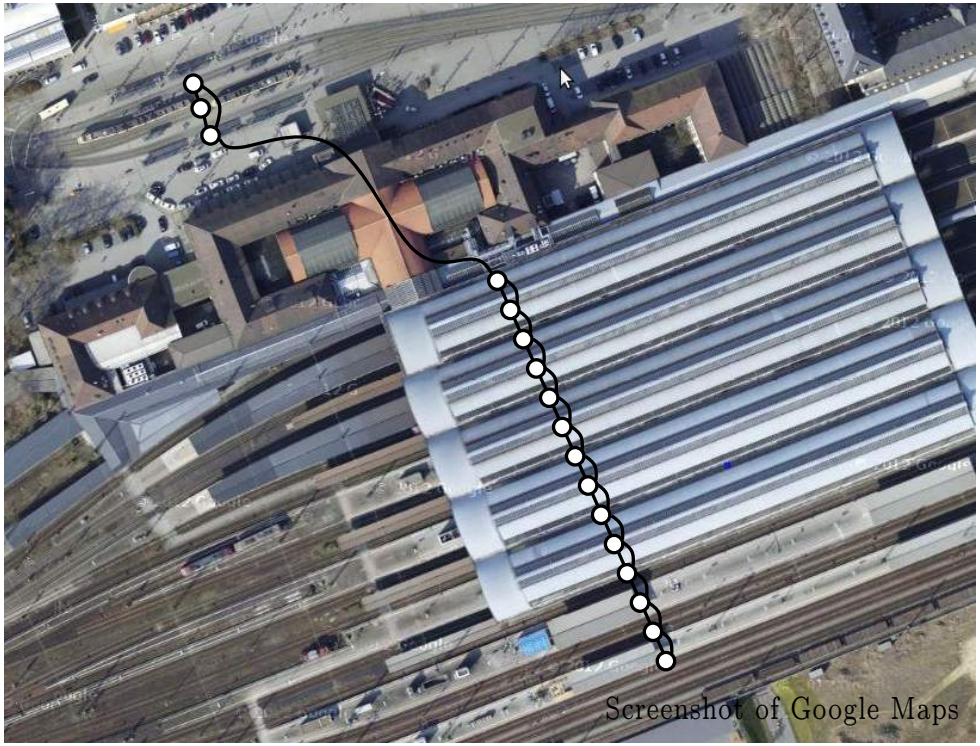
Schafft der Fahrgast das in 0 Sekunden?

Was ist ein Fahrplan?



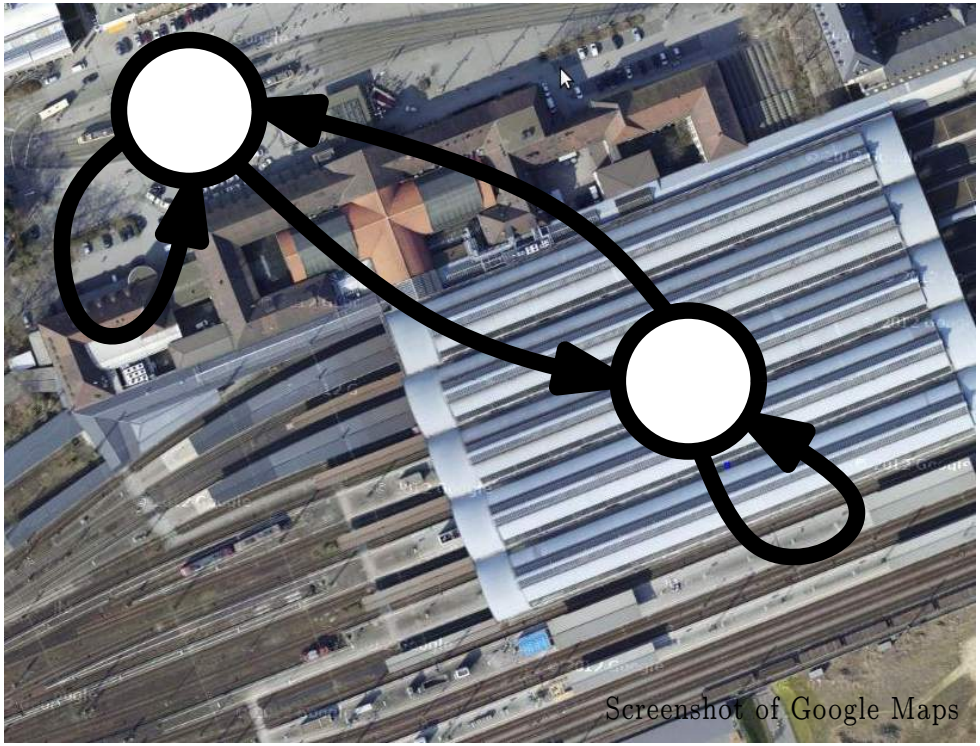
Ein Stop mit 10 min Umstiegszeit

Was ist ein Fahrplan?



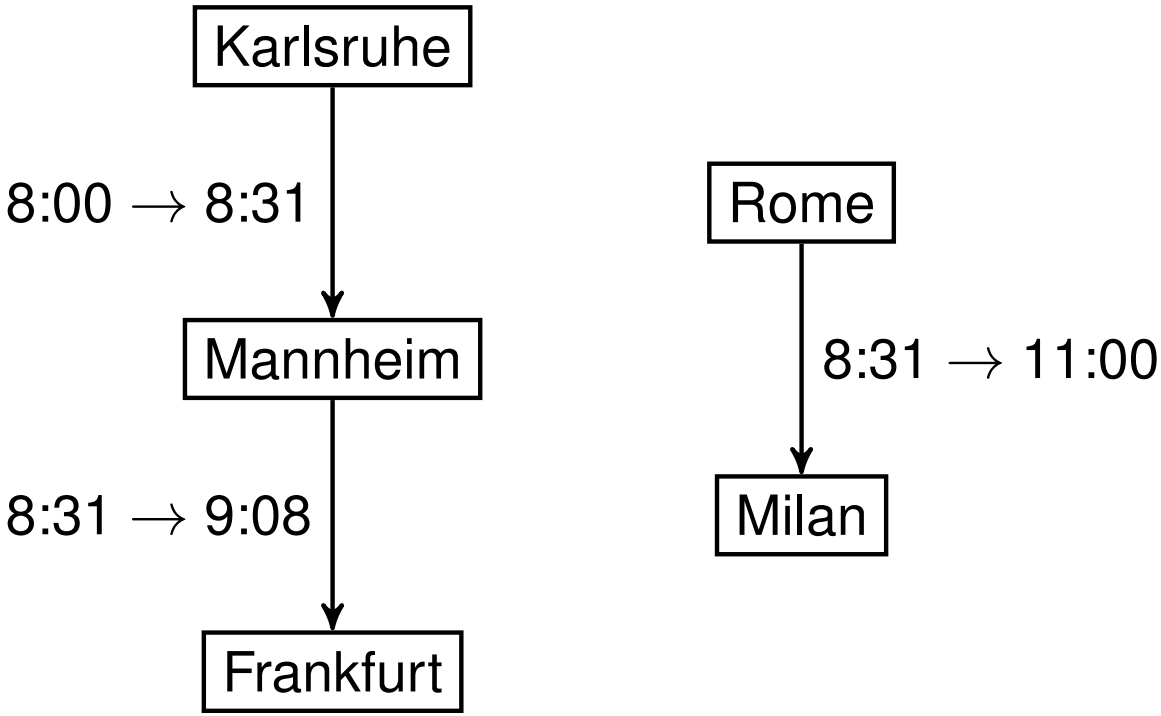
Viele Stops mit Fußwegen
Keine Umstiegszeit

Was ist ein Fahrplan?

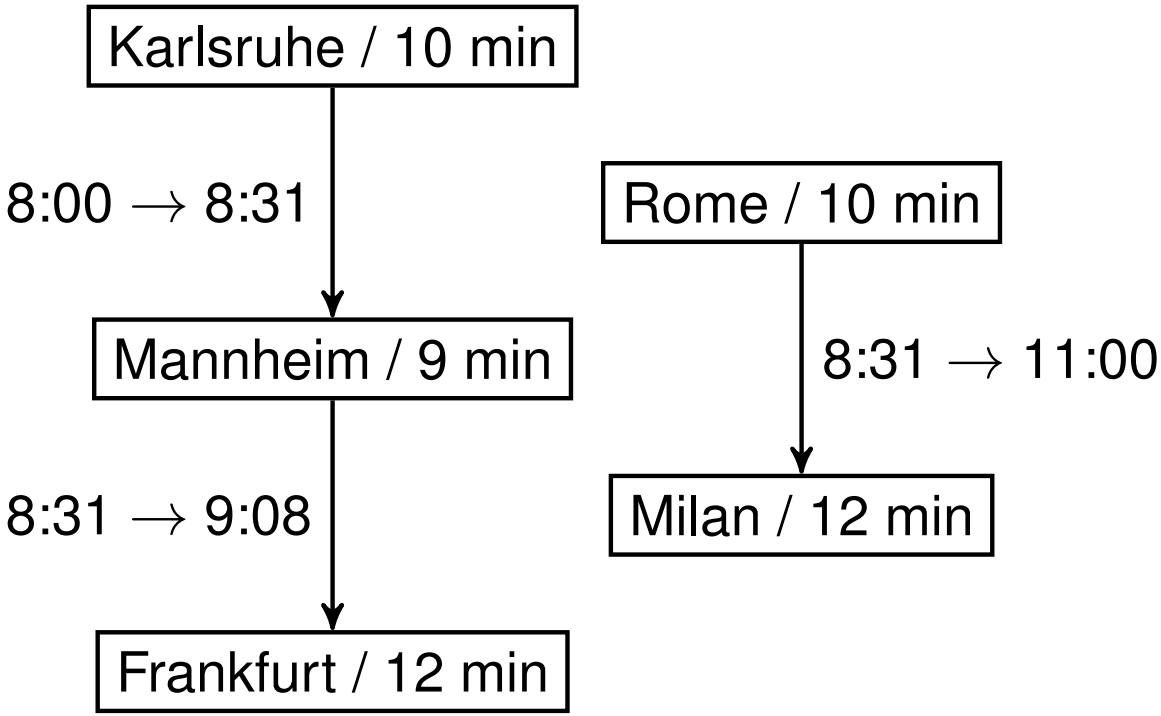


Mischung aus Fußwegen und Umstiegszeit

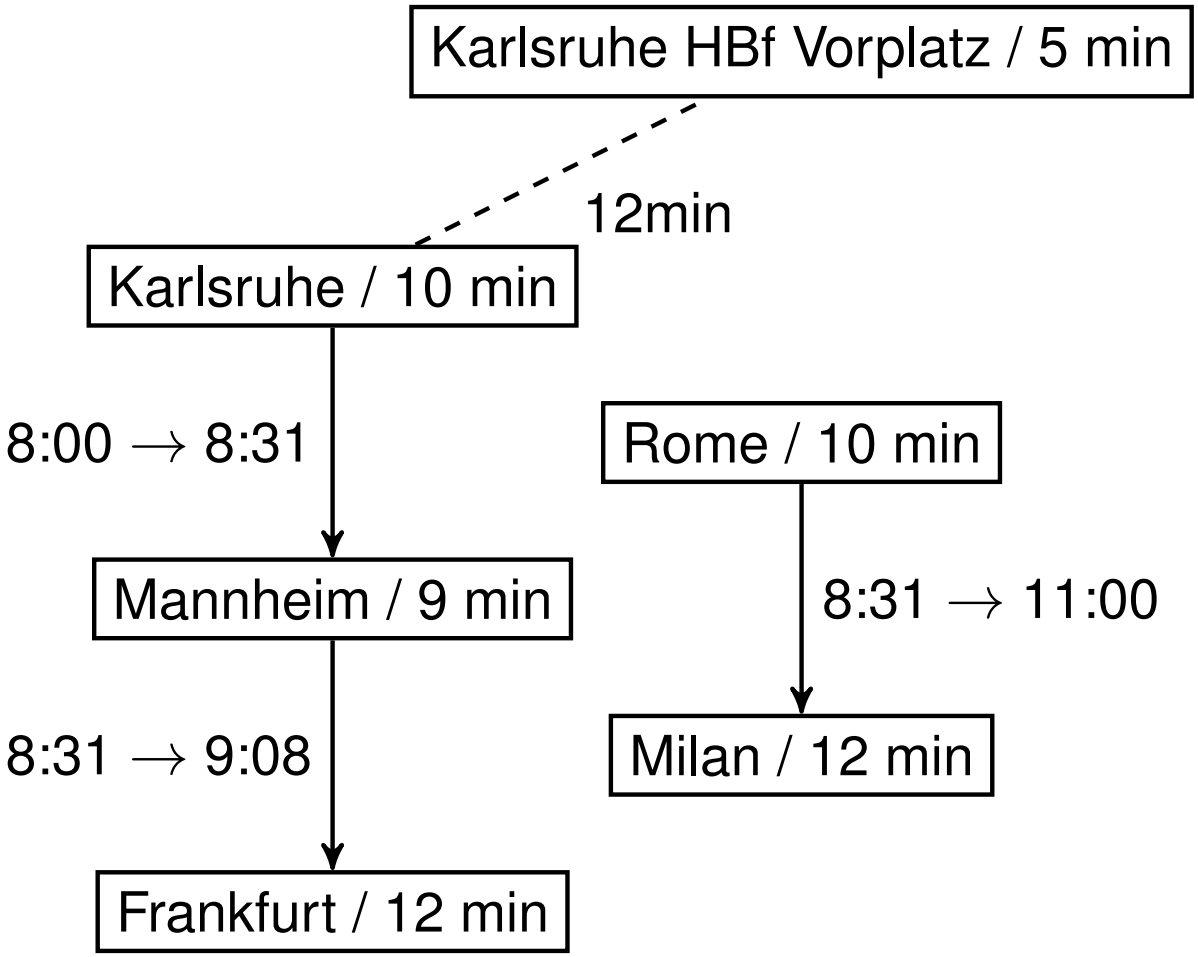
Was ist ein Fahrplan?



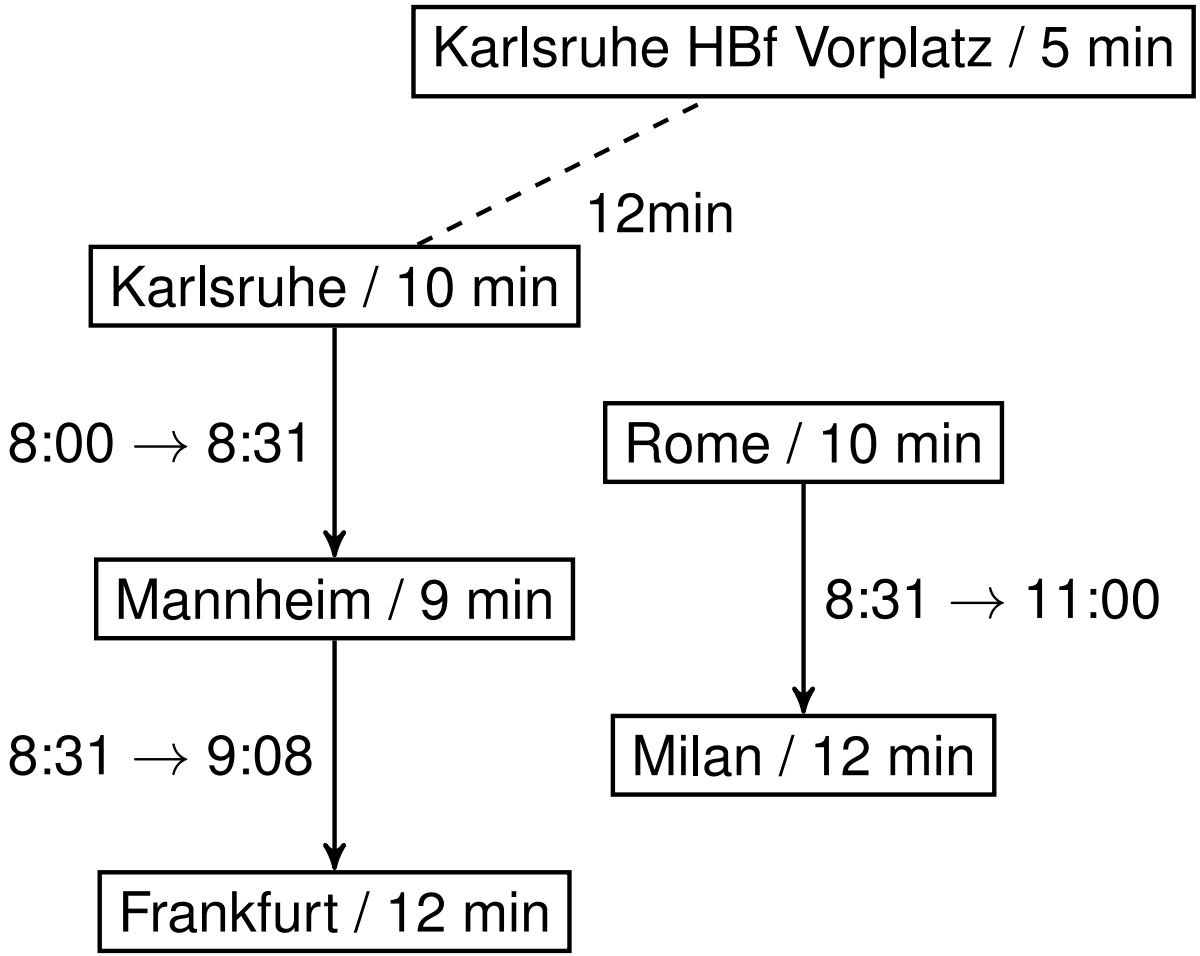
Was ist ein Fahrplan?



Was ist ein Fahrplan?



Was ist ein Fahrplan?



Vereinfachung: Keine Fußwege in dieser Vorlesung.

Was ist ein Fahrplan?

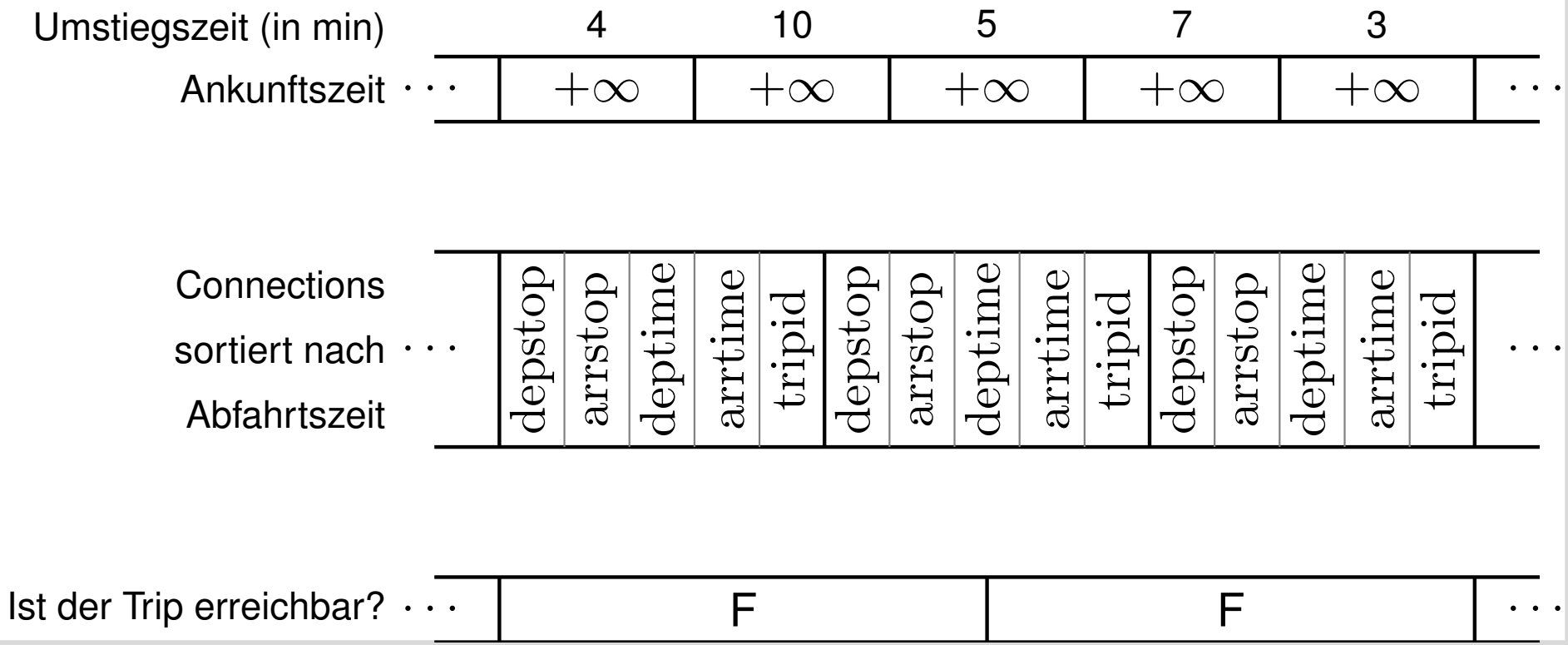
- Ein Fahrplan entählt Stops, Connections und Trips.
- Eine Connection is ein Zug der von Stop nach Stop fährt ohne Zwischenhalt.
- Ein Trip ist eine Folge von Connections des selben Zugs.
- Eine Connection hat eine Abfahrts- und Ankunftszeit und ein Abfahrts- und Ankunftsstop und eine Trip ID.
- Ein Fahrplan ist **aperiodisch**: Connections wiederholen sich nicht.
- Ein Weg durch den Fahrplan heißt Journey.
- Zugwechseln heißt Transfer.
- Transfers benötigen Zeit. Dies ist wie folgt umgesetzt:
  - Stops haben eine Umstiegszeit
    Jeder Transfer innerhalb eines Stops benötigt mindestens diese Zeit.
  - Fußwege verbinden Stops mit einer konstanten Laufgeschindigkeit.
    (Der Fußweggraph ist oft nicht zusammenhängend, d.h., die Hauptgleise eines Hbf sind mit der U-Bahn verbunden, aber benachbarte Stops der selben Tramlinie sind nicht verbunden.)

Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit

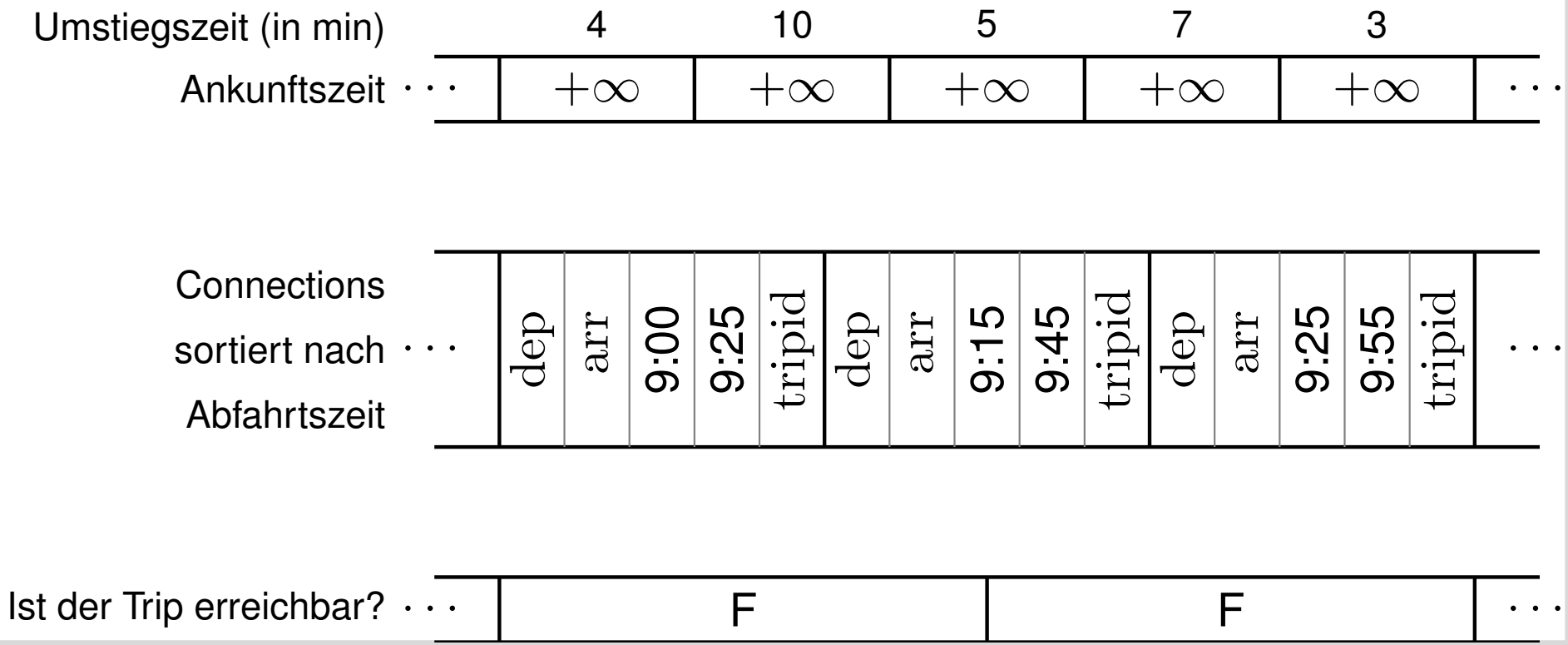


Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit

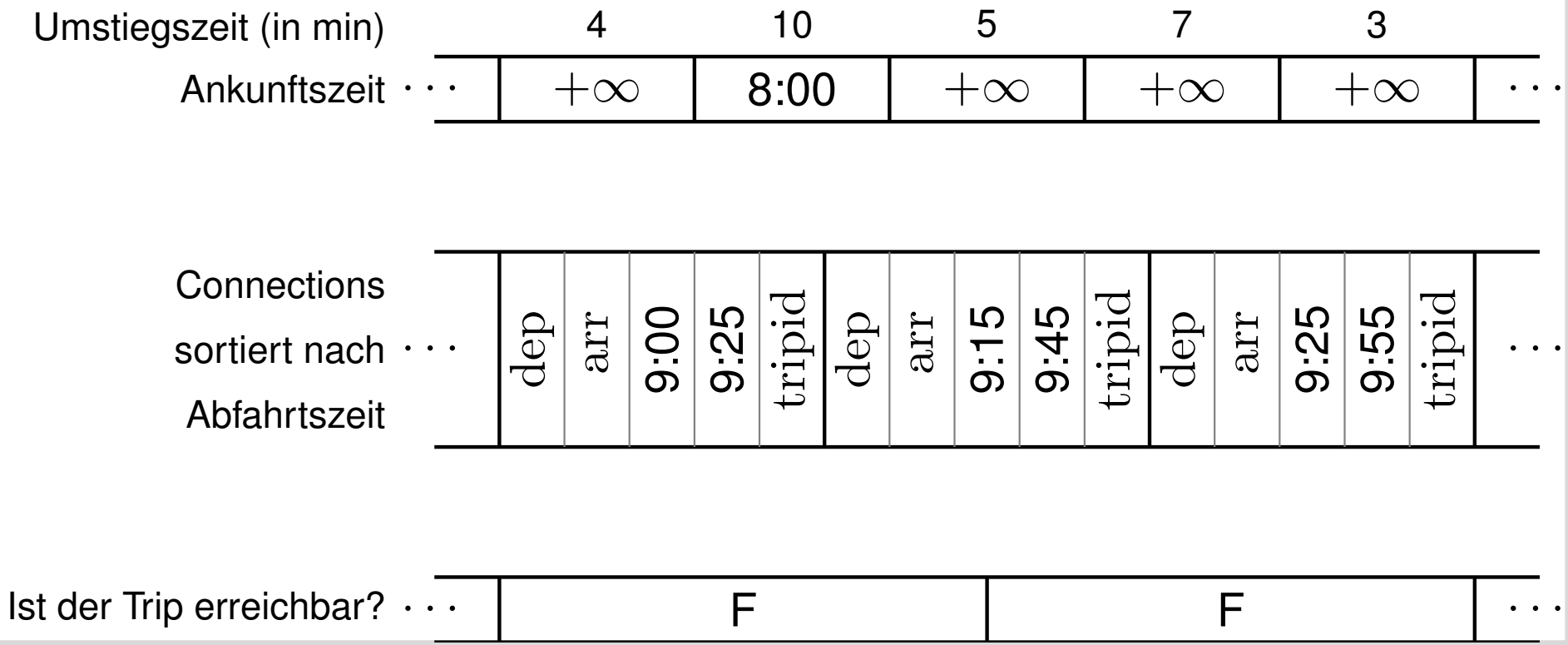


Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit

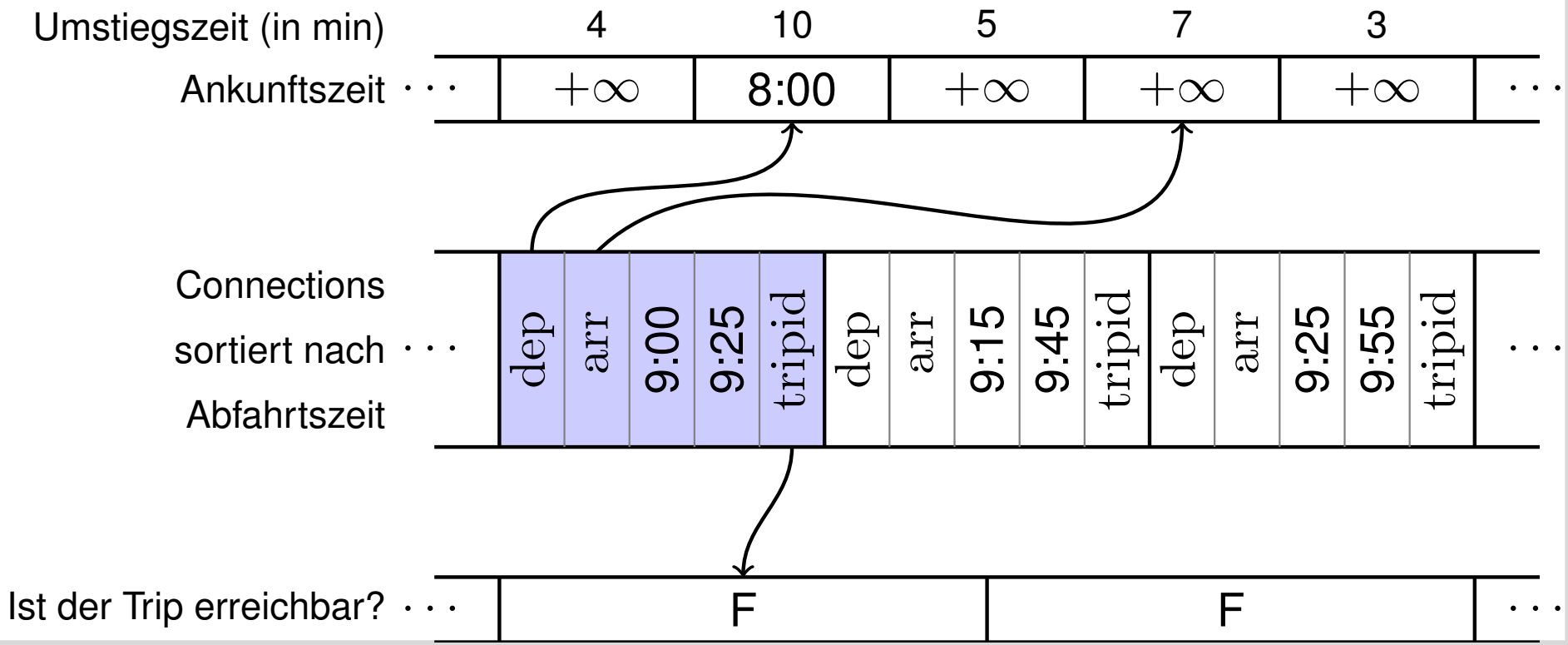


Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit

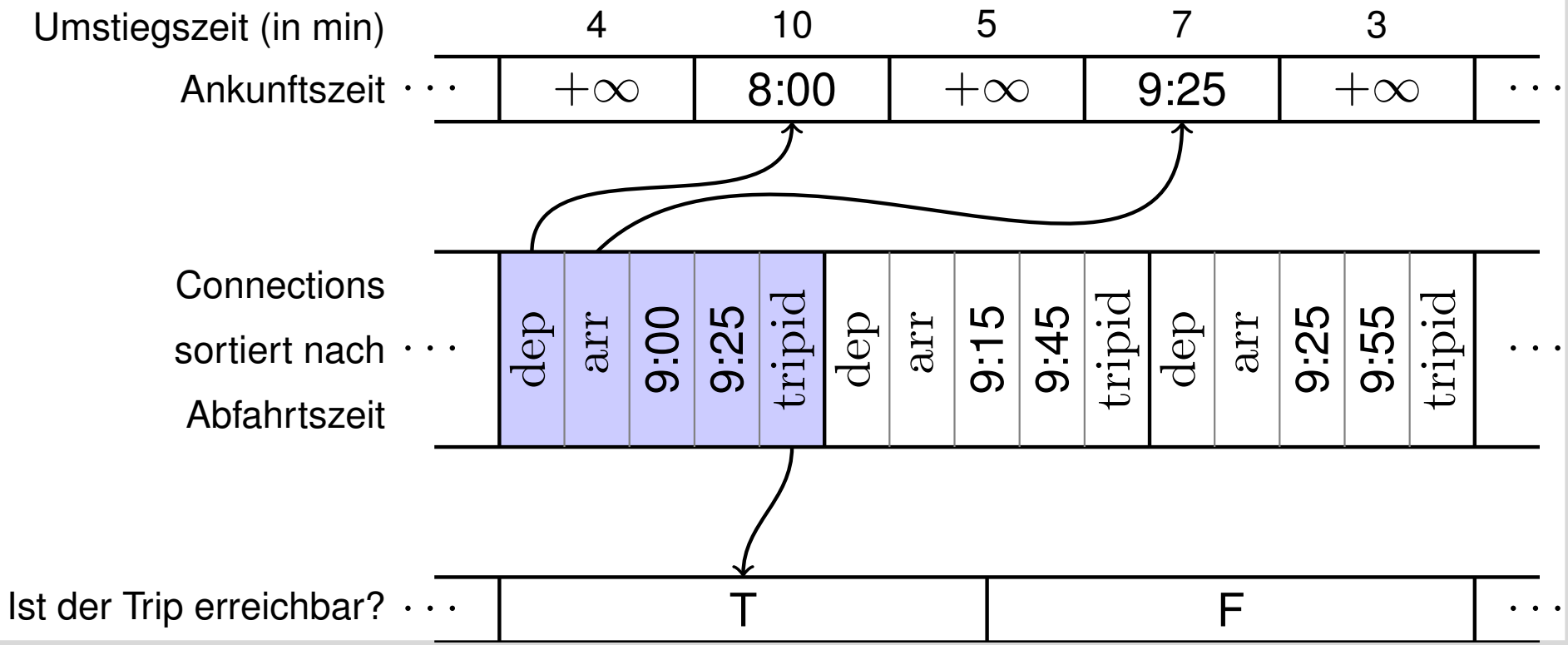


Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit

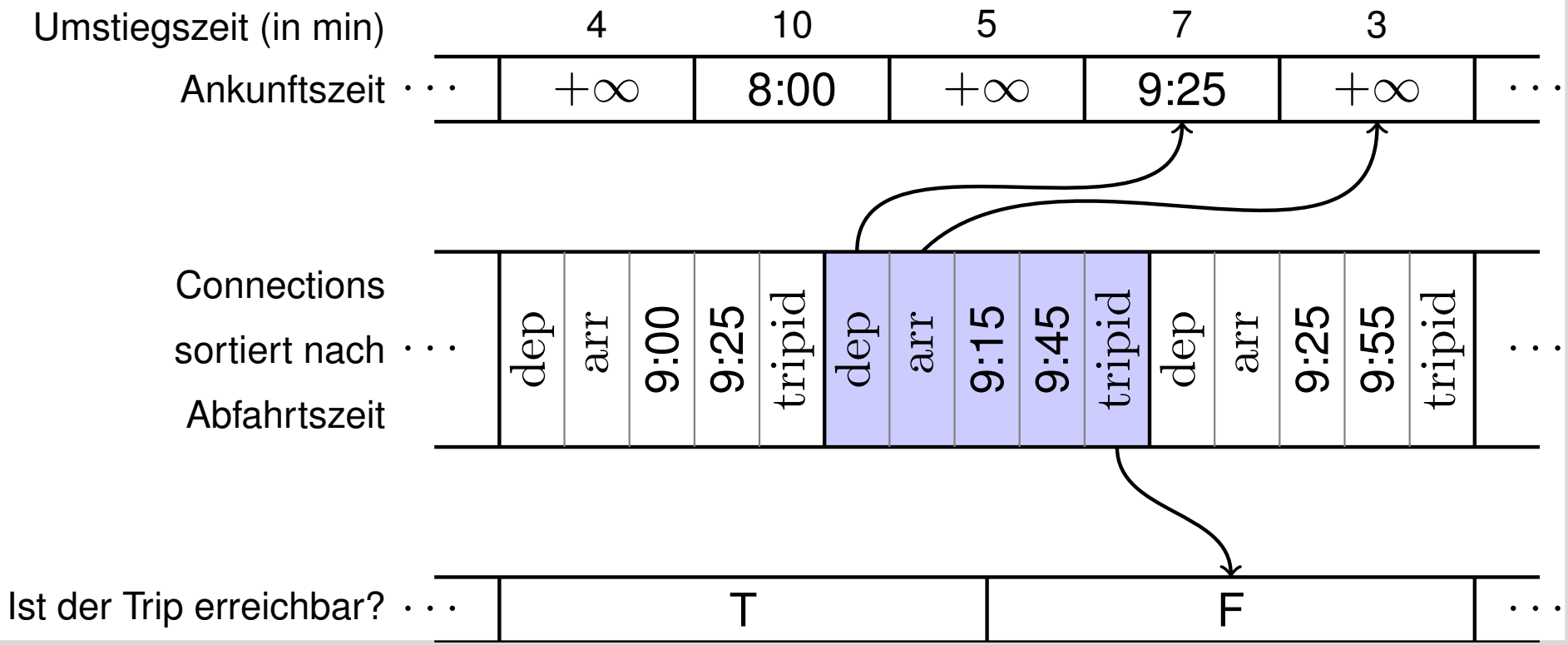


Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit

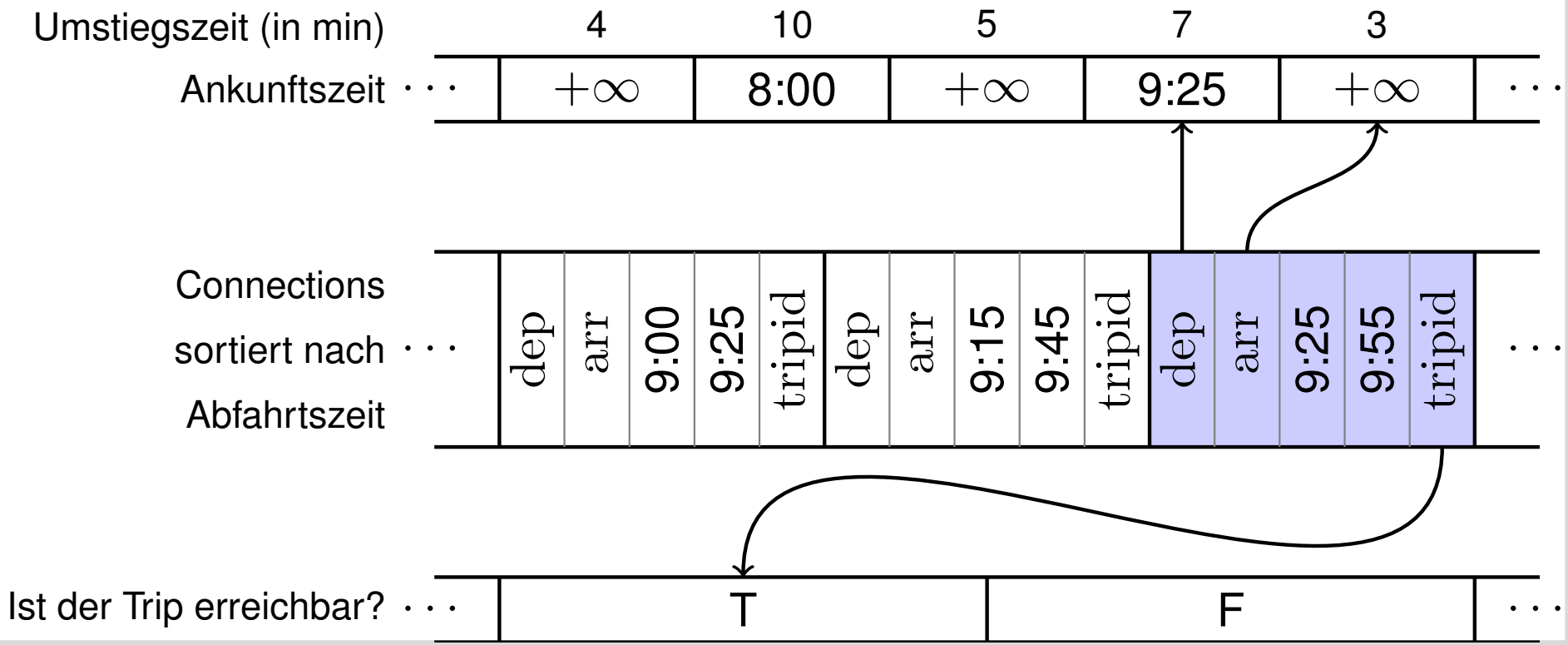


Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit

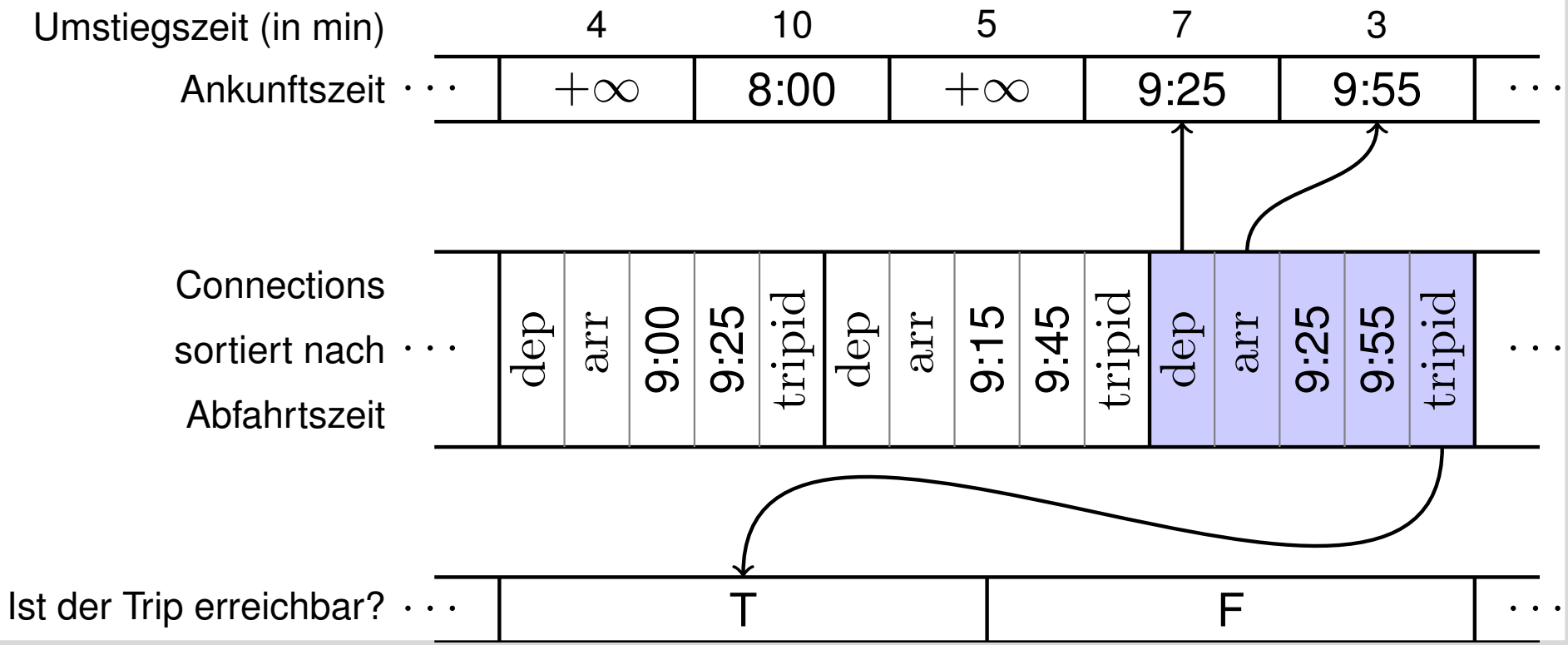


Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit

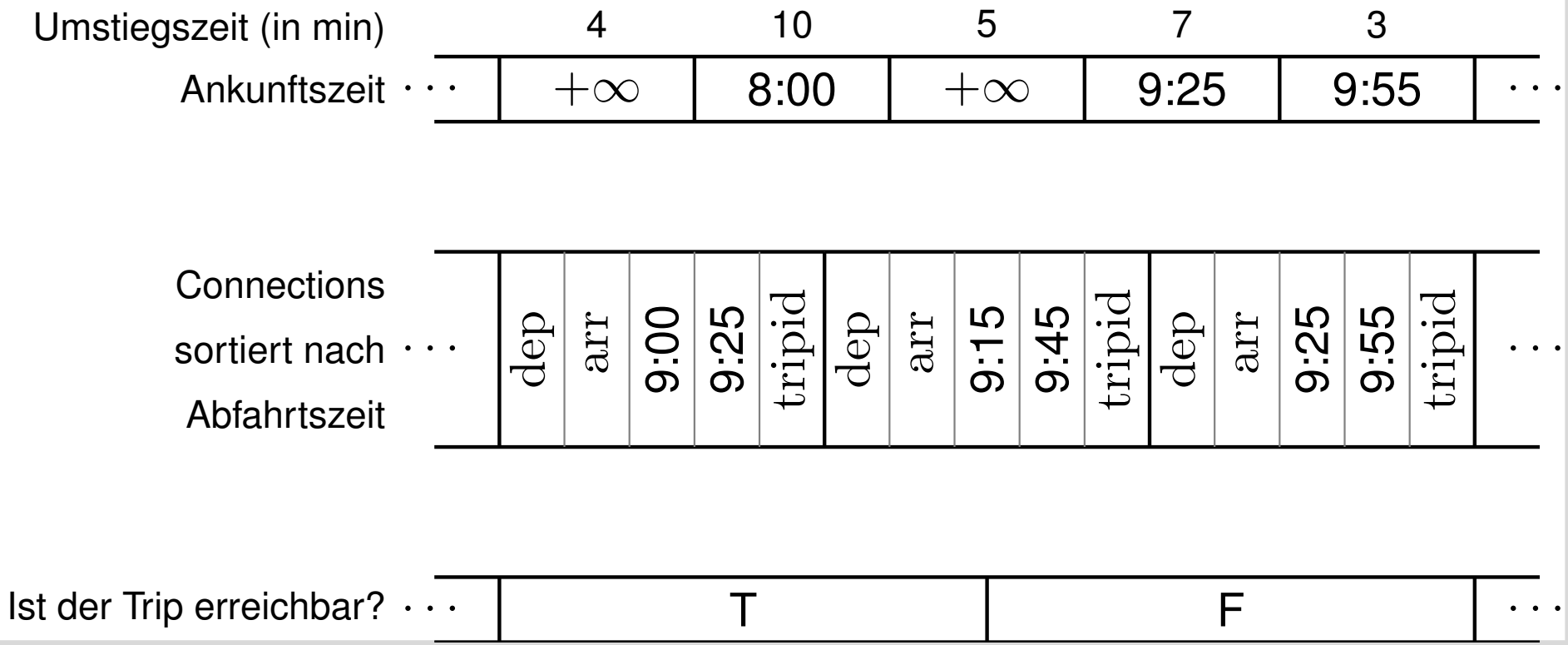


Grundlegender Connection Scan

Problem der Frühesten Ankunft

Eingabe: Sortiere Connectionliste, Startstop, Startzeit, Zielstop

Ausgabe: Früheste Ankunftszeit



Startkriterium

Beobachtung: Connections vor der Abfahrtszeit können nicht verwendet werden.

$\Rightarrow$ Per binärer Suche die erste connection bestimmen, die nicht vor der Startzeit abfährt und erst ab dieser scanen.

Stopkriterium

Bisher: Wir lösen das one-to-all Problem.

Frage: Geht es besser, wenn wir den Zielstop t kennen?

Stopkriterium

Bisher: Wir lösen das one-to-all Problem.

Frage: Geht es besser, wenn wir den Zielstop t kennen?

Observation: Züge die abfahren, nach der Ankunftszeit an t sind nie nützlich.

$\Rightarrow$ Scan abbrechen, wenn die Zeit an t nicht mehr größer ist als die Abfahrtszeit der aktuell gescannten Connection.

Profil Anfragen

Problem: Der Fahrgast kennt seine Abfahrts- und Ankunftszeit oft nicht.
Lösung: Journeys für eine Zeitspanne angeben.

Profil Anfragen

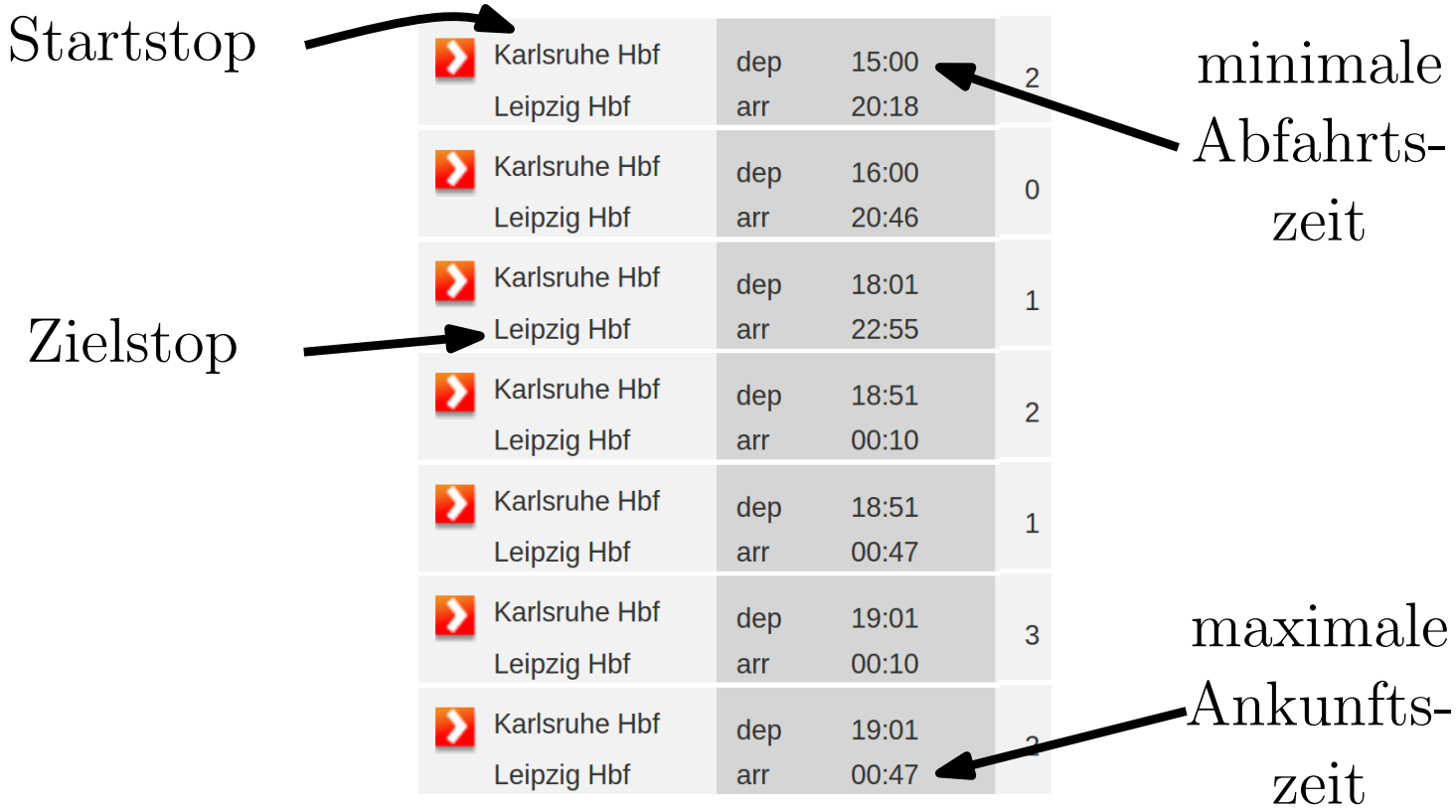
Problem: Der Fahrgast kennt seine Abfahrts- und Ankunftszeit oft nicht.
Lösung: Journeys für eine Zeitspanne angeben.

Karlsruhe Hbf	dep	15:00	2
Leipzig Hbf	arr	20:18	
Karlsruhe Hbf	dep	16:00	0
Leipzig Hbf	arr	20:46	
Karlsruhe Hbf	dep	18:01	1
Leipzig Hbf	arr	22:55	
Karlsruhe Hbf	dep	18:51	2
Leipzig Hbf	arr	00:10	
Karlsruhe Hbf	dep	18:51	1
Leipzig Hbf	arr	00:47	
Karlsruhe Hbf	dep	19:01	3
Leipzig Hbf	arr	00:10	
Karlsruhe Hbf	dep	19:01	2
Leipzig Hbf	arr	00:47	

Screenshot von bahn.de

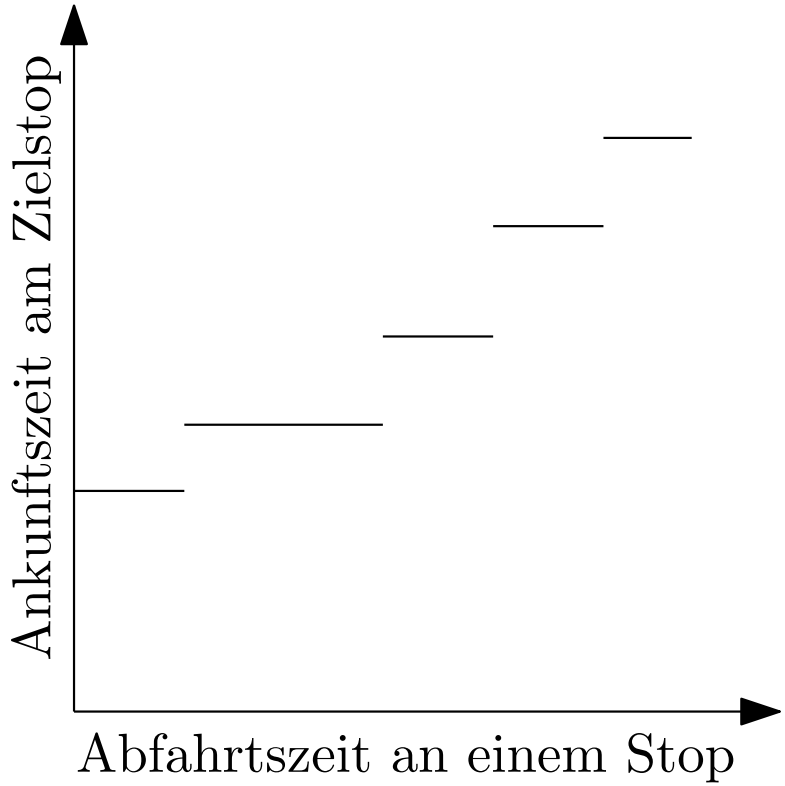
Profil Anfragen

Problem: Der Fahrgast kennt seine Abfahrts- und Ankunftszeit oft nicht.
Lösung: Journeys für eine Zeitspanne angeben.



Screenshot von bahn.de

Profil Connection Scan

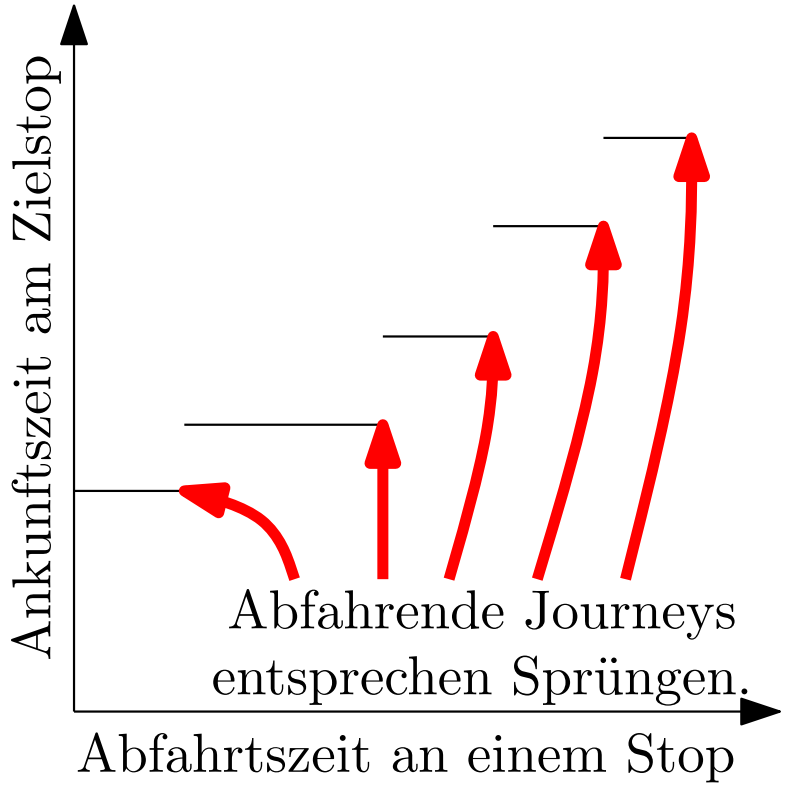


Problem der frühesten Rückwärtsankunftsprofile

Eingabe: Fahrplan, Zielstop t

Ausgabe: st-Profil für jeden Stop s (außer t)

Profil Connection Scan



Problem der frühesten Rückwärtsankunftsprofile

Eingabe: Fahrplan, Zielstop t

Ausgabe: st-Profil für jeden Stop s (außer t)

Profil Connection Scan

Kernidee: Dynamische Programmierung

Bei der Ankunft einer Connection gibt es für den Fahrgast 3 Optionen:

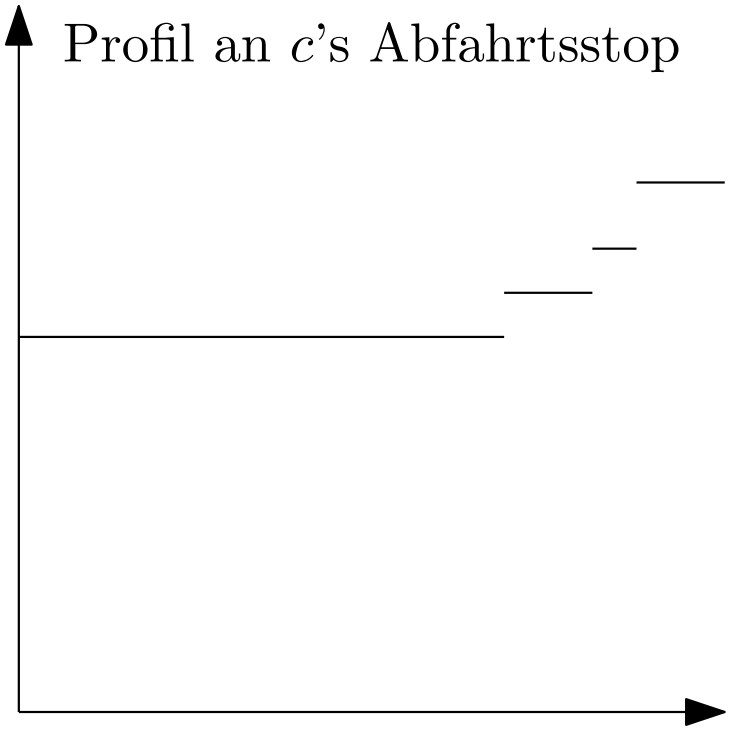
- Er bleibt sitzen.
- Er verlässt den Zug und wartet auf einen anderen.
- Er beendet seine Reise (geht nur am Zielstop).

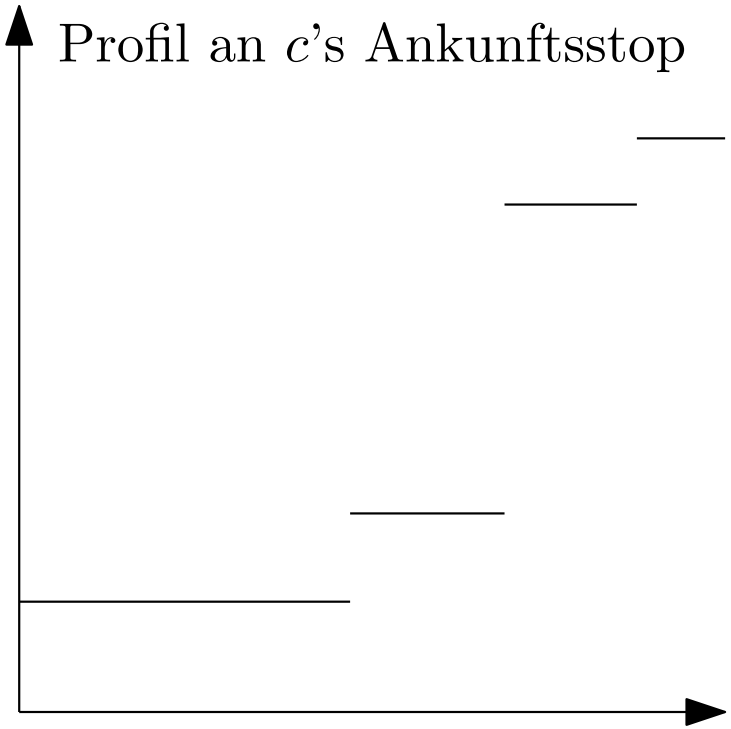
Profile-Repräsentation: Speichere Profilfunktion as dynamisches Array von geordneten $(\mathrm{deptime}, \mathrm{arrtime})$-Paaren

Hinweis: Die Paar sind sowohl aufsteigend nach $\mathrm{deptime}$ als auch nach $\mathrm{arrtime}$ geordnet.

Profil Connection Scan

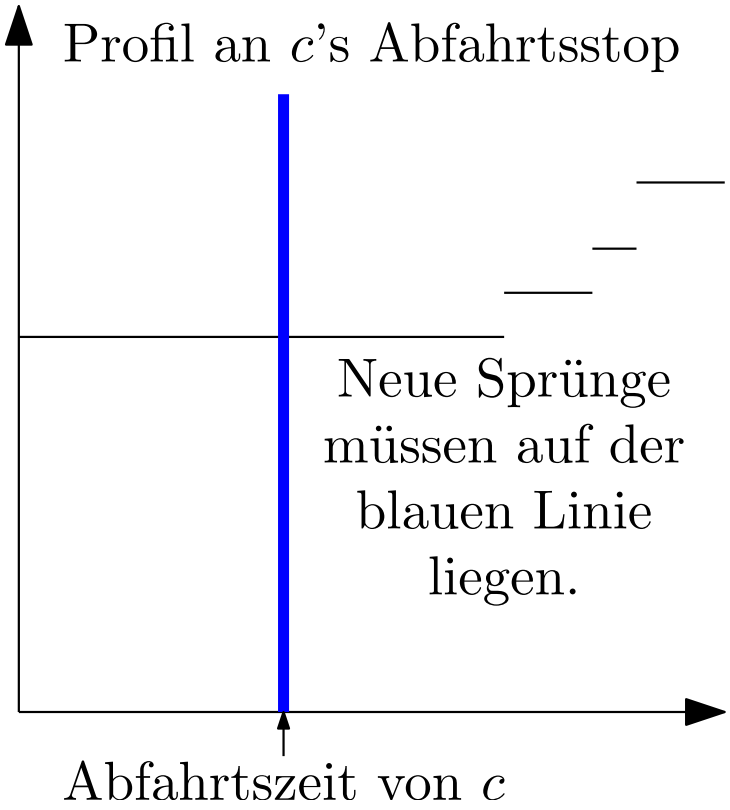
Für jede Connection c absteigend nach Abfahrtszeit:

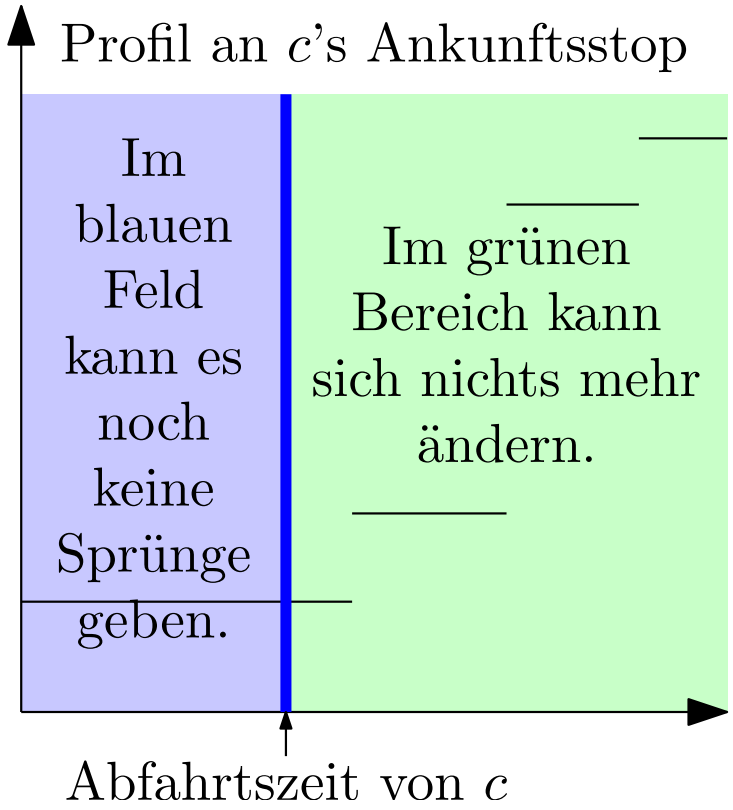




Profil Connection Scan

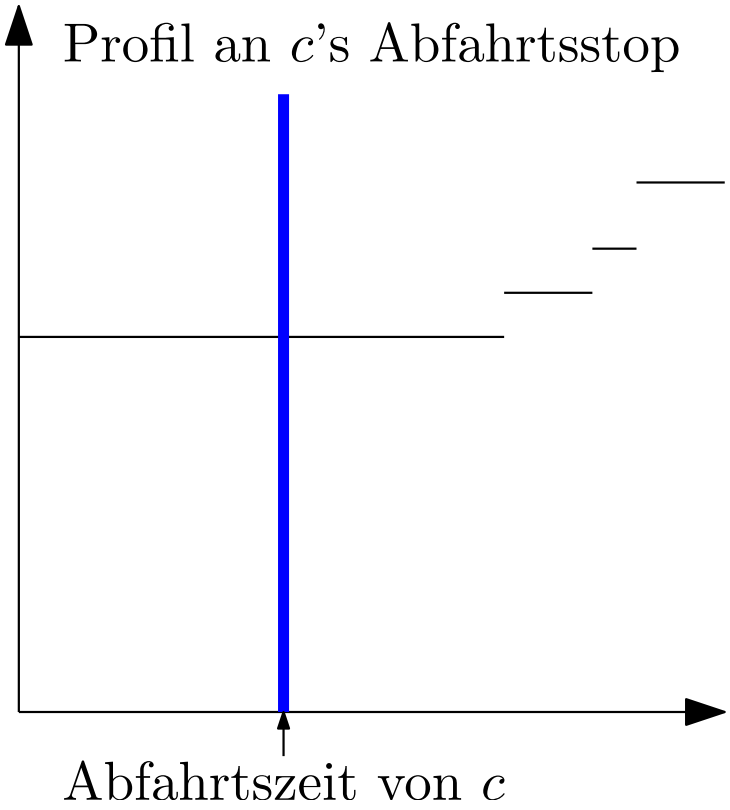
Für jede Connection c absteigend nach Abfahrtszeit:

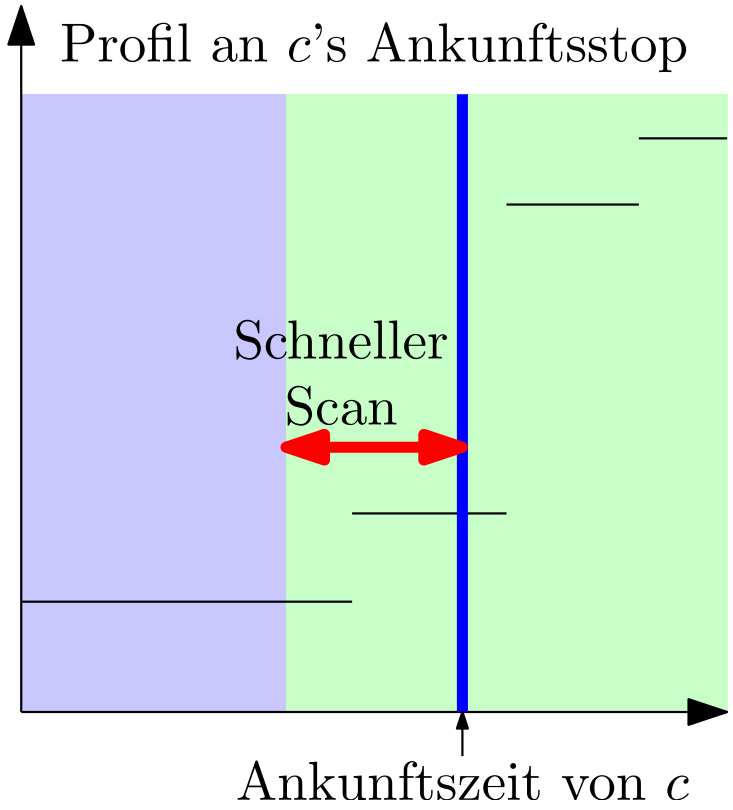




Profil Connection Scan

Für jede Connection c absteigend nach Abfahrtszeit:

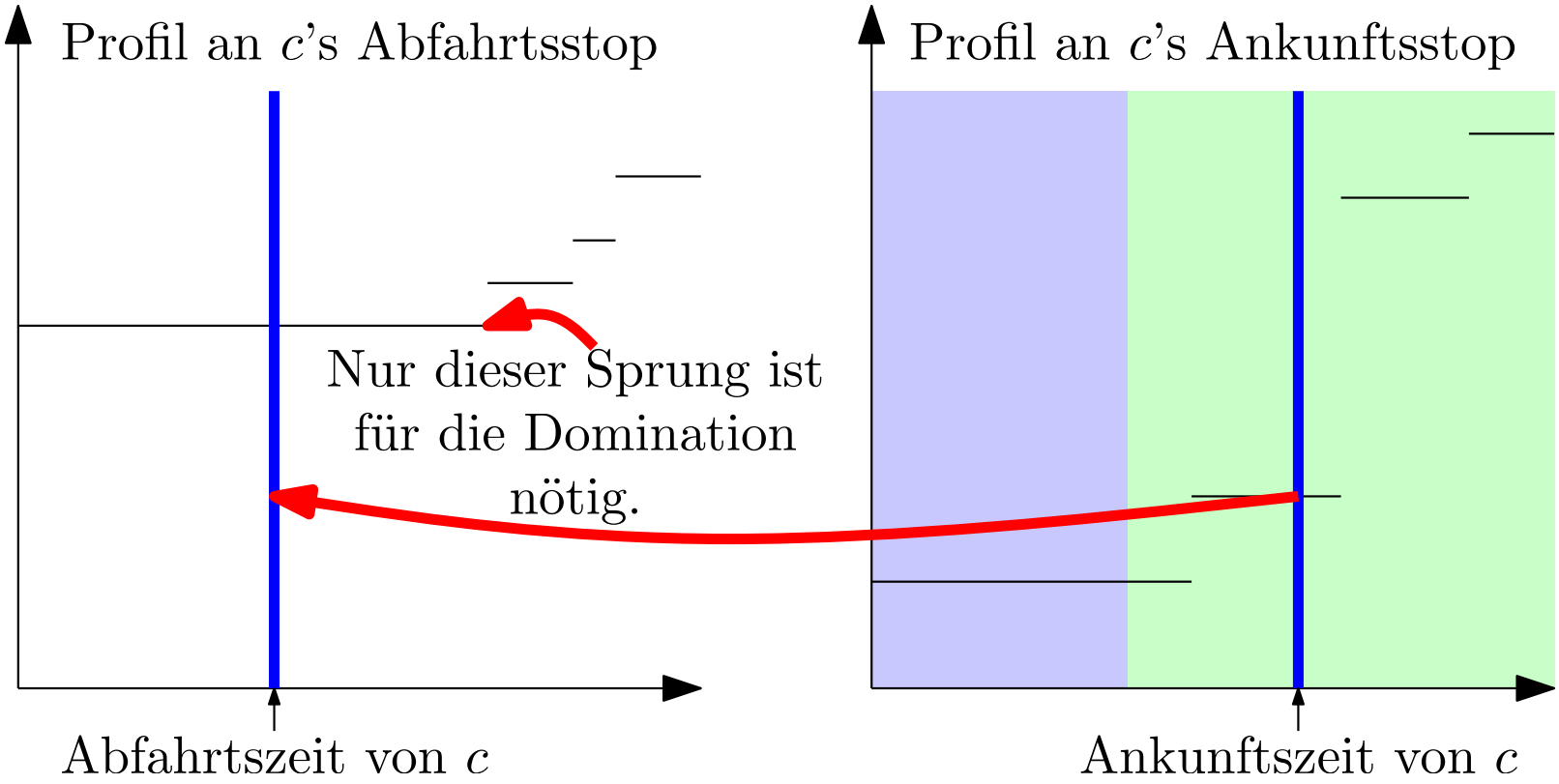




In der Praxis: sehr kurzer lineare Scan.

Profil Connection Scan

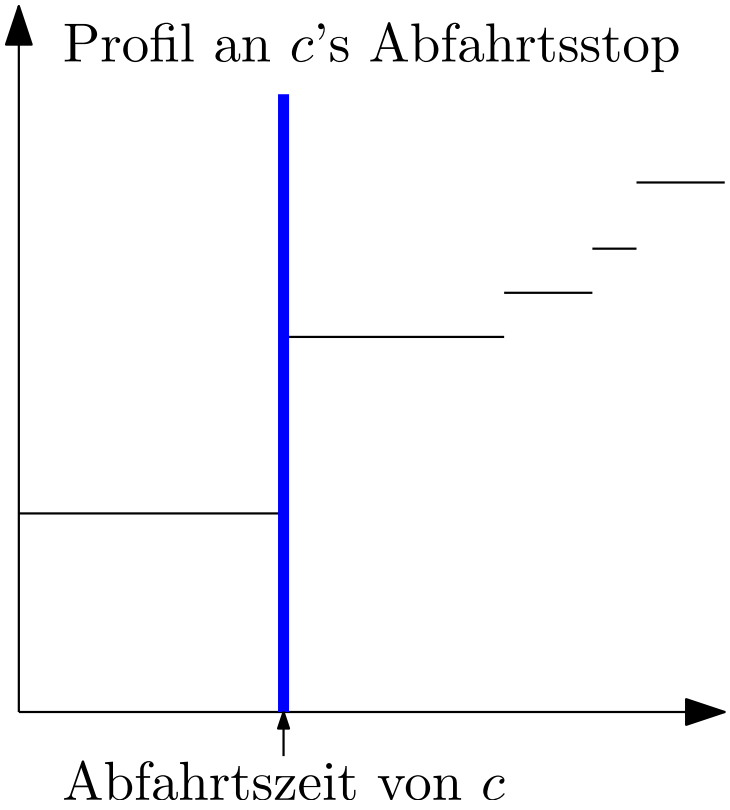
Für jede Connection c absteigend nach Abfahrtszeit:

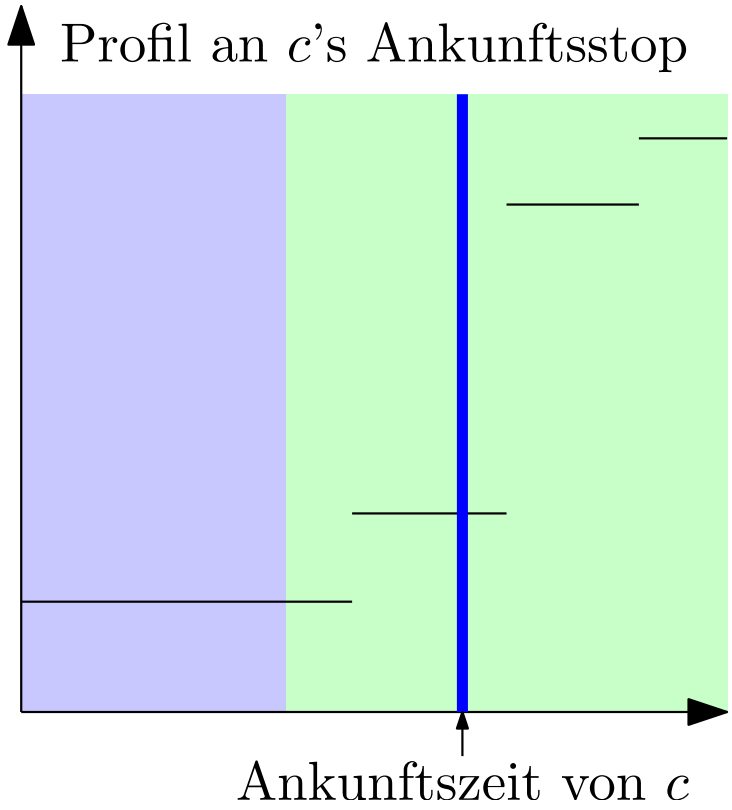


Teste ob der neue Sprung über oder unter dem bereits existierenden ist.

Profil Connection Scan

Für jede Connection c absteigend nach Abfahrtszeit:





Neuen Sprung einfügen, wenn er unterhalb ist. (Im Beispiel ist er unterhalb.)

Profil Connection Scan

$P(s)$: Profil von s, ein Array von Paaren;
$T(t)$: Früheste Ankunft von Trip t, ein einzelner Zeitpunkt;

$P(s) \leftarrow \{(\infty, \infty)\}$ für jeden Stop s;
$T(t) \leftarrow \infty$ für jeden Trip t;
for *jede Connection c absteigend nach* c_{deptime} **do**
    $t \leftarrow$ evaluiere $P(c_{\text{arrstop}})$ um $c_{\text{arrtime}} + \text{minchange}(c_{\text{arrstop}})$;
    $t \leftarrow \min(t, T(c_{\text{tripid}}))$;
    if $c_{\text{arrstop}} =$ *Zielstop* **then**
        $t \leftarrow \min(t, c_{\text{arrtime}})$;
    $T(c_{\text{tripid}}) \leftarrow t$;
    if $t < P(c_{\text{depstop}})[\text{front}].\,\text{arrtime}$ **then**
        Hänge (c_{deptime}, t) an den Anfang von $P(c_{\text{depstop}})$;

Wie Profile Evaluieren?

Das Evaluieren eines Profils um τ besteht daraus, der erste Paar (d, a) zu finden, mit $\tau \leq d$.

Option 1:

- $c_{\text{arrtime}} - c_{\text{deptime}}$ ist klein vergleichen mit dem Zeithorizont des Fahrplans.
- Lineare Suche funktioniert deswegen gut.
- Es im Durchschnitt weniger als 2 Paare angeschaut.
- Quasi $O(1)$.

Wie Profile Evaluieren?

Option 2: Modifiziere den Algorithmus leicht. Ersetze:

---

if $t < P(c_{\text{depstop}})[\text{front}].\,\text{arrtime}$ **then**
  Hänge (c_{deptime}, t) an den Anfang von $P(c_{\text{depstop}})$;

---

durch

Hänge $(c_{\text{deptime}}, \min(t, P(c_{\text{depstop}})[\text{front}].\,\text{arrtime}))$
  an den Anfang von $P(c_{\text{depstop}})$;

- Die Abfahrtszeit sind nun unabhängig vom Zielstop.
- Es werden also immer die selbe Paare generiert mit den selben Abfahrtszeiten (aber unterschiedlichen Ankunftszeiten), egal was der Zielstop ist.
- In einem schnell Vorberechnungsschritt: Speichere für jede Connection die Position c_{eval} des relevanten Paars.
- $O(1)$ Evaluierung $\rightarrow$ Gesamtlaufzeit in $O(\#\text{connections})$

Profile Connection Scan

$P(s)$: Profil von s;
$T(t)$: Früheste Ankunft von Trip t, ein einzelner Zeitpunkt;
c_{eval} : Evaluierungspaar ID, unabhängig vom Zielstop;

$P(s) \leftarrow \{(\infty, \infty)\}$ für jeden Stop s;
$T(t) \leftarrow \infty$ für jeden Trip t;
for *jede Connection c absteigend nach* c_{deptime} **do**
    $t \leftarrow P(c_{\text{arrstop}})[c_{\text{eval}}]$;
    $t \leftarrow \min(t, T(c_{\text{tripid}}))$;
    if $c_{\text{arrstop}} =$ *Zielstop* **then**
        $t \leftarrow \min(t, c_{\text{arrtime}})$;
    $T(c_{\text{tripid}}) \leftarrow t$;
    Hänge $(c_{\text{deptime}}, \min(t, P(c_{\text{depstop}})[\text{front}].\,\text{arrtime}))$
        an den Anfang von $P(c_{\text{depstop}})$;

Eingeschränktes Profil

Die Eingabe enthält auch:

- Eine minimale Abfahrtszeit d
- Eine maximale Ankunftszeit a

Optimierung:

- Scanne nur die Connections c mit $d \leq c_{\text{deptime}} \leq a$.
- Dieses Subarray kann man mit zwei binären Suchen bestimmen.

Eingeschränktes Profil

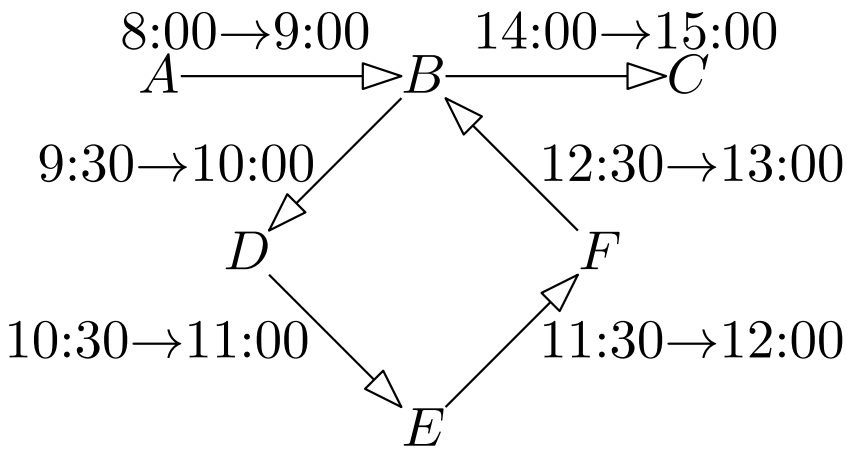
Die Eingabe enthält auch:

- Eine minimale Abfahrtszeit d
- Eine maximale Ankunftszeit a
- Startstop s
- Zielstop t

Optimierung:

- Für jeden Stop x bestimme die früheste Ankunftszeit $\tau(x)$ von s mit Abfahrtszeit d Dies kann man mit dem grundlegenden Connection Scan, eingeschränkt auf das Subarray machen.
- Ehe man eine Connection c im Profilalgorithmus bearbeitet, testet man ob $\tau(c_{\text{depstop}}) \leq c_{\text{deptime}}$. Wenn dies nicht gilt, dann überspringt man die Connection, da sie nicht Teil einer Journey sein kann.

Probleme mit Frühester Ankunft



Der Fahrgast will von A nach C.

Früheste Ankunfts-Jounreys:

1. $A \to B \to C$
2. $A \to B \to D \to E \to F \to B \to C$

Der Fahrgast will Journey 1.

Aber auch Jounrey 2 ist optimal bezüglich Ankunftszeit. →Eine Lösung:

Minimiere Anzahl von Transfers

Probleme mit Frühester Ankunft

Einfache Lösung: Wähle unter allen früheste Ankunfts-Jounreys die mit den wenigsten Transfers.

Änderungen am Algorithmus: Kleine Konstante auf die Ankunftszeit addieren, die von der $P(c_{\text{arrstop}})$ Evaluierung zurück gegeben wird.

Oft gut genug.

Problems with Earliest Arrival Time

Komplizierte Lösung: Berechne alle Pareto-optimalen Journeys.

Pareto-optimal: Eine Journey ist Pareto-optimal, wenn es keine andere Journey gibt, die sowohl schneller ist, als auch weniger Transfers benötigt.

Änderungen am Algorithmus:

- Wir beschränken uns auf Journeys mit n Zugeinstiegen (i.e. $n-1$ Transfers).
- Ersetze alle Ankunftszeiten mit $(a_0, a_1, a_2 \dots a_n)$-Tupeln. a_i gibt die früheste Ankunftszeit an, wenn man in höchsten i Züge einsteigt.
- Ein Paar einfügen entspricht dem Einsteigen in ein Zug. $\rightarrow$ Verschiebe alle Komponenten, d.h., $\mathrm{shift}(a_0, a_1, \dots, a_n) = (\infty, a_0, a_1, \dots, a_{n-1})$
- Minimum wird durch Komponentenminimum ersetzt.

Pareto Profil Connection Scan

$P(s)$: profile of stop s;
$T(t)$: Früheste Ankunft von Trip t, ein Tupel;
c_{eval} : Evaluierungspaar ID, unabhängig vom Zielstop;

$P(s) \leftarrow \{(\infty, (\infty, \infty \ldots \infty))\}$ for every stop s;
$T(t) \leftarrow (\infty, \infty \ldots \infty)$ für jeden Trip t;
for *jede Connection c absteigend nach* c_{deptime} **do**
    $t \leftarrow P(c_{\text{arrstop}})[c_{\text{eval}}]$;
    $t \leftarrow \min(t, T(c_{\text{tripid}}))$;
    if $c_{\text{arrstop}} =$ *target stop* **then**
        $t \leftarrow \min(t, (c_{\text{arrtime}}, c_{\text{arrtime}} \ldots c_{\text{arrtime}}))$;
    $T(c_{\text{tripid}}) \leftarrow t$;
    Hänge $(c_{\text{deptime}}, \min(\text{shift}(t), P(c_{\text{depstop}})[\text{front}].\,\text{arrtime}))$
        an den Anfang von $P(c_{\text{depstop}})$;

SIMD/SSE

- SIMD : Single Instruction Multiple Data
- Spezielle CPU-Instruktion die mit Vektoren fester Länge arbeiten und nur einen CPU-Takt benötigen.

- Gibt es in allen modernen x86 Prozessoren.
- 128 Bit Registers à 4×32 Bit oder à 8×16 Bit Ganzzahlen.

- Die x86 Implementierung heißt SSE.
- SIMD ist das generelle Konzept.

- Bei neueren Prozessoren gibt auch AVX mit 256 Bit Registern.

SIMD/SSE

Ohne SSE:

```
int a[8],b[8],c[8];
c[0]=a[0]+b[0];   c[1]=a[1]+b[1];   c[2]=a[2]+b[2];
c[3]=a[3]+b[3];   c[4]=a[4]+b[4];   c[5]=a[5]+b[5];
c[6]=a[6]+b[6];   c[7]=a[7]+b[7];
```

8 Zeiteinheiten

Mit SSE:

```
__m128i a[2],b[2],c[2];
c[0]=_mm_add_epi32(a[0],b[0]);
c[1]=_mm_add_epi32(a[1],b[1]);
```

2 Zeiteinheiten

Aber: Beschleunigung von 4 gibt es nur bei compute-bound Algorithmen.

SIMD/SSE

Ohne SSE:

```
short a[8],b[8],c[8];
c[0]=a[0]+b[0];   c[1]=a[1]+b[1];   c[2]=a[2]+b[2];
c[3]=a[3]+b[3];   c[4]=a[4]+b[4];   c[5]=a[5]+b[5];
c[6]=a[6]+b[6];   c[7]=a[7]+b[7];
```

8 Zeiteinheiten

Mit SSE:

```
__m128i a,b,c;
c=_mm_add_epi16(a,b);
```

1 Zeiteinheit

Aber: Beschleunigung von 8 gibt es nur bei compute-bound Algorithmen.

SIMD/SSE

Einfache if/else-Konstrukte gibt es auch.

Ohne SSE:

```
short a[8],b[8],c[8];
for(int i=0; i<8; ++i)
    c[i] = a[i] < b[i] ? a[i] : b[i];
```

Mit SSE:

```
__m128i a,b,c;
c=_mm_blendv_epi8(a,b,_mm_cmplt_epi16(a,b));
```

Oder einfach die Minimumanweisung verwenden:

```
__m128i a,b,c;
c=_mm_min_epi16(a,b);
```

Pareto Profil Connection Scan

Die Operationen auf den Ankunftszeittupeln können mit SSE-Anweisungen gemacht werden.

Zeit Kompression

Problem: Algorithmus ist memory-bound.
Der Speicher ist fast vollständig mit Ankunftszeiten gefüllt.
$\rightarrow$ Komprimiere Ankunftszeiten.

Beobachtung: Nicht an jedem Zeitpunkt fährt ein Zug.

Idee: Berechne für jeden Stop ein geordnetes Array von Zeitpunkten $t_0, t_1, \ldots, t_n$ an denen ein Zug abfährt oder ankommt.

- Indizes respektieren die zeitliche Ordnung, d.h., $t_i < t_j \iff i < j$.
- Indizes passen oft in 16 Bit.
- Kopiere Indizes anstatt von Zeitpunkten.

Probleme mit Verspätungen

Karlsruhe Hbf	Fr, 31.05.13	dep	08:00	10:30	2	ICE, EC, ES	Connection i
Roma Termini	Fr, 31.05.13	arr	18:30				

Station/Stop	Date	Time		Platform	Products	
Karlsruhe Hbf	Fr, 31.05.13	dep	08:00	2	ICE 5	Intercity-Express
Zürich HB	Fr, 31.05.13	arr	11:00	10		Bordrestaurant
Transfer time 9 min.						→ Adjust the transfer time
Zürich HB	Fr, 31.05.13	dep	11:09	5	EC 17	Eurocity
Milano Centrale	Fr, 31.05.13	arr	14:50 approx. +25			Subject to compulsory reser available
Transfer time 20 min.		Connecting train may not be reached in time.				→ Adjust the transfer time
Milano Centrale	Fr, 31.05.13	dep	15:10		ES 9541	EuroStar Italia
Roma Termini	Fr, 31.05.13	arr	18:30			Subject to compulsory reser wheelchairs

composed out of several screenshots of bahn.de, specific situation was not observed

Probleme mit Verspätungen

Karlsruhe Hbf	Fr, 31.05.13	dep	08:00	10:30	2	ICE, EC, ES	Connection i
Roma Termini	Fr, 31.05.13	arr	18:30				

Station/Stop	Date	Time		Platform	Products	
Karlsruhe Hbf	Fr, 31.05.13	dep	08:00	2	ICE 5	Intercity-Express
Zürich HB	Fr, 31.05.13	arr	11:00	10		Bordrestaurant
Transfer time 9 min.						→ Adjust the transfer time
Zürich HB	Fr, 31.05.13	dep	11:09	5	EC 17	Eurocity
Milano Centrale	Fr, 31.05.13	arr	14:50 approx. +25			Subject to compulsory reser available
Transfer time 20 min.		Connecting train may not be reached in time.				→ Adjust the transfer time
Milano Centrale	Fr, 31.05.13	dep	15:10		ES 9541	EuroStar Italia
Roma Termini	Fr, 31.05.13	arr	18:30			Subject to compulsory reser wheelchairs

composed out of several screenshots of bahn.de, specific situation was not observed

25 min Verspätung vs 20 min Transfer

Probleme mit Verspätungen

Karlsruhe Hbf	Fr, 31.05.13	dep	08:00	10:30	2	ICE, EC, ES	Connection i
Roma Termini	Fr, 31.05.13	arr	18:30				

Station/Stop	Date	Time		Platform	Products	
Karlsruhe Hbf	Fr, 31.05.13	dep	08:00	2	ICE 5	Intercity-Express
Zürich HB	Fr, 31.05.13	arr	11:00	10		Bordrestaurant
Transfer time 9 min.						→ Adjust the transfer time
Zürich HB	Fr, 31.05.13	dep	11:09	5	EC 17	Eurocity
Milano Centrale	Fr, 31.05.13	arr	14:50 approx. +25			Subject to compulsory reser available
Transfer time 20 min.		Connecting train may not be reached in time.				→ Adjust the transfer time
Milano Centrale	Fr, 31.05.13	dep	15:10		ES 9541	EuroStar Italia
Roma Termini	Fr, 31.05.13	arr	18:30			Subject to compulsory reser wheelchairs

composed out of several screenshots of bahn.de, specific situation was not observed

Was wenn 9 min nicht reichen?

Probleme mit Verspätungen

Karlsruhe Hbf	Fr, 31.05.13	dep	08:00	10:30	2	ICE, EC, ES	Connection i
Roma Termini	Fr, 31.05.13	arr	18:30				

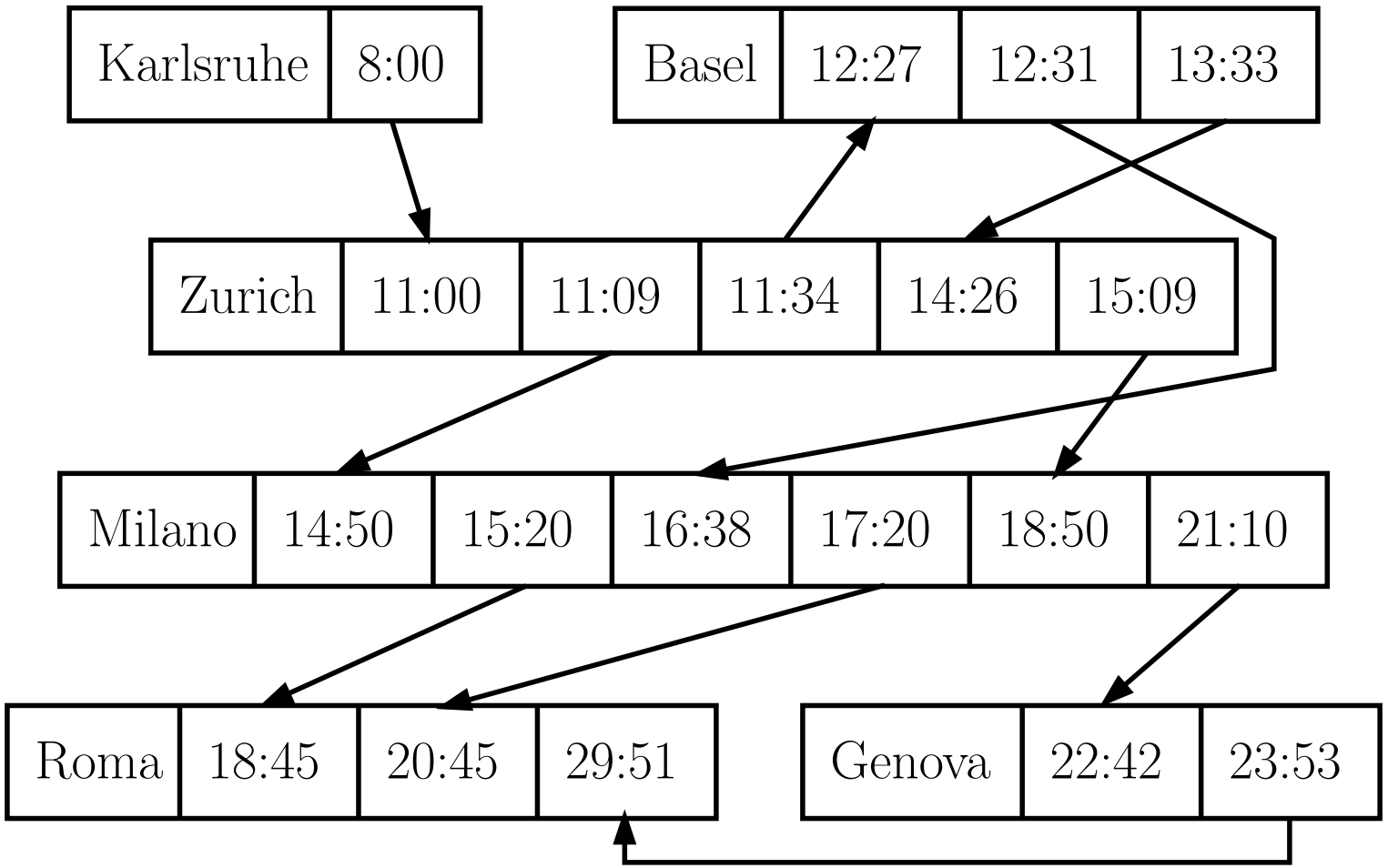
Station/Stop	Date	Time		Platform	Products	
Karlsruhe Hbf	Fr, 31.05.13	dep	08:00	2	ICE 5	Intercity-Express
Zürich HB	Fr, 31.05.13	arr	11:00	10		Bordrestaurant
Transfer time 9 min.						Adjust the transfer time
Zürich HB	Fr, 31.05.13	dep	11:09	5	EC 17	Eurocity
Milano Centrale	Fr, 31.05.13	arr	14:50 approx. +25			Subject to compulsory reser available
Transfer time 20 min.		Connecting train may not be reached in time.				Adjust the transfer time
Milano Centrale	Fr, 31.05.13	dep	15:10		ES 9541	EuroStar Italia
Roma Termini	Fr, 31.05.13	arr	18:30			Subject to compulsory reser wheelchairs

composed out of several screenshots of bahn.de, specific situation was not observed

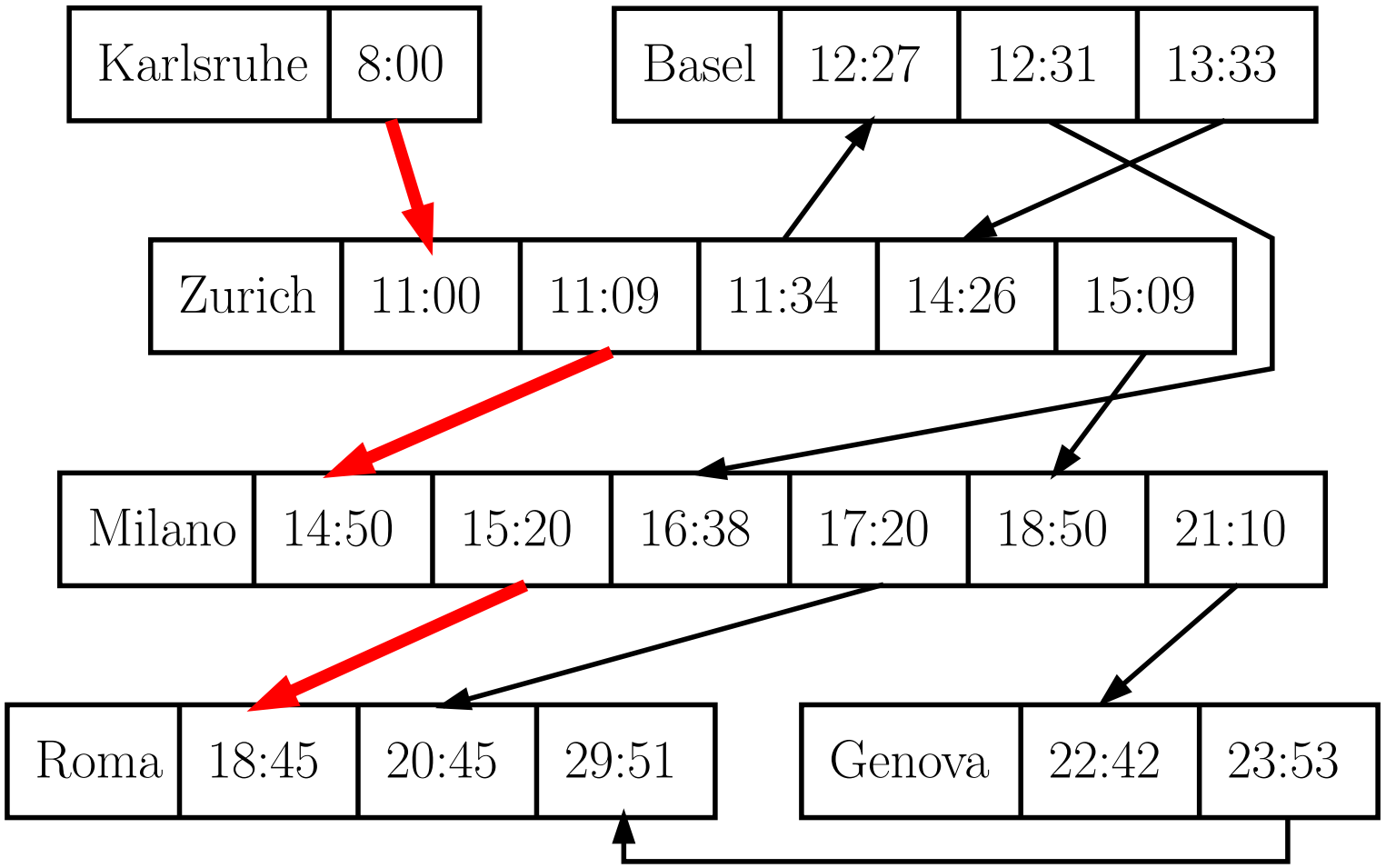
. . . aber eventuell reichen sie . . .

$\rightarrow$ Backup-Journeys sind notwendig

Entscheidungsgraph

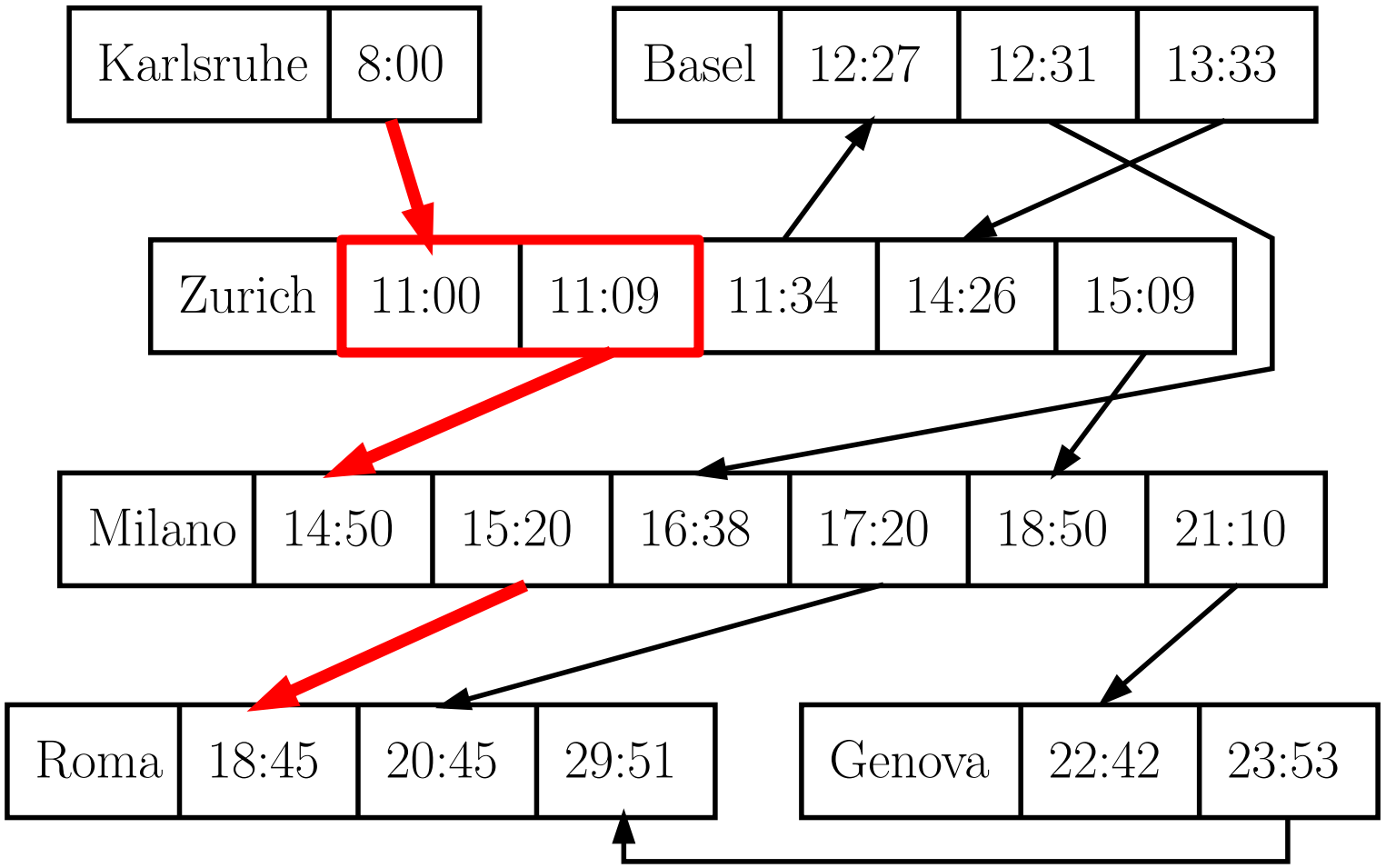


Entscheidungsgraph



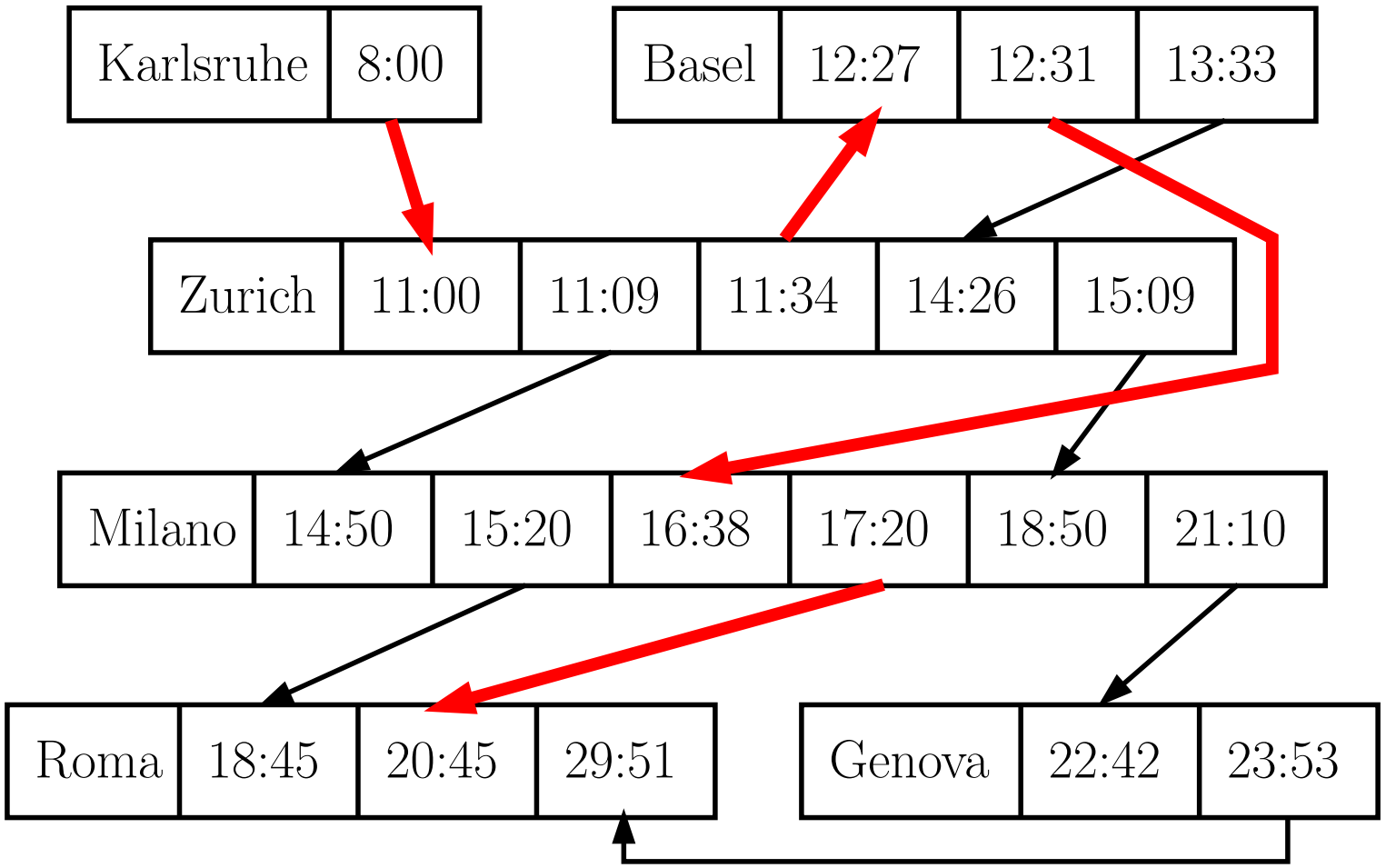
Wenn alles gut geht

Entscheidungsgraph



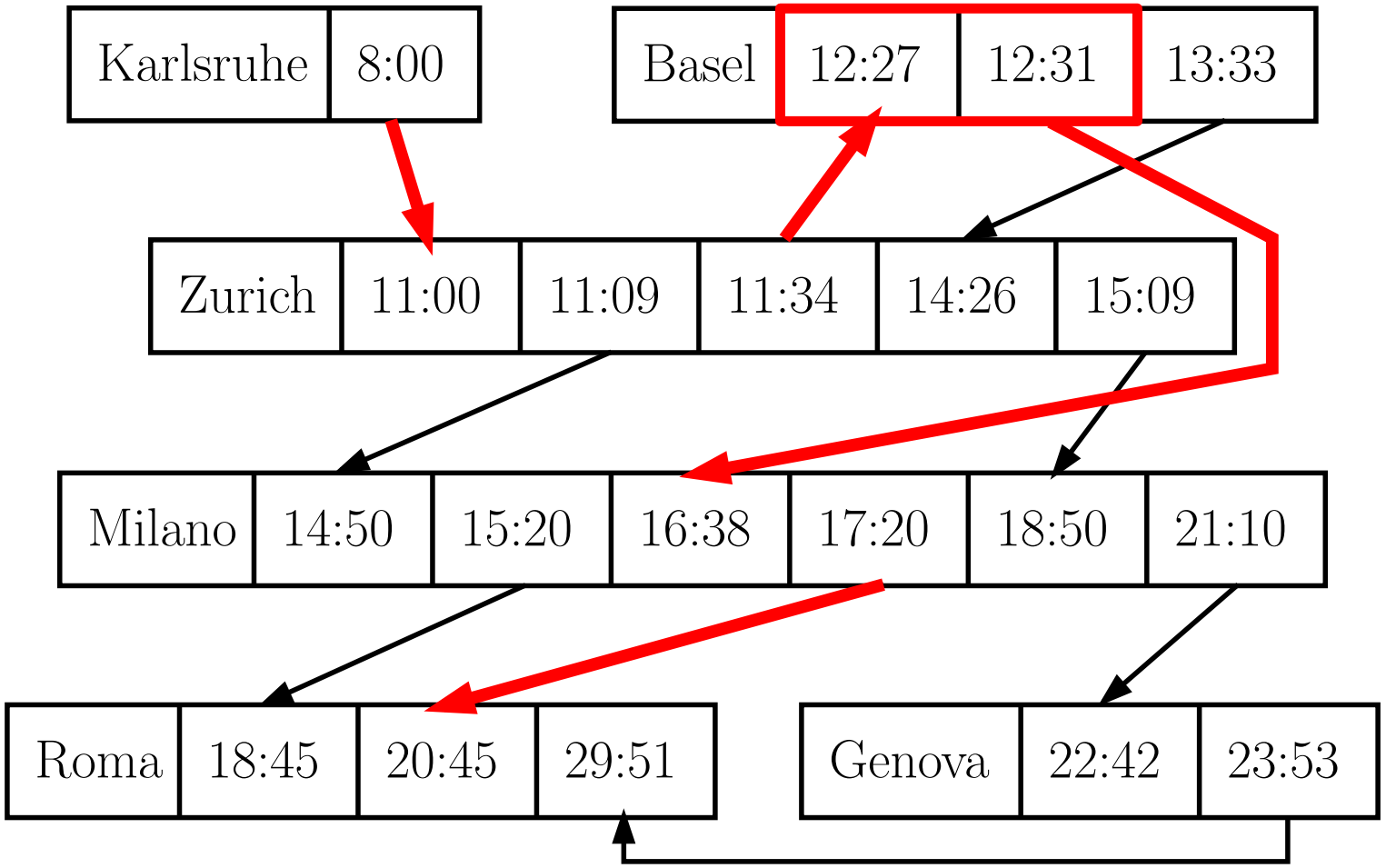
9 min Umstiegszeit $\rightarrow$ Geht oft schief

Entscheidungsgraph



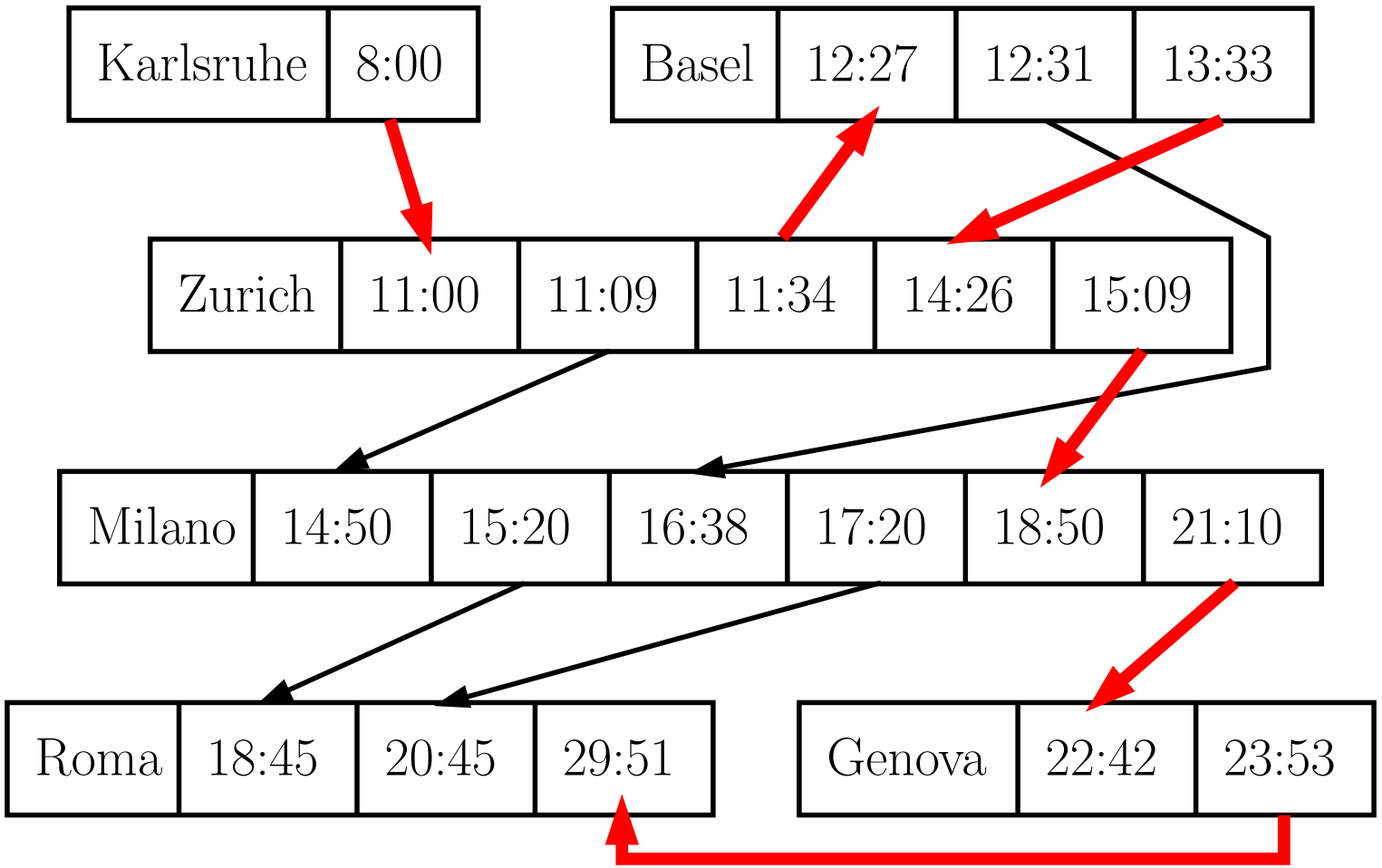
Der Backup

Entscheidungsgraph



Unsicher Umstieg auf dem Backup $\rightarrow$ Der Backup braucht ein Backup

Entscheidungsgraph

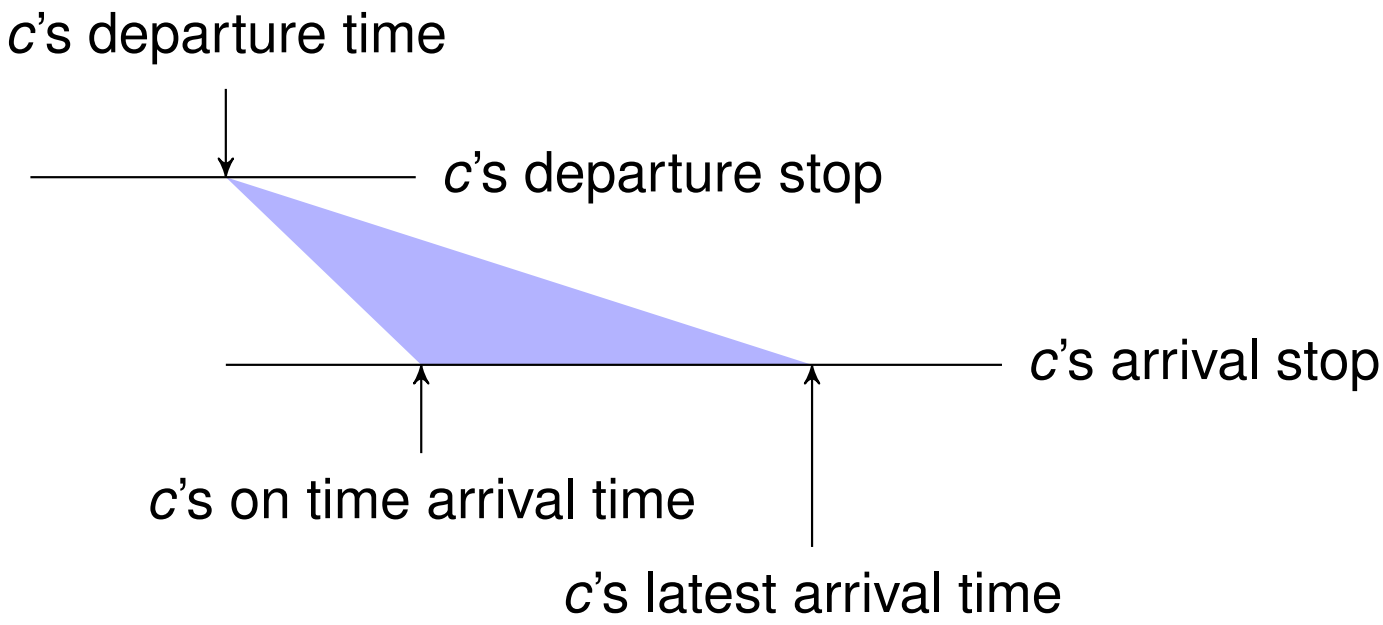


Backup des Backups

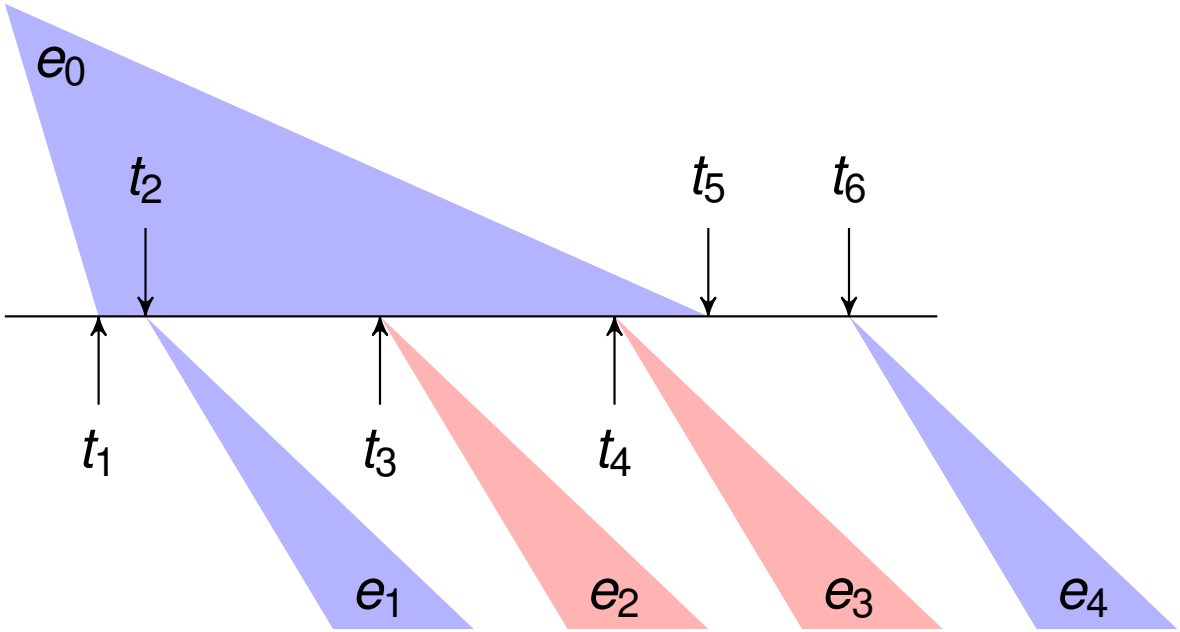
Verpätungsmodel

Annahmen:

- Connections kommen mit einer zufälligen Verspätung an
- Connections haben eine maximale Verspätung
- Zufallsverteilungen sind bekannt
- Alle Zufallsvariablen sind unabhängig
- Connections fahren immer pünktlich ab

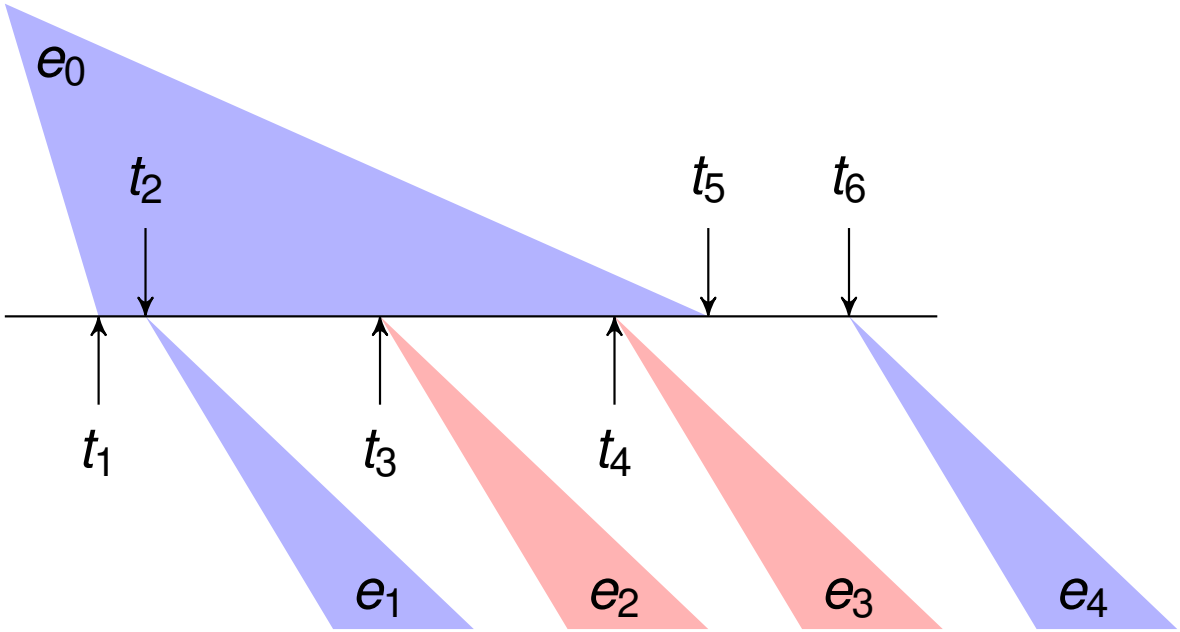


Erwartete Ankunftszeit



$e_0 \ldots e_4$:	erwartete Ankunftszeit
$t_1 \ldots t_6$:	feste Zeitpunkte
blau:	im Entscheidungsgraph
rot:	nicht im Entscheidungsgraph

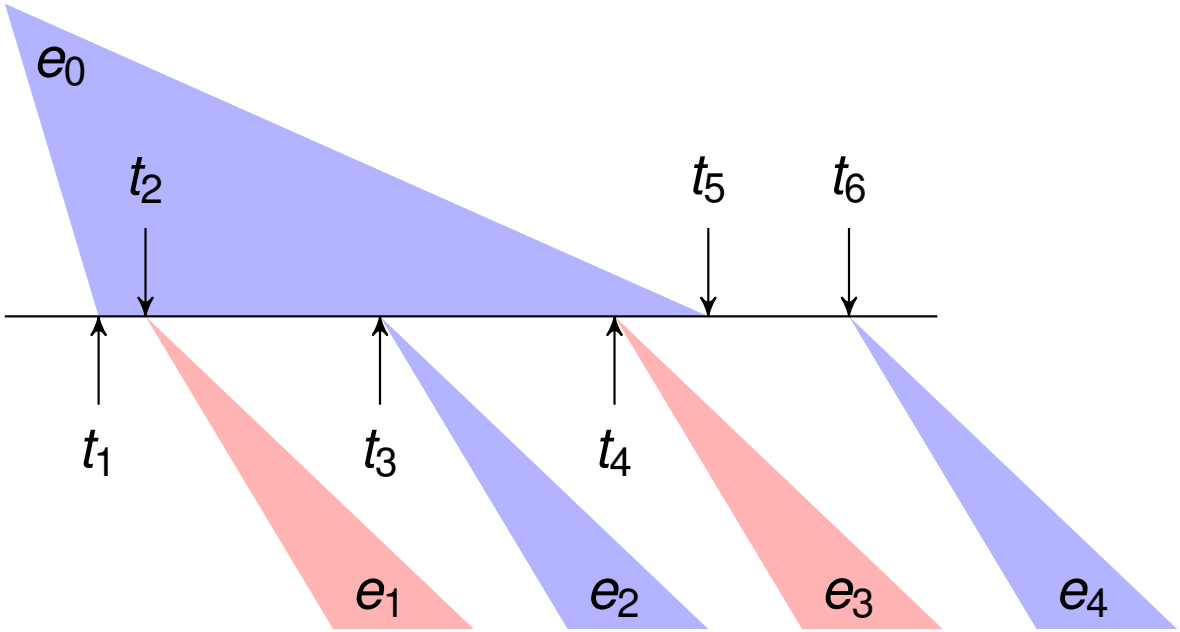
Erwartete Ankunftszeit



t: tatsächliche Ankunftszeit

$$e_0 = P(t_1 \leq t \leq t_2) \cdot e_1 + P(t_2 \leq t \leq t_5) \cdot e_4$$

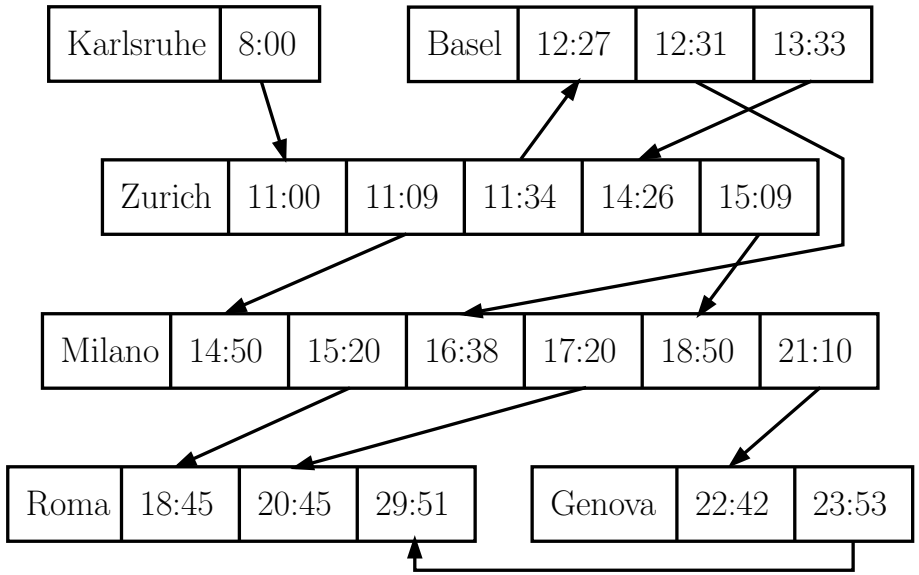
Erwartete Ankunftszeit



t: tatsächliche Ankunftszeit

$$e_0 = P(t_1 \leq t \leq t_3) \cdot e_2 + P(t_3 \leq t \leq t_5) \cdot e_4$$

Minimale Erwarte Ankunftszeit
Minimum Expected Arrival Time (MEAT)

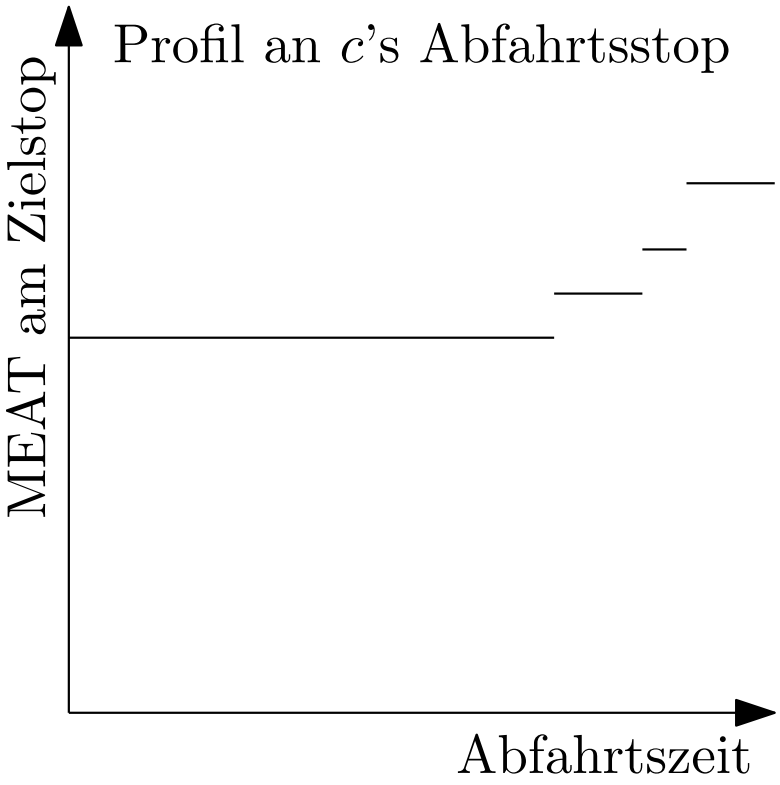


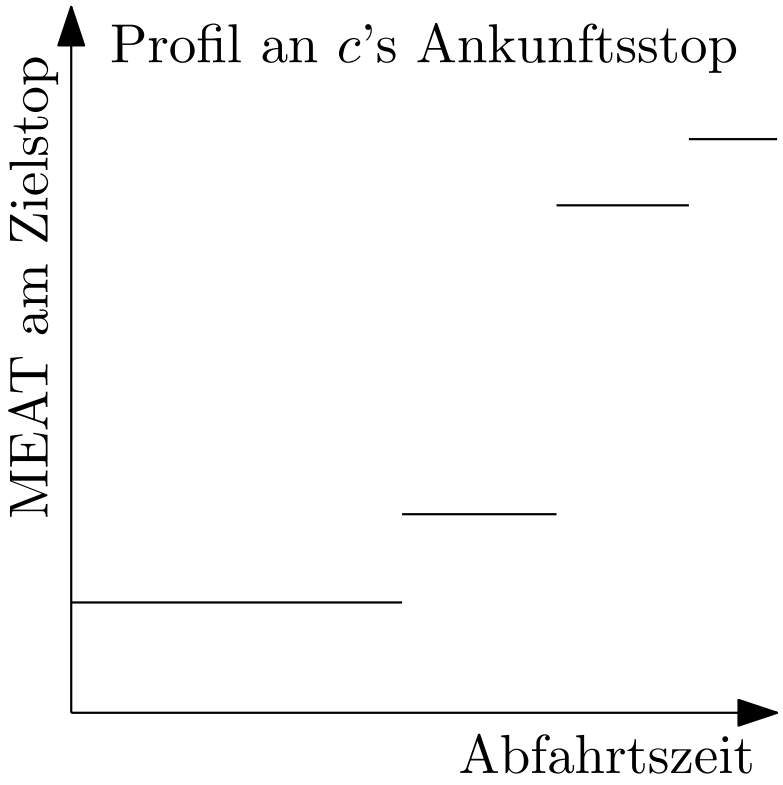
Entscheidungsgraphen berechnen

Eingabe: Fahrplan, Verspätungswahrscheinlichkeiten, Zielstop

Ausgabe: Connectionteilmenge mit *minimaler erwarteter Ankunftszeit* für jeden Startstop und Startzeit

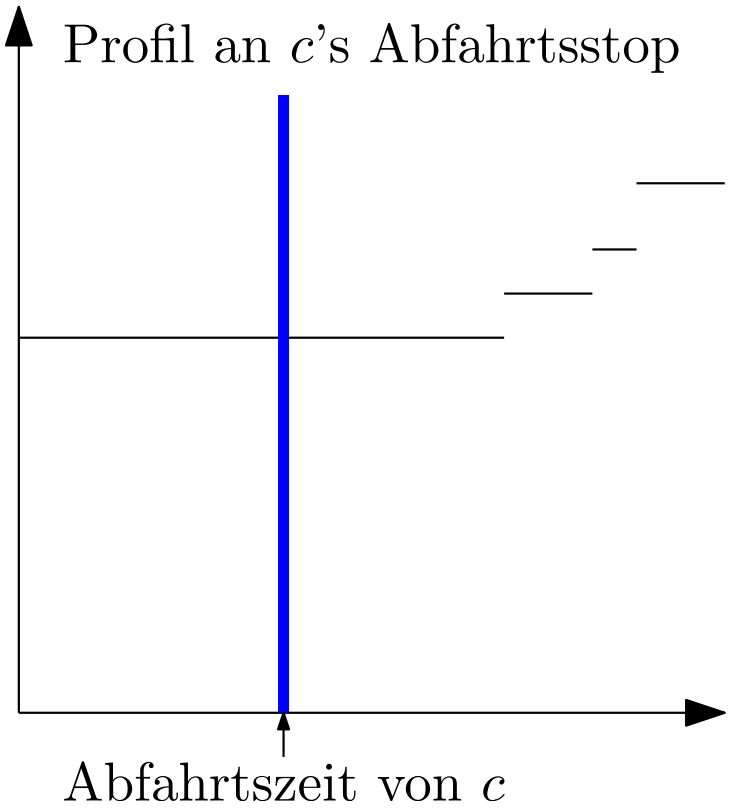
MEAT Connection Scan

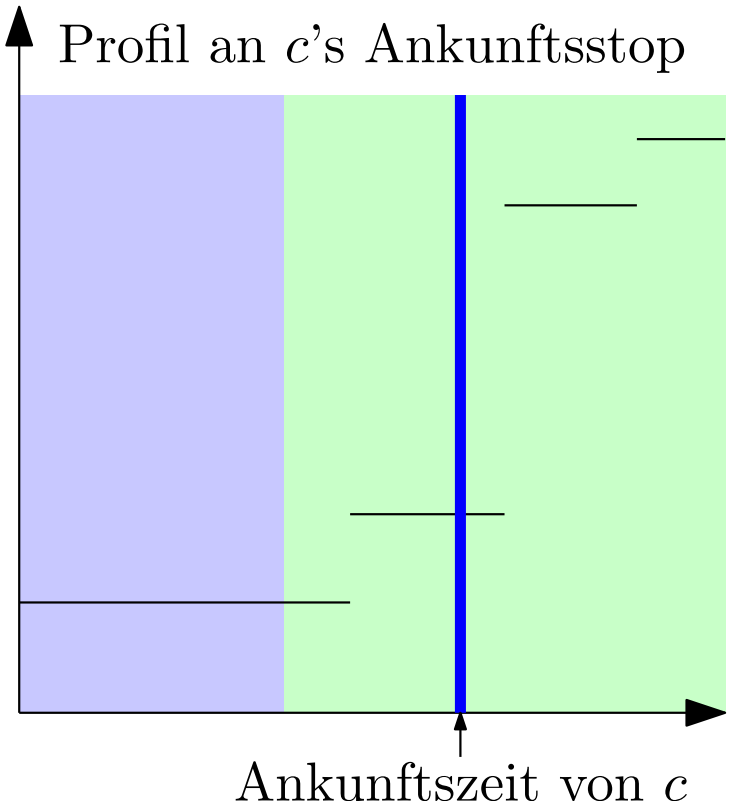




Profil bilden Abfahrtszeit auf MEAT am Zielstop ab.

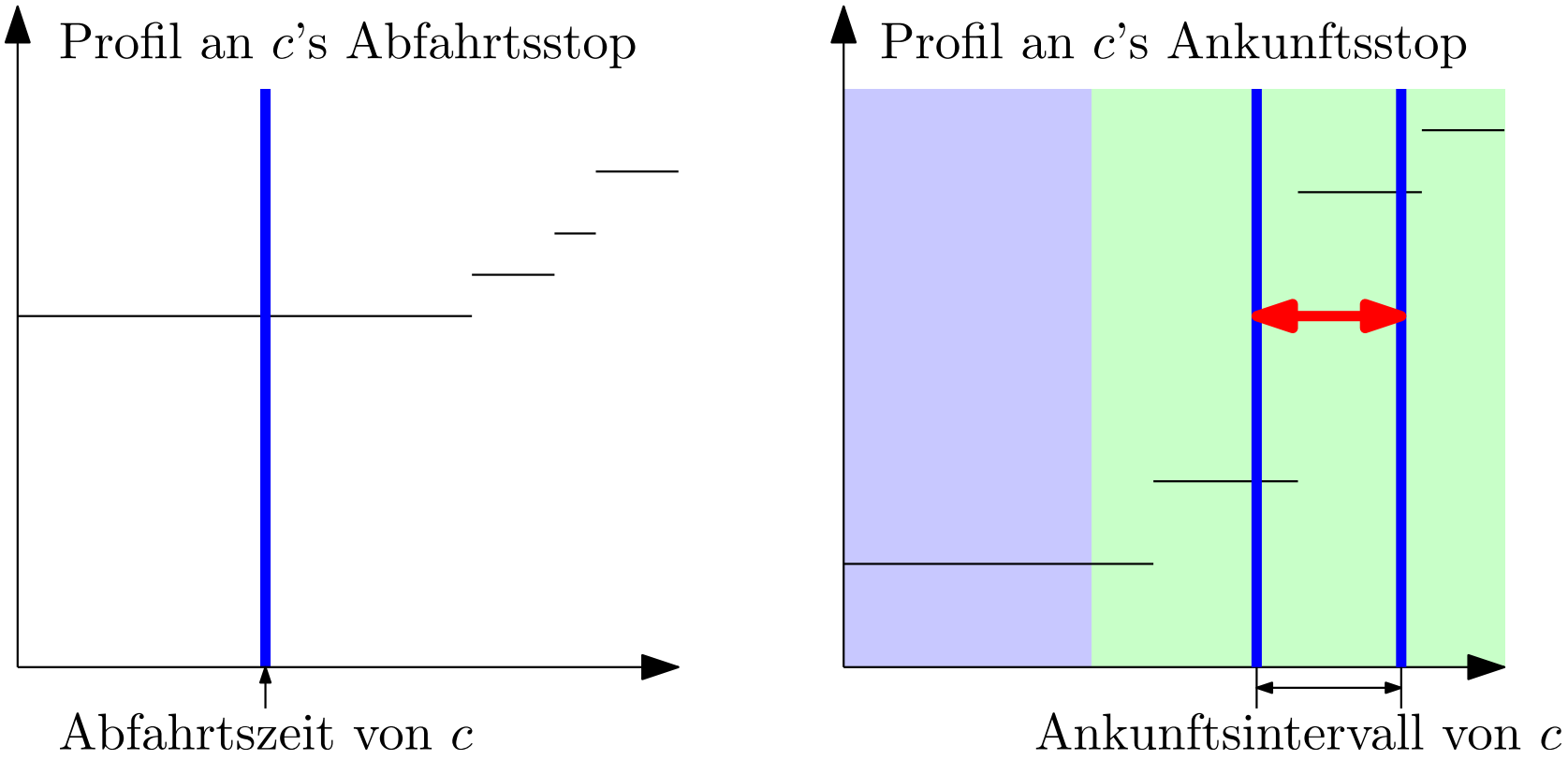
MEAT Connection Scan





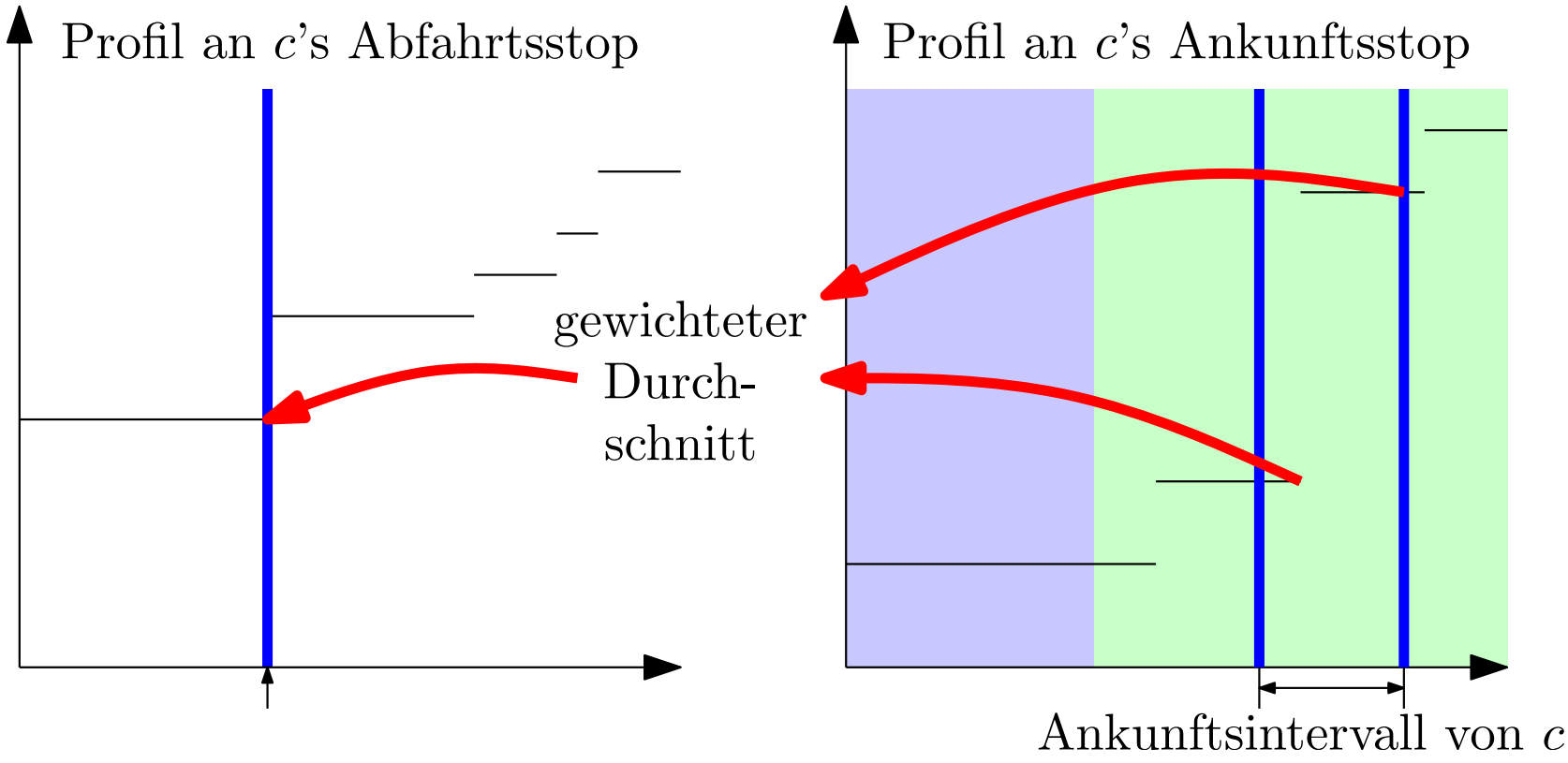
Profil bilden Abfahrtszeit auf MEAT am Zielstop ab.

MEAT Connection Scan



Etwas weiter scannen um alle relevanten Züge einzusammeln.
Das geht nicht in $O(1)$.

MEAT Connection Scan



Bilde das gewichtete Mittel mit der Wahrscheinlichkeit, dass man einen Zug erreichen kann. Einfügen ist $O(1)$ wie bisher.

Laufzeiten

London Instanz mit 4 850 431 Connections.

Früheste Ankunft One-to-One:

- Time-Expanded: 64.4 ms
- Time-Dependent: 10.9 ms
- Connection Scan: 2.0 ms

Früheste Ankunft One-to-All:

- Time-Expanded: 876.2 ms
- Time-Dependent: 18.9 ms
- Connection Scan: 9.7 ms

(Time-Dependent kriegt man etwas schneller, mit Ideeen die nicht im Kurs vorkommen.)

Laufzeiten

Non-Pareto Profil All-to-One:

- Self-Pruning-Connection-Setting : 1 262 ms
- Connection Scan: 177 ms
- + constant eval: 134 ms
- + time compress: 104 ms

Pareto Profil All-to-One (mit höchstens 8 Zügen pro Journey):

- RAPTOR : 1 179 ms
- Connection Scan: 255 ms
- + SSE: 221 ms

MEAT: 272 ms