

## Aufgabe 2 – Reiseplanung

### Aufgabenstellung

#### Teilaufgabe 1

Schreibe ein Programm, das als Eingabe den Fahrtwunsch eines Fahrgastes in der Form [Startort, Zielort, Startzeit] verlangt und die zeitlich minimale Verbindung mit Umstiegsorten und Umstiegszeiten berechnet und ausgibt. Bei zwei zeitlich gleichen Verbindungen ist die kostengünstigere zu wählen.

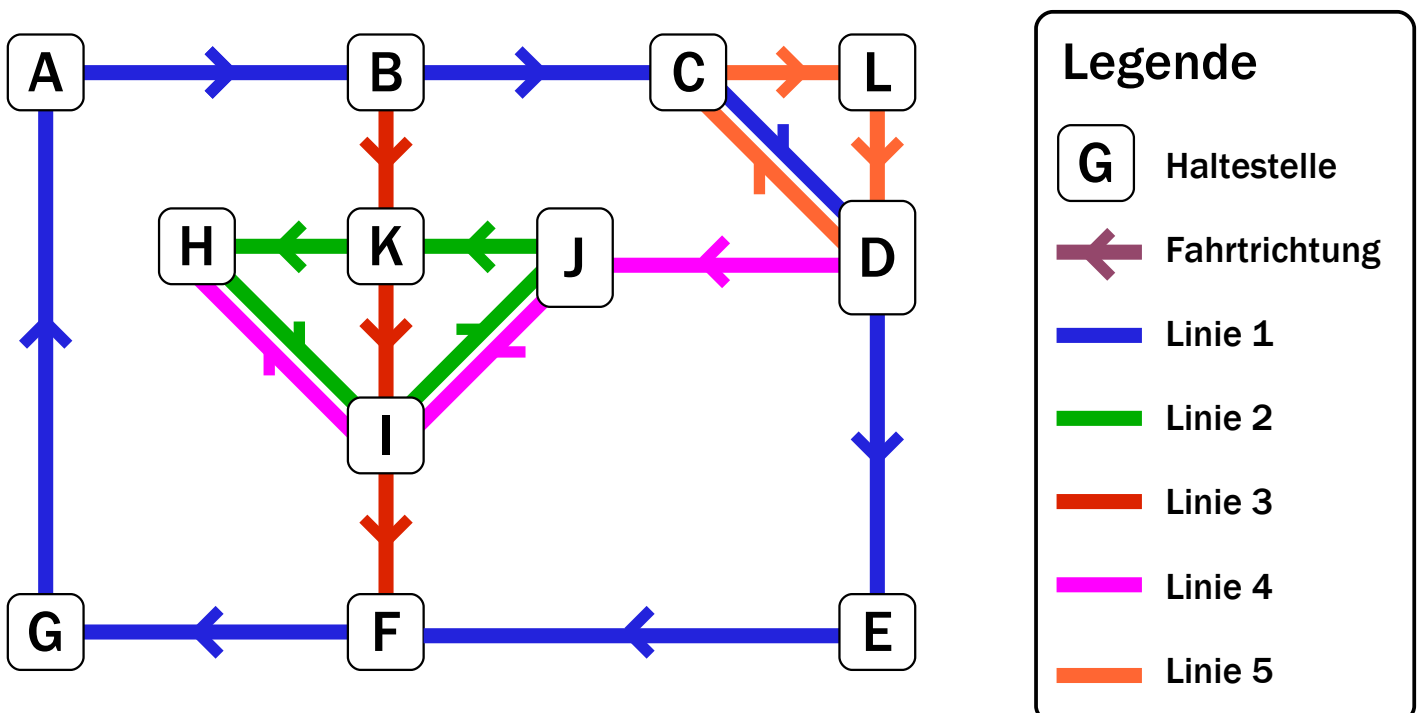
#### Teilaufgabe 2

Erweitere Dein Programm und berechne für den Fahrgast auch immer die kostengünstigste Verbindung und gib diese ebenfalls aus. Protokolliere die Eingabe und Ausgabe von mindestens 5 Strecken.

### Anmerkungen

Teilaufgaben 1 und 2 wurden von mir nicht einzeln behandelt, sondern fallen in einem Skript zusammen, da zwischen beiden Aufgaben kein großer Unterschied besteht.

Mit Hilfe der gegebenen Fahrplandaten habe ich mir erstmal einen Liniennetzplan von Neckarmonien erstellt, um einen Überblick zu bekommen:



**Lösungsidee**

Für den Weg zum Ziel sind grob unterteilt drei Schritte notwendig:

- A) Der erste Schritt, um die Verbindungen vom gewünschten Startort zum gewünschten Zielort zu ermitteln, besteht darin, den Weg zu finden. Das heißt also herauszufinden über welche Haltestellen man mit welchen Buslinien fahren kann, um letztendlich am Ziel anzukommen.

Das Verfahren, um die Wege zu finden lautet dabei folgendermaßen:

Man beginnt bei der Starthaltestelle und ermittelt alle Linien, die diese Haltestelle auf ihrer Strecke abfahren und nach dieser Haltestelle noch mindestens eine weitere anfahren (die Starthaltestelle sollte also nicht die Endstation einer Buslinie sein).

Für jede dieser ermittelten Linien fährt man jetzt bis zur nächsten Haltestelle. Von hier aus prüft man wieder, mit welchen Linien man weiterfahren kann und fährt dann wieder mit den ermittelten Linien zur nächsten Haltestelle. (Rekursion)

Zu prüfen ist an jeder Haltestelle, ob man schon am Ziel angekommen ist. Wenn ja, ist ein Weg zum Ziel gefunden. Dieser Weg kann gespeichert werden.

Des Weiteren ist es wichtig zu prüfen, ob man an dieser Haltestelle schon einmal war. Alleine schon dafür ist eine Zwischenspeicherung des bisherigen Weges nötig. Wenn das der Fall ist, sollte ein Abbruch folgen, denn schließlich wurden von dieser Haltestelle aus auf bestimmten Wegen schon mal die weiteren Anschlüsse überprüft und man würde dann gegebenenfalls unendlich oft zurück an diese Haltestelle kommen. Die Folge wäre wohl unglücklicherweise eine Endlosschleife.

- B) Nachdem die Wege ermittelt wurden, die zum Ziel führen, folgt der zweite Schritt: jetzt müssen noch für jeden Weg (im folgenden Verbindung) die entsprechenden Uhrzeiten von Ein- und Ausstiegen und in Verbindung damit die Haltestellen der Umstiege berechnet werden.

Die gewünschte Abfahrtszeit wird vom Kunden am Client-Rechner festgelegt. Das Skript ermittelt die Abfahrtszeit des Buses von der Starthaltestelle.

Liegt die gewünschte Abfahrtszeit zeitlich vor der Abfahrtszeit des ersten Buses der Linie mit der gefahren werden soll, so wird als Abfahrtszeit letztere gewählt.

Liegt die gewünschte Abfahrtszeit zeitlich nach der Abfahrtszeit des letzten Buses der Linie mit der gefahren werden soll, so wird als Abfahrtszeit die Abfahrtszeit des ersten Buses gleicher Linie am Folgetag gewählt. Der Client müsste also ggf. dort übernachten oder er wählt später dann eine frühere Verbindung.

Liegt die gewünschte Abfahrtszeit zwischen diesen beiden Grenzen, wird die zeitlich nächste Abfahrt des Busses der gewünschten Linie ermittelt.

Ist die Abfahrtszeit ermittelt, folgt als nächstes die Berechnung der Ankunftszeit an der nächsten Umstiegshaltestelle oder der Zielhaltestelle. Eine Umstiegshaltestelle wird daran erkannt, dass entweder die Linie mit der man zur aktuellen Haltestelle gefahren ist von der

unterscheidet, mit der man weiterfahren möchte und/oder darin, dass es sich um die Starthaltestelle der Linie handelt mit der man weiterfahren möchte.

Die Ankunftszeit berechnet sich dann aus der Zeit die man von der letzten Haltestelle bei der man eingestiegen ist mit der Linie gefahren ist zuzüglich der Abfahrtszeit. Die Zeit, die man unterwegs gewesen ist (kurz Fahrtzeit) ermittelt sich aus der Zeit, die man für die einzelnen Strecken zwischen den Haltestellen gefahren ist und der einminütigen Wartezeit an den Haltestellen die auf der Wegstrecke liegen.

An der nächsten Haltestelle an der man aussteigt, beginnt das Spiel der Berechnung der Abfahrts- und Ankunftszeiten von Neuem, außerdem man ist am Ziel angelangt. Die gewünschte Abfahrtszeit entspricht der Ankunftszeit, da ein Neckarmonier ja unendlich schnell umsteigen kann.

- C) Sind alle Daten der Verbindungen ermittelt (Weg und Uhrzeiten) geht es darum, die geforderte "zeitlich minimale" Verbindung und die kostengünstigste Verbindung zu finden.

Erst ein mal zur Definition der zeitlich minimalen Verbindung: das ist bei mir die Verbindung, deren Ankunftszeit an der Zielhaltestelle am frühesten ist (im Folgenden auch als schnellste Verbindung bezeichnet).

Für beide Fälle muss letztlich nur eine aufsteigende Sortierung nach Ankunftszeit oder Preis (beziehungsweise Einstiegen, da diese ja proportional zueinander sind) erfolgen.

Gibt es keine eindeutige schnellste Verbindung, werden die schnellsten zeitgleichen Verbindungen noch auf ihren Preis hin unterschieden und die günstigste unter den schnellsten ausgegeben. Gibt es unter den schnellsten zeitgleichen Verbindungen auch keine eindeutige günstigste Verbindung, werden bei mir alle zugleich schnellsten und preisgleichen Verbindungen ausgegeben. Es ist prinzipiell egal, welche Verbindung gewählt wird. Das ist allerdings ein sehr unwahrscheinlicher Fall.

Gibt es keine eindeutige günstigste Verbindung, wird unter den günstigsten preisgleichen die schnellste unter diesen gesucht und ausgegeben. Gibt es unter den gleichgünstigen keine eindeutig schnellste, so werden alle zugleich günstigsten und schnellsten Verbindungen ausgegeben. Auch hier ist dann egal, welche Verbindung gewählt wird (vgl. oben).

***Programm-Dokumentation***

Bei der Umsetzung habe ich mich für PHP entschieden. Das liegt daran, dass es eine meiner Meinung nach recht schöne, vielseitige Skriptsprache ist, die ich recht gut beherrsche. Die einfache Möglichkeit einer Datenbankanbindung an Systeme wie beispielsweise MySQL macht dynamische Systeme sehr schnell und leicht realisierbar. Das Skript ließe sich so erweitern, dass Daten aus einer Datenbank entnommen werden. Im Moment befinden sich die gegebenen Fahrplandaten noch direkt im Skript in einer Konfigurationsdatei.

Das entwickelte Skript ließe sich auch als Webanwendung des Verkehrsverbundes in Neckarregion anbieten bei welchem Kunden online nachschauen könnten, welcher Bus sie als nächstes zum gewünschten Ziel bringt. Denkbar wäre aber auch die Anwendung an Fahrkartenautomaten mit einer Fahrplanauskunft, wenn diese das Hypertext Transfer Protocol unterstützen.

Das Skript wurde lokal auf meinem Rechner entwickelt und getestet. Es handelt sich dabei um einen Apache-Webserver mit PHP Version 5.1.4 unter Microsoft Windows XP.

### Datenstruktur

Die ersten Überlegungen für die Umsetzung der Lösungsidee gingen in die Richtung, eine vernünftige, brauchbare und leicht weiter zu verarbeitende Datenstruktur für die gegebenen Fahrplandaten zu finden. Ich entschied mich dabei für ein mehrdimensionales Array `$linien` mit folgendem Aufbau und Inhalt:

```
$linien = array(
  1 => array(
    'first_h' => 6,
    'first_m' => 0,
    'last_h'  => 23,
    'last_m'  => 30,
    'intervall' => 10,
    'start'   => 'A',
    'stops'   => array(
      'B' => 8,
      'C' => 12,
      'D' => 20,
      'E' => 10,
      'F' => 30,
      'G' => 8,
      'A' => 10
    )
  ),
  2 => array(
    'first_h' => 6,
    'first_m' => 30,
    'last_h'  => 23,
    'last_m'  => 30,
    'intervall' => 10,
    'start'   => 'J',
    'stops'   => array(
      'K' => 14,
      ...
    )
  ),
  ...
)
```

Die **Schlüssel in der ersten Dimension** stellen den jeweiligen Liniennamen (beziehungsweise die Nummer) der Buslinie dar. Es wäre also durchaus auch eine Bezeichnung wie 'N3' oder ähnlich möglich, um beispielsweise noch einen möglichen Nachtbus der Linie 3 hinzuzufügen.

Die Elemente beinhalten verschiedene Elemente, die die Linie näher beschreiben. Die **Schlüssel der zweiten Dimension** erklären eigentlich die Form und den Zweck der Werte:

<code>first_h</code>	Stunden der ersten Fahrt
<code>first_m</code>	Minuten der ersten Fahrt
<code>last_h</code>	Stunden der letzten Fahrt
<code>last_m</code>	Minuten der letzten Fahrt
<code>intervall</code>	Intervall
<code>startort</code>	Startort

Das Element mit dem Schlüssel `stops` ist wiederum ein Array. Es beinhaltet die einzelnen Haltestellen (ab Startort) als Schlüssel und die zugehörigen Fahrtzeiten, um von der vorigen Station zu dieser Haltestelle zu gelangen.

`$linien[1]['stops']['D']` beispielsweise beinhaltet also die Fahrzeit, die nötig ist, um mit Linie 1 von C nach D zu kommen (20 Minuten).

Die Haltestellennamen eignen sich als eindeutige Schlüssel, da eine Linie keine Haltestelle zwei mal anfährt, abgesehen von Rundkursen wie bei Linien 1, 2 und 5 bei denen allerdings innerhalb des Arrays `stops` auch keine Haltestelle doppelt auftaucht, da bereits ein Eintrag in `startort` vorhanden ist (deshalb auch Ausgliederung des Startortes aus `stops`).

Das gesamte, mehrdimensionale Array `$linien` ist in der Datei `data.inc.php` gespeichert und kann von dort aus eingebunden werden.

### *Anmerkung*

`$linien` musste in verschiedenste Funktionen mit übergeben werden und konnte nicht global eingebunden werden, da PHP sonst bei verschachtelten `foreach`-Schleifen und Funktionsaufrufen abgebrochen hat. Das ist leider nicht die optimale Lösung konnte aber nur durch diesen Workaround umgangen werden.

### *Eingabe*

Die Eingabe des Fahrtwunsches erfolgt über Aufruf des Formulars in der Datei `form.php`.

Startort

Zielort

Startzeit  :

Hier kann der User (Kunde) in je einer select-Box Start- und Zielort angeben. Diese Orte wurden vorher dynamisch aus dem via `data.inc.php` eingebundenen Array `$linien` erzeugt. In zwei weiteren select-Boxen kann die gewünschte Startzeit eingestellt werden. Die Voreinstellung der Startzeit wird serverseitig mit der Uhrzeit des Aufrufes bereits voreingestellt. Durch Drücken des Absende-Buttons erfolgt die Übergabe der Daten via Methode `HTTP-POST-Request` an die Datei `calc.php` auf dem Server.

### Verarbeitung und Ausgabe

Serverseitig könnte nun eine Prüfung der Daten auf Richtigkeit folgen. Das habe ich allerdings im Rahmen des Wettbewerbs weggelassen, da es für die korrekte und vollständige Lösung meiner Meinung nach nicht relevant ist.

Erst ein mal sollten wir uns jetzt die rekursiv aufgerufene Funktion `finde_weg` näher anschauen. Sie ist das Herzstück des in der Lösungs idee genannten **Schrittes 1** und befasst sich mit dem Finden möglicher Wege vom Start- zum Zielort.

```
finde_weg($linien, $ziel, $history, $linien_history, $r)
```

Die Parameter im Einzelnen:

<code>\$linien</code>	ist uns bereits bekannt (enthält Fahrplandaten)
<code>\$ziel</code>	Zielort
<code>\$history</code>	[Array] Speicher der auf dem Weg bisher abgefahrenen Haltestellen
<code>\$linien_history</code>	[Array] Speicher der Linien mit welchen das geschehen ist
<code>\$r</code>	Rekursionstiefe (nur für eine vernünftige, erweiterte Dokumentationsausgabe nötig)

Der erste Aufruf erfolgt über

```
finde_weg($linien, $ziel, array($start), array(), 0);
```

Die History enthält die Starthaltestelle, die Linien-History bleibt leer, schließlich hat ja noch keine Fahrt statt gefunden, und die Rekursionstiefe ist 0.

Die Funktion prüft dann an jeder Haltestelle, mit welchen Linien von dort aus weitergefahren werden kann. Dazu durchläuft sie in einer Schleife alle Linien und ermittelt mit der Funktion `fahre_mit_linie`, ob eine Weiterfahrt möglich ist.

Der Aufruf erfolgt mit folgenden Parametern:

```
fahre_mit_linie($linien, $linie, $stop, $position);
```

<code>\$linien</code>	ist uns bereits bekannt (enthält Fahrplandaten)
<code>\$linie</code>	Linie mit der gefahren werden soll
<code>\$stop</code>	Haltestelle von der gefahren werden soll
<code>\$position</code>	hier wird die Position der Haltestelle von der gefahren soll innerhalb der Strecke der Linien mit der gefahren werden soll festgehalten, um mögliche folgende Haltestellen leichter zu ermitteln, falls möglich (call by reference)

`fahre_mit_linie` überprüft erst, ob die Haltestelle der Starthaltestelle der gewünschten Linie entspricht. Ist dies der Fall, wird `true` zurück gegeben und die Position auf `-1` gestellt.

Ist das nicht der Fall, so wird geprüft, ob die Haltestelle überhaupt von der Linie angefahren wird. Ist das nicht der Fall, wird `false` zurück gegeben. Ansonsten wird geprüft, ob die Haltestelle nicht der letzten Haltestelle auf der Strecke der gegebenen Linie ist – eine Weiterfahrt wäre hier ja nicht möglich – und gibt dann entweder `true` (Weiterfahrt möglich) zurück und stellt die Position auf oder `false` (Haltestelle ist letzte auf der Strecke;

Weiterfahren nicht möglich) zurück.

Die ermittelten Wege werden im global eingebundenen, mehrdimensionalen Array `$verbindungen` gespeichert. Die enthaltenen Informationen bestehen nur aus den abgefahrenen Haltestellen und die Linien mit denen zwischen den Haltestellen gefahren wurde.

Das könnte dann in etwa so aussehen:

```
Array
(
    [0] => Array
        (
            [haltstellen] => Array
                (
                    [0] => A
                    [1] => B
                    [2] => C
                    [3] => D
                    [4] => J
                    [5] => K
                    [6] => H
                    [7] => I
                )
            [linien] => Array
                (
                    [0] => 1
                    [1] => 1
                    [2] => 1
                    [3] => 4
                    [4] => 2
                    [5] => 2
                    [6] => 2
                )
        )
    [1] => Array
        (
            [haltstellen] => Array
                (
                    [0] => A
                    [1] => B
                    [2] => C
                    [3] => D
                    [4] => J
                    [5] => K
                    [6] => I
                )
            [linien] => Array
                (
                    [0] => 1
                    [1] => 1
                    [2] => 1
                    [3] => 4
                    [4] => 2
                    [5] => 3
                )
        )
)
```

... )

Der **Schlüssel der ersten Dimension** bezeichnet die ID der gefundenen Verbindung. Der Wert stellt wiederum ein Array aus einem Array dar, welches die Haltestellen (Schlüssel "**haltstellen**") und einem Array (Schlüssel "**linien**"), welches die Liniennamen speichert.

Die genauere Funktionsweise entnehmen Sie bitte dem hoffentlich ausreichend kommentierten Quellcode.

Der erste Schritt ist damit abgeschlossen.

Der **zweite Schritt** beginnt mit einem Durchlauf der einzelnen Verbindungen, um Abfahrts-/Ankunftszeiten an den einzelnen Haltestellen und die Gesamtkosten für die Fahrt zu berechnen.

Das Skript arbeitet im Zeitformat der – von mir so benannten – “Tagesminuten”. Das heißt in diesem Integer ist die Minute des Tages gespeichert. Ist es beispielsweise 23:42 Uhr nachts, so entspricht das der Tagesminute  $23 \cdot 60 + 42 = 1422$ . Dieses Format hilft später in Schritt 3, die Uhrzeiten besser sortieren zu können. Die Funktion `mat` wiederum nimmt Tagesminuten entgegen und gibt sie als “gewöhnliche” Uhrzeiten im Format `HH:mm` zurück.

Für jede Haltestelle der Verbindung wird eine weitere Schleife durchlaufen: das Skript berechnet die Abfahrtszeit an der derzeitigen Haltestelle und die Ankunftszeit an der nächsten. Die Berechnung der Abfahrtszeit geschieht folgendermaßen: es wird erst einmal die erste Abfahrtszeit an der gewünschten Haltestelle ermittelt. Dazu wird die Uhrzeit der ersten Fahrt an der Starthaltestelle der jeweiligen Linie genommen und die Fahrtzeiten (inklusive der einminütigen Wartezeit an den einzelnen Haltestellen) addiert. Mit Hilfe des Intervalles, welches angibt, wie oft ein Bus vorbeikommt, wird die der gewünschten Abfahrtszeit nächstmögliche reale Abfahrtszeit ermittelt.

Das Skript prüft an jeder Haltestelle, ob es ein Umstiegsort ist (Bedingungen siehe Lösungsidee / Schritt 3) und speichert die Uhrzeiten für die Ausgabe zwischen. Bei einem Umstiegsort wird eine Variable inkrementiert, die die Anzahl der Umstiege speichert.

Am Ende der Schleife wird die Höhe der Kosten berechnet (direkt proportional zur Anzahl der Einstiege, die in `$verbindungen` gespeichert wurden) und in einem Array (`$verbindungen_preis`) für die Kosten mit dem Schlüssel der ID der jeweiligen Verbindung festgehalten. Dieses Array dient der späteren Sortierung in Schritt 3.

Die Ankunftsuhrzeit im Zielort wird ebenfalls in einem Array (`$verbindungen_ankunft`) gespeichert, welches der späteren Sortierung dient. Auch hier dient die ID der Verbindung als Schlüssel.

Des weiteren wird die zwischengespeicherte Ausgabe im Array `$verbindungen` gespeichert.

Das Array `$verbindungen` könnte jetzt folgendermaßen aussehen:

```
Array
(
  [0] => Array
    (
      [haltestellen] => Array
        (
          [0] => A
          [1] => B
          [2] => C
          [3] => D
          [4] => J
          [5] => K
          [6] => H
          [7] => I
        )
      [linien] => Array
        (
```

```
        [0] => 1
        [1] => 1
        [2] => 1
        [3] => 4
        [4] => 2
        [5] => 2
        [6] => 2
    )

    [einstiege] => 3
    [doku] => Array
    (
        [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
        [1] => Ausstieg aus dem Bus der Linie 1 in D um 17:20
        [2] => Einstieg in den Bus der Linie 4 in D um 17:20
        [3] => Ausstieg aus dem Bus der Linie 4 in J um 17:30
        [4] => Einstieg in den Bus der Linie 2 in J um 17:30
        [5] => Ausstieg aus dem Bus der Linie 2 in I um 18:25 (Ziel)
        [6] => Fahrtkosten: 3 Neckarmonische Taler
    )
)

[1] => Array
(
    [haltestellen] => Array
    (
        [0] => A
        [1] => B
        [2] => C
        [3] => D
        ...
    )
    ...
)
```

\$verbindungen\_preis könnte so aussehen:

```
Array
(
    [0] => 3
    [1] => 4
    [2] => 2
    [3] => 5
    [4] => 6
    [5] => 4
    [6] => 3
    [7] => 2
)
```

Und \$verbindungen\_ankunft so:

```
Array
(
    [0] => 1105
    [1] => 1070
    [2] => 1052
    [3] => 1095
    [4] => 1060
    [5] => 1042
    [6] => 1045
    [7] => 1010
)
```

Es folgt **Schritt 3**...

*schnellste Verbindung(en)*

Das Array `$verbindungen_ankunft` wird an die Funktion `verbindung_ankunft` übergeben:

```
verbindung_ankunft($verbindungen_ankunft);
```

Diese Funktion sortiert das Array mit den Verbindungen erst einmal aufsteigend nach der Ankunftszeit (den Werten im Array) und gibt anschließend die durch Vergleich beim Durchlauf einer Schleife die ID bzw. IDs der schnellste(n) Verbindung(en) in einem Array zurück.

Beinhaltet dieses Array nur ein Element – gibt es also eine eindeutig schnellste Verbindung – so wird die zugehörige Verbindung ausgegeben. Andernfalls wird ein neues Array - `$p` – zusammengestellt, welches wie `$verbindungen_preis` aufgebaut ist, allerdings nur noch die IDs der schnellsten Verbindungen beinhaltet.

Dieses Array wird jetzt an die Funktion `verbindung_preis` übergeben:

```
verbindung_preis($p);
```

Diese Funktion arbeitet ähnlich wie die Funktion `verbindung_ankunft`. Hier wird das Array allerdings aufsteigend nach den Fahrtkosten (die auch hier die Werte im Array darstellen) sortiert und anschließend die ID bzw. IDs der günstigste(n) Verbindung(en) in einem Array zurückgegeben.

Beinhaltet dieses Array nur ein Element – gibt es also eine eindeutig günstigste Verbindung (unter den schnellsten) – so wird die zugehörige ID der Verbindung ausgegeben. Andernfalls werden die günstigsten der schnellsten Verbindungen ausgegeben.

*kostengünstigste Verbindung(en)*

Die Berechnung der kostengünstigste(n) Verbindung(en) erfolgt analog der Berechnung der schnellsten Verbindung(en); siehe dazu auch Lösungsidee / Schritt 3 und Quellcode.

Genutzt werden wieder die Funktionen `verbindung_ankunft` und `verbindung_preis` - diesmal nur in vertauschten Rollen.

*Anmerkungen:*

Die Ausgabe liese sich bezüglich der Strukturierung und Formatierung natürlich noch verbessern. Im Moment werden Daten nur mit dem Content-type `text/plain` vom Server übergeben. Das hat sich zu Debug- und Entwicklungszwecken sehr stark positiv ausgewirkt. So konnten diverse Strukturen und Inhalte von Variablen über die PHP-Funktion `print_r` anschaulich dargestellt werden. Beigetragen dazu hat die Ausgabe über eine dicktengleiche Schrift im Browser (verursacht durch den Content-type). So konnte auch in Schritt 1 die Rekursion anschaulich dargestellt werden. Hilfreich dazu war auch die Funktion `r`, welche je nach Rekursionstiefe mehr oder weniger Leerzeichen ausgegeben hat.

Ich habe noch extra noch Reste einer genaueren Ausgabe (für eine bessere Protokollierung) übriggelassen. Diese lässt sich in `calc.php` einstellen in dem man `$advanced_output` den booleschen Wert `true` zuweist.

Zum Vergleich werden im folgenden Abschnitt verschiedene Programm-Protokolle mitgeliefert.

### Programm-Protokoll

❶ Start: A — Ziel: B — Gewünschte Startzeit: 16:30

Start: A; Ziel: I  
Gewünschte Startzeit: 16:30

-----  
Schnellste Verbindung:

Einstieg in den Bus der Linie 1 in A um 16:30 (Start)  
Ausstieg aus dem Bus der Linie 1 in B um 16:38  
Einstieg in den Bus der Linie 3 in B um 16:40  
Ausstieg aus dem Bus der Linie 3 in I um 16:50 (Ziel)  
Fahrkosten: 2 Neckarmonische Taler

-----  
Schnellste günstigste Verbindung:

Einstieg in den Bus der Linie 1 in A um 16:30 (Start)  
Ausstieg aus dem Bus der Linie 1 in B um 16:38  
Einstieg in den Bus der Linie 3 in B um 16:40  
Ausstieg aus dem Bus der Linie 3 in I um 16:50 (Ziel)  
Fahrkosten: 2 Neckarmonische Taler

### Anmerkung:

Das Beispiel der Ausgabe auf dem Aufgabenblatt beinhaltet einen Fehler. Ich habe die gleiche Fahrt mit dem Programm getestet. Die Ankunftszeit in I stimmt nicht ganz es fehlt eine Minute (wohl die oblogatorische Minute, die der Bus in B gewartet hat).

❷ Start: A — Ziel: B — Gewünschte Startzeit: 16:30 — erweiterte Ausgabe **aktiviert**

Start: A; Ziel: I  
Gewünschte Startzeit: 16:30

aktuelle Haltestelle: A

Fahren (von A) mit Linie 1 möglich? ja  
Stop B [History: A]

aktuelle Haltestelle: B

Fahren (von B) mit Linie 1 möglich? ja  
Stop C [History: A, B]

aktuelle Haltestelle: C

Fahren (von C) mit Linie 1 möglich? ja  
Stop D [History: A, B, C]

aktuelle Haltestelle: D

Fahren (von D) mit Linie 1 möglich? ja  
Stop E [History: A, B, C, D]

aktuelle Haltestelle: E

Fahren (von E) mit Linie 1 möglich? ja  
Stop F [History: A, B, C, D, E]

aktuelle Haltestelle: F

Fahren (von F) mit Linie 1 möglich? ja  
Stop G [History: A, B, C, D, E, F]

aktuelle Haltestelle: G

Fahren (von G) mit Linie 1 möglich? ja  
Stop A [History: A, B, C, D, E, F, G]  
--> Haltestelle lag bereits auf bisherigem  
Weg

Fahren (von G) mit Linie 2 möglich? nein

Fahren (von G) mit Linie 3 möglich? nein

Fahren (von G) mit Linie 4 möglich? nein

Fahren (von G) mit Linie 5 möglich? nein

Fahren (von F) mit Linie 2 möglich? nein

Fahren (von F) mit Linie 3 möglich? nein

Fahren (von F) mit Linie 4 möglich? nein

Fahren (von F) mit Linie 5 möglich? nein

Fahren (von E) mit Linie 2 möglich? nein

Fahren (von E) mit Linie 3 möglich? nein

Fahren (von E) mit Linie 4 möglich? nein

Fahren (von E) mit Linie 5 möglich? nein

Fahren (von D) mit Linie 2 möglich? nein

Fahren (von D) mit Linie 3 möglich? nein

Fahren (von D) mit Linie 4 möglich? ja  
Stop J [History: A, B, C, D]

aktuelle Haltestelle: J

Fahren (von J) mit Linie 1 möglich? nein

Fahren (von J) mit Linie 2 möglich? ja  
Stop K [History: A, B, C, D, J]

aktuelle Haltestelle: K

Fahren (von K) mit Linie 1 möglich? nein

Fahren (von K) mit Linie 2 möglich? ja  
Stop H [History: A, B, C, D, J, K]

aktuelle Haltestelle: H

Fahren (von H) mit Linie 1 möglich? nein

Fahren (von H) mit Linie 2 möglich? ja  
Stop I [History: A, B, C, D, J, K, H]  
--> Ziel erreicht!

Fahren (von H) mit Linie 3 möglich? nein

Fahren (von H) mit Linie 4 möglich? nein

Fahren (von H) mit Linie 5 möglich? nein

Fahren (von K) mit Linie 3 möglich? ja

```

    Stop I [History: A, B, C, D, J, K]
      --> Ziel erreicht!

    Fahren (von K) mit Linie 4 möglich? nein

    Fahren (von K) mit Linie 5 möglich? nein

    Fahren (von J) mit Linie 3 möglich? nein

    Fahren (von J) mit Linie 4 möglich? ja
    Stop I [History: A, B, C, D, J]
      --> Ziel erreicht!

    Fahren (von J) mit Linie 5 möglich? nein

    Fahren (von D) mit Linie 5 möglich? ja
    Stop C [History: A, B, C, D]
      --> Haltestelle lag bereits auf bisherigem Weg

    Fahren (von C) mit Linie 2 möglich? nein

    Fahren (von C) mit Linie 3 möglich? nein

    Fahren (von C) mit Linie 4 möglich? nein

    Fahren (von C) mit Linie 5 möglich? ja
    Stop L [History: A, B, C]

    aktuelle Haltestelle: L

    Fahren (von L) mit Linie 1 möglich? nein

    Fahren (von L) mit Linie 2 möglich? nein

    Fahren (von L) mit Linie 3 möglich? nein

    Fahren (von L) mit Linie 4 möglich? nein

    Fahren (von L) mit Linie 5 möglich? ja
    Stop D [History: A, B, C, L]

    aktuelle Haltestelle: D

    Fahren (von D) mit Linie 1 möglich? ja
    Stop E [History: A, B, C, L, D]

    aktuelle Haltestelle: E

    Fahren (von E) mit Linie 1 möglich? ja
    Stop F [History: A, B, C, L, D, E]

    aktuelle Haltestelle: F

    Fahren (von F) mit Linie 1 möglich? ja
    Stop G [History: A, B, C, L, D, E, F]

    aktuelle Haltestelle: G

    Fahren (von G) mit Linie 1 möglich? ja
    Stop A [History: A, B, C, L, D, E, F, G]
      --> Haltestelle lag bereits auf
          bisherigem Weg

    Fahren (von G) mit Linie 2 möglich? nein

    Fahren (von G) mit Linie 3 möglich? nein

    Fahren (von G) mit Linie 4 möglich? nein

    Fahren (von G) mit Linie 5 möglich? nein
```

```
Fahren (von F) mit Linie 2 möglich? nein
Fahren (von F) mit Linie 3 möglich? nein
Fahren (von F) mit Linie 4 möglich? nein
Fahren (von F) mit Linie 5 möglich? nein
Fahren (von E) mit Linie 2 möglich? nein
Fahren (von E) mit Linie 3 möglich? nein
Fahren (von E) mit Linie 4 möglich? nein
Fahren (von E) mit Linie 5 möglich? nein
Fahren (von D) mit Linie 2 möglich? nein
Fahren (von D) mit Linie 3 möglich? nein
Fahren (von D) mit Linie 4 möglich? ja
  Stop J [History: A, B, C, L, D]
aktuelle Haltestelle: J
Fahren (von J) mit Linie 1 möglich? nein
Fahren (von J) mit Linie 2 möglich? ja
  Stop K [History: A, B, C, L, D, J]
aktuelle Haltestelle: K
Fahren (von K) mit Linie 1 möglich? nein
Fahren (von K) mit Linie 2 möglich? ja
  Stop H [History: A, B, C, L, D, J, K]
aktuelle Haltestelle: H
Fahren (von H) mit Linie 1 möglich? nein
Fahren (von H) mit Linie 2 möglich? ja
  Stop I [History: A, B, C, L, D, J, K,
        H]
  --> Ziel erreicht!
Fahren (von H) mit Linie 3 möglich? nein
Fahren (von H) mit Linie 4 möglich? nein
Fahren (von H) mit Linie 5 möglich? nein
Fahren (von K) mit Linie 3 möglich? ja
  Stop I [History: A, B, C, L, D, J, K]
  --> Ziel erreicht!
Fahren (von K) mit Linie 4 möglich? nein
Fahren (von K) mit Linie 5 möglich? nein
Fahren (von J) mit Linie 3 möglich? nein
Fahren (von J) mit Linie 4 möglich? ja
  Stop I [History: A, B, C, L, D, J]
  --> Ziel erreicht!
Fahren (von J) mit Linie 5 möglich? nein
Fahren (von D) mit Linie 5 möglich? ja
  Stop C [History: A, B, C, L, D]
```

```
--> Haltestelle lag bereits auf bisherigem Weg

Fahren (von B) mit Linie 2 möglich? nein

Fahren (von B) mit Linie 3 möglich? ja
  Stop K [History: A, B]

  aktuelle Haltestelle: K

    Fahren (von K) mit Linie 1 möglich? nein

    Fahren (von K) mit Linie 2 möglich? ja
      Stop H [History: A, B, K]

      aktuelle Haltestelle: H

        Fahren (von H) mit Linie 1 möglich? nein

        Fahren (von H) mit Linie 2 möglich? ja
          Stop I [History: A, B, K, H]
          --> Ziel erreicht!

        Fahren (von H) mit Linie 3 möglich? nein

        Fahren (von H) mit Linie 4 möglich? nein

        Fahren (von H) mit Linie 5 möglich? nein

        Fahren (von K) mit Linie 3 möglich? ja
          Stop I [History: A, B, K]
          --> Ziel erreicht!

        Fahren (von K) mit Linie 4 möglich? nein

        Fahren (von K) mit Linie 5 möglich? nein

    Fahren (von B) mit Linie 4 möglich? nein

    Fahren (von B) mit Linie 5 möglich? nein

  Fahren (von A) mit Linie 2 möglich? nein

  Fahren (von A) mit Linie 3 möglich? nein

  Fahren (von A) mit Linie 4 möglich? nein

  Fahren (von A) mit Linie 5 möglich? nein
```

Es wurden 8 verschiedene Wege von A nach I ermittelt.

\$verbindungen:

```
Array
(
  [0] => Array
    (
      [haltestellen] => Array
        (
          [0] => A
          [1] => B
          [2] => C
          [3] => D
          [4] => J
          [5] => K
          [6] => H
          [7] => I
        )
      [linien] => Array
        (
          [0] => 1
```

```
        [1] => 1
        [2] => 1
        [3] => 4
        [4] => 2
        [5] => 2
        [6] => 2
    )
[einstiege] => 3
[doku] => Array
(
    [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
    [1] => Ausstieg aus dem Bus der Linie 1 in D um 17:20
    [2] => Einstieg in den Bus der Linie 4 in D um 17:20
    [3] => Ausstieg aus dem Bus der Linie 4 in J um 17:30
    [4] => Einstieg in den Bus der Linie 2 in J um 17:30
    [5] => Ausstieg aus dem Bus der Linie 2 in I um 18:25 (Ziel)
    [6] => Fahrtkosten: 3 Neckarmonische Taler
)
)
[1] => Array
(
    [haltestellen] => Array
    (
        [0] => A
        [1] => B
        [2] => C
        [3] => D
        [4] => J
        [5] => K
        [6] => I
    )
    [linien] => Array
    (
        [0] => 1
        [1] => 1
        [2] => 1
        [3] => 4
        [4] => 2
        [5] => 3
    )
    [einstiege] => 4
    [doku] => Array
    (
        [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
        [1] => Ausstieg aus dem Bus der Linie 1 in D um 17:20
        [2] => Einstieg in den Bus der Linie 4 in D um 17:20
        [3] => Ausstieg aus dem Bus der Linie 4 in J um 17:30
        [4] => Einstieg in den Bus der Linie 2 in J um 17:30
        [5] => Ausstieg aus dem Bus der Linie 2 in K um 17:44
        [6] => Einstieg in den Bus der Linie 3 in K um 17:45
        [7] => Ausstieg aus dem Bus der Linie 3 in I um 17:50 (Ziel)
        [8] => Fahrtkosten: 4 Neckarmonische Taler
    )
)
[2] => Array
(
    [haltestellen] => Array
    (
        [0] => A
        [1] => B
        [2] => C
        [3] => D
        [4] => J
        [5] => I
    )

```

```
)
[linien] => Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 4
    [4] => 4
)

[einstiege] => 2
[doku] => Array
(
    [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
    [1] => Ausstieg aus dem Bus der Linie 1 in D um 17:20
    [2] => Einstieg in den Bus der Linie 4 in D um 17:20
    [3] => Ausstieg aus dem Bus der Linie 4 in I um 17:32 (Ziel)
    [4] => Fahrtkosten: 2 Neckarmonische Taler
)
)

[3] => Array
(
    [haltestellen] => Array
    (
        [0] => A
        [1] => B
        [2] => C
        [3] => L
        [4] => D
        [5] => J
        [6] => K
        [7] => H
        [8] => I
    )

    [linien] => Array
    (
        [0] => 1
        [1] => 1
        [2] => 5
        [3] => 5
        [4] => 4
        [5] => 2
        [6] => 2
        [7] => 2
    )

    [einstiege] => 5
    [doku] => Array
    (
        [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
        [1] => Ausstieg aus dem Bus der Linie 1 in C um 16:52
        [2] => Einstieg in den Bus der Linie 5 in C um 16:55
        [3] => Ausstieg aus dem Bus der Linie 5 in L um 17:01
        [4] => Einstieg in den Bus der Linie 5 in L um 17:05
        [5] => Ausstieg aus dem Bus der Linie 5 in D um 17:07
        [6] => Einstieg in den Bus der Linie 4 in D um 17:10
        [7] => Ausstieg aus dem Bus der Linie 4 in J um 17:20
        [8] => Einstieg in den Bus der Linie 2 in J um 17:20
        [9] => Ausstieg aus dem Bus der Linie 2 in I um 18:15 (Ziel)
        [10] => Fahrtkosten: 5 Neckarmonische Taler
    )
)

[4] => Array
(
    [haltestellen] => Array
```

```
(
    [0] => A
    [1] => B
    [2] => C
    [3] => L
    [4] => D
    [5] => J
    [6] => K
    [7] => I
)

[linien] => Array
(
    [0] => 1
    [1] => 1
    [2] => 5
    [3] => 5
    [4] => 4
    [5] => 2
    [6] => 3
)

[einstiege] => 6
[doku] => Array
(
    [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
    [1] => Ausstieg aus dem Bus der Linie 1 in C um 16:52
    [2] => Einstieg in den Bus der Linie 5 in C um 16:55
    [3] => Ausstieg aus dem Bus der Linie 5 in L um 17:01
    [4] => Einstieg in den Bus der Linie 5 in L um 17:05
    [5] => Ausstieg aus dem Bus der Linie 5 in D um 17:07
    [6] => Einstieg in den Bus der Linie 4 in D um 17:10
    [7] => Ausstieg aus dem Bus der Linie 4 in J um 17:20
    [8] => Einstieg in den Bus der Linie 2 in J um 17:20
    [9] => Ausstieg aus dem Bus der Linie 2 in K um 17:34
    [10] => Einstieg in den Bus der Linie 3 in K um 17:35
    [11] => Ausstieg aus dem Bus der Linie 3 in I um 17:40 (Ziel)
    [12] => Fahrtkosten: 6 Neckarmonische Taler
)
)

[5] => Array
(
    [haltestellen] => Array
    (
        [0] => A
        [1] => B
        [2] => C
        [3] => L
        [4] => D
        [5] => J
        [6] => I
    )

    [linien] => Array
    (
        [0] => 1
        [1] => 1
        [2] => 5
        [3] => 5
        [4] => 4
        [5] => 4
    )

    [einstiege] => 4
    [doku] => Array
    (
        [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
        [1] => Ausstieg aus dem Bus der Linie 1 in C um 16:52
        [2] => Einstieg in den Bus der Linie 5 in C um 16:55
    )
)
)
```

```

    [3] => Ausstieg aus dem Bus der Linie 5 in L um 17:01
    [4] => Einstieg in den Bus der Linie 5 in L um 17:05
    [5] => Ausstieg aus dem Bus der Linie 5 in D um 17:07
    [6] => Einstieg in den Bus der Linie 4 in D um 17:10
    [7] => Ausstieg aus dem Bus der Linie 4 in I um 17:22 (Ziel)
    [8] => Fahrtkosten: 4 Neckarmonische Taler
  )
)
[6] => Array
(
  [haltestellen] => Array
  (
    [0] => A
    [1] => B
    [2] => K
    [3] => H
    [4] => I
  )

  [linien] => Array
  (
    [0] => 1
    [1] => 3
    [2] => 2
    [3] => 2
  )

  [einstiege] => 3
  [doku] => Array
  (
    [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
    [1] => Ausstieg aus dem Bus der Linie 1 in B um 16:38
    [2] => Einstieg in den Bus der Linie 3 in B um 16:40
    [3] => Ausstieg aus dem Bus der Linie 3 in K um 16:43
    [4] => Einstieg in den Bus der Linie 2 in K um 16:50
    [5] => Ausstieg aus dem Bus der Linie 2 in I um 17:25 (Ziel)
    [6] => Fahrtkosten: 3 Neckarmonische Taler
  )
)
[7] => Array
(
  [haltestellen] => Array
  (
    [0] => A
    [1] => B
    [2] => K
    [3] => I
  )

  [linien] => Array
  (
    [0] => 1
    [1] => 3
    [2] => 3
  )

  [einstiege] => 2
  [doku] => Array
  (
    [0] => Einstieg in den Bus der Linie 1 in A um 16:30 (Start)
    [1] => Ausstieg aus dem Bus der Linie 1 in B um 16:38
    [2] => Einstieg in den Bus der Linie 3 in B um 16:40
    [3] => Ausstieg aus dem Bus der Linie 3 in I um 16:50 (Ziel)
    [4] => Fahrtkosten: 2 Neckarmonische Taler
  )
)
)
```

```
)

$verbindungen_preis:
Array
(
    [0] => 3
    [1] => 4
    [2] => 2
    [3] => 5
    [4] => 6
    [5] => 4
    [6] => 3
    [7] => 2
)

$verbindungen_preis [nach Sortierung]:
Array
(
    [7] => 2
    [2] => 2
    [6] => 3
    [0] => 3
    [5] => 4
    [1] => 4
    [3] => 5
    [4] => 6
)

$verbindungen_ankunft:
Array
(
    [0] => 1105
    [1] => 1070
    [2] => 1052
    [3] => 1095
    [4] => 1060
    [5] => 1042
    [6] => 1045
    [7] => 1010
)

$verbindungen_ankunft [nach Sortierung]:
Array
(
    [7] => 1010
    [5] => 1042
    [6] => 1045
    [2] => 1052
    [4] => 1060
    [1] => 1070
    [3] => 1095
    [0] => 1105
)
)
```

-----

Schnellste Verbindung:  
Einstieg in den Bus der Linie 1 in A um 16:30 (Start)  
Ausstieg aus dem Bus der Linie 1 in B um 16:38  
Einstieg in den Bus der Linie 3 in B um 16:40  
Ausstieg aus dem Bus der Linie 3 in I um 16:50 (Ziel)  
Fahrtkosten: 2 Neckarmonische Taler

-----

Schnellste günstigste Verbindung:  
Einstieg in den Bus der Linie 1 in A um 16:30 (Start)  
Ausstieg aus dem Bus der Linie 1 in B um 16:38  
Einstieg in den Bus der Linie 3 in B um 16:40  
Ausstieg aus dem Bus der Linie 3 in I um 16:50 (Ziel)  
Fahrtkosten: 2 Neckarmonische Taler

**3 Start: B — Ziel: L — Gewünschte Startzeit: 20:00**

Start: B; Ziel: L  
Gewünschte Startzeit: 20:00

-----  
Schnellste Verbindung:

Einstieg in den Bus der Linie 1 in B um 20:00 (Start)  
Ausstieg aus dem Bus der Linie 1 in C um 20:12  
Einstieg in den Bus der Linie 5 in C um 20:15  
Ausstieg aus dem Bus der Linie 5 in L um 20:21 (Ziel)  
Fahrkosten: 2 Neckarmonische Taler

-----  
Günstigste Verbindung:

Einstieg in den Bus der Linie 1 in B um 20:00 (Start)  
Ausstieg aus dem Bus der Linie 1 in C um 20:12  
Einstieg in den Bus der Linie 5 in C um 20:15  
Ausstieg aus dem Bus der Linie 5 in L um 20:21 (Ziel)  
Fahrkosten: 2 Neckarmonische Taler

**4 Start: D — Ziel: G — Gewünschte Startzeit: 23:59**

Start: D; Ziel: G  
Gewünschte Startzeit: 23:59

-----  
Schnellste Verbindung:

ACHTUNG: Anschluss erst am Folgetag!  
Einstieg in den Bus der Linie 4 in D um 06:00 (Start)  
Ausstieg aus dem Bus der Linie 4 in I um 06:12  
Einstieg in den Bus der Linie 3 in I um 06:15  
Ausstieg aus dem Bus der Linie 3 in F um 06:17  
Einstieg in den Bus der Linie 1 in F um 06:20  
Ausstieg aus dem Bus der Linie 1 in G um 06:28 (Ziel)  
Fahrkosten: 3 Neckarmonische Taler

-----  
Günstigste Verbindung:

ACHTUNG: Anschluss erst am Folgetag!  
Einstieg in den Bus der Linie 1 in D um 06:00 (Start)  
Ausstieg aus dem Bus der Linie 1 in G um 06:48 (Ziel)  
Fahrkosten: 1 Neckarmonischer Taler

**5 Start: B — Ziel: F — Gewünschte Startzeit: 05:23**

Start: B; Ziel: F  
Gewünschte Startzeit: 05:23

-----  
Schnellste Verbindung:

Einstieg in den Bus der Linie 1 in B um 06:00 (Start)  
Ausstieg aus dem Bus der Linie 1 in C um 06:12  
Einstieg in den Bus der Linie 5 in C um 06:15  
Ausstieg aus dem Bus der Linie 5 in L um 06:21  
Einstieg in den Bus der Linie 5 in L um 06:25  
Ausstieg aus dem Bus der Linie 5 in D um 06:27  
Einstieg in den Bus der Linie 1 in D um 06:30  
Ausstieg aus dem Bus der Linie 1 in F um 07:10 (Ziel)  
Fahrkosten: 4 Neckarmonische Taler

-----  
Schnellste günstigste Verbindung:

Einstieg in den Bus der Linie 1 in B um 06:00 (Start)  
Ausstieg aus dem Bus der Linie 1 in F um 07:20 (Ziel)  
Fahrkosten: 1 Neckarmonischer Taler

⑥ Start: H — Ziel: A — Gewünschte Startzeit: 18:14

Start: H; Ziel: A  
Gewünschte Startzeit: 18:14

aktuelle Haltestelle: H

Fahren (von H) mit Linie 1 möglich? nein

Fahren (von H) mit Linie 2 möglich? ja  
Stop I [History: H]

aktuelle Haltestelle: I

Fahren (von I) mit Linie 1 möglich? nein

Fahren (von I) mit Linie 2 möglich? ja  
Stop J [History: H, I]

aktuelle Haltestelle: J

Fahren (von J) mit Linie 1 möglich? nein

Fahren (von J) mit Linie 2 möglich? ja  
Stop K [History: H, I, J]

aktuelle Haltestelle: K

Fahren (von K) mit Linie 1 möglich? nein

Fahren (von K) mit Linie 2 möglich? ja  
Stop H [History: H, I, J, K]  
--> Haltestelle lag bereits auf bisherigem Weg

Fahren (von K) mit Linie 3 möglich? ja  
Stop I [History: H, I, J, K]  
--> Haltestelle lag bereits auf bisherigem Weg

Fahren (von K) mit Linie 4 möglich? nein

Fahren (von K) mit Linie 5 möglich? nein

Fahren (von J) mit Linie 3 möglich? nein

Fahren (von J) mit Linie 4 möglich? ja  
Stop I [History: H, I, J]  
--> Haltestelle lag bereits auf bisherigem Weg

Fahren (von J) mit Linie 5 möglich? nein

Fahren (von I) mit Linie 3 möglich? ja  
Stop F [History: H, I]

aktuelle Haltestelle: F

Fahren (von F) mit Linie 1 möglich? ja  
Stop G [History: H, I, F]

aktuelle Haltestelle: G

Fahren (von G) mit Linie 1 möglich? ja  
Stop A [History: H, I, F, G]  
--> Ziel erreicht!

Fahren (von G) mit Linie 2 möglich? nein

Fahren (von G) mit Linie 3 möglich? nein

Fahren (von G) mit Linie 4 möglich? nein

```
Fahren (von G) mit Linie 5 möglich? nein
Fahren (von F) mit Linie 2 möglich? nein
Fahren (von F) mit Linie 3 möglich? nein
Fahren (von F) mit Linie 4 möglich? nein
Fahren (von F) mit Linie 5 möglich? nein
Fahren (von I) mit Linie 4 möglich? ja
  Stop H [History: H, I]
  --> Haltestelle lag bereits auf bisherigem Weg
Fahren (von I) mit Linie 5 möglich? nein
Fahren (von H) mit Linie 3 möglich? nein
Fahren (von H) mit Linie 4 möglich? nein
Fahren (von H) mit Linie 5 möglich? nein

Es wurden 1 verschiedene Wege von H nach A ermittelt.

$verbindungen:
Array
(
  [0] => Array
    (
      [haltstellen] => Array
        (
          [0] => H
          [1] => I
          [2] => F
          [3] => G
          [4] => A
        )
      [linien] => Array
        (
          [0] => 2
          [1] => 3
          [2] => 1
          [3] => 1
        )
      [einstiege] => 3
      [doku] => Array
        (
          [0] => Einstieg in den Bus der Linie 2 in H um 18:20 (Start)
          [1] => Ausstieg aus dem Bus der Linie 2 in I um 18:35
          [2] => ACHTUNG: Anschluss erst am Folgetag!
          [3] => Einstieg in den Bus der Linie 3 in I um 09:00
          [4] => Ausstieg aus dem Bus der Linie 3 in F um 09:02
          [5] => Einstieg in den Bus der Linie 1 in F um 09:10
          [6] => Ausstieg aus dem Bus der Linie 1 in A um 09:30 (Ziel)
          [7] => Fahrtkosten: 3 Neckarmonische Taler
        )
      )
    )
)

$verbindungen_preis:
Array
(
  [0] => 3
)
```

```
$verbindungen_preis [nach Sortierung]:  
Array  
(  
  [0] => 3  
)
```

```
$verbindungen_ankunft:  
Array  
(  
  [0] => 1435  
)
```

```
$verbindungen_ankunft [nach Sortierung]:  
Array  
(  
  [0] => 1435  
)
```

-----  
Schnellste Verbindung:

```
Einstieg in den Bus der Linie 2 in H um 18:20 (Start)  
Ausstieg aus dem Bus der Linie 2 in I um 18:35  
ACHTUNG: Anschluss erst am Folgetag!  
Einstieg in den Bus der Linie 3 in I um 09:00  
Ausstieg aus dem Bus der Linie 3 in F um 09:02  
Einstieg in den Bus der Linie 1 in F um 09:10  
Ausstieg aus dem Bus der Linie 1 in A um 09:30 (Ziel)  
Fahrtkosten: 3 Neckarmonische Taler
```

-----  
Günstigste Verbindung:

```
Einstieg in den Bus der Linie 2 in H um 18:20 (Start)  
Ausstieg aus dem Bus der Linie 2 in I um 18:35  
ACHTUNG: Anschluss erst am Folgetag!  
Einstieg in den Bus der Linie 3 in I um 09:00  
Ausstieg aus dem Bus der Linie 3 in F um 09:02  
Einstieg in den Bus der Linie 1 in F um 09:10  
Ausstieg aus dem Bus der Linie 1 in A um 09:30 (Ziel)  
Fahrtkosten: 3 Neckarmonische Taler
```

### Programm-Listing

data.inc.php

```
1 <?
2 $linien = array(
3     1 => array(
4         'first_h' => 6,
5         'first_m' => 0,
6         'last_h' => 23,
7         'last_m' => 30,
8         'intervall' => 10,
9         'start' => 'A',
10        'stops' => array(
11            'B' => 8,
12            'C' => 12,
13            'D' => 20,
14            'E' => 10,
15            'F' => 30,
16            'G' => 8,
17            'A' => 10
18        )
19    ),
20    2 => array(
21        'first_h' => 6,
22        'first_m' => 30,
23        'last_h' => 23,
24        'last_m' => 30,
25        'intervall' => 10,
26        'start' => 'J',
27        'stops' => array(
28            'K' => 14,
29            'H' => 15,
30            'I' => 15,
31            'J' => 6
32        )
33    ),
34    3 => array(
35        'first_h' => 9,
36        'first_m' => 0,
37        'last_h' => 18,
38        'last_m' => 30,
39        'intervall' => 5,
40        'start' => 'B',
41        'stops' => array(
42            'K' => 3,
43            'I' => 5,
44            'F' => 2
45        )
46    ),
47    4 => array(
48        'first_h' => 6,
49        'first_m' => 0,
50        'last_h' => 22,
51        'last_m' => 0,
52        'intervall' => 10,
53        'start' => 'D',
54        'stops' => array(
55            'J' => 10,
56            'I' => 2,
57            'H' => 10
58        )
59    ),
60    5 => array(
61        'first_h' => 6,
62        'first_m' => 0,
63        'last_h' => 23,
64        'last_m' => 30,
65        'intervall' => 5,
66        'start' => 'L',
67        'stops' => array(
```

```
68         'D' => 2,  
69         'C' => 18,  
70         'L' => 6  
71     )  
72 )  
73 );  
74 ?>
```

calc.php

```
1 <?
2 header("Content-type: text/plain");
3
4 # erweiterte Ausgabe
5 $advanced_output = false;
6
7 # $linien einbinden
8 include('data.inc.php');
9
10
11 # Daten aus Formular
12 $start = $_POST['start'];
13 $ziel = $_POST['ziel'];
14
15 $h = $_POST['h'];
16 $m = $_POST['m'];
17
18
19 # Funktion, um dokumentierende Ausgabe wegen Rekursion strukturiert einzurücken
20 function r($r) {
21     $u = '';
22     for($t = 0; $t < $r; $t++) {
23         $u .= '    ';
24     }
25     return $u;
26 }
27
28
29 # Ist es möglich mit Linie $linie von $stop aus zu fahren?
30 function fahre_mit_linie($linien, $linie, $stop, &$amp;position) {
31     # linien: enthält Daten über die Linien
32     # linie: Linie
33     # stop: Haltestelle
34     # position: aktuelle Position auf dem Weg der Linie
35
36     if($linie['start'] == $stop) {
37         # Starthaltestelle der Linie
38         $position = -1;
39         return true;
40     }
41     elseif(array_key_exists($stop, $linie['stops'])) {
42         # Linie hält an dieser Haltestelle
43
44         # Haltestellen der Linie
45         $stops = array_keys($linie['stops']);
46
47         # Position der Haltestelle auf dem Weg der Linie
48         $position = array_search($stop, $stops);
49
50         if($position < count($stops)-1) {
51             return true;
52         }
53         else {
54             # Station ist letzte Haltestelle der Linie; Weiterfahrt mit
55             # dieser Linie nicht möglich
56             return false;
57         }
58     }
59     else {
60         # Linie hält nicht an dieser Haltestelle
61         return false;
62     }
63 }
64 }
65
66
67 # Den Weg zum Ziel finden
68 function finde_weg($linien, $ziel, $history, $linien_history, $r) {
69     global $verbindungen, $advanced_output;
70     # linien: enthält Daten über die Linien
```

```
71 # ziel:      Ziel
72 # history:   [Array] enthält die Haltestellen auf dem bisherigen Weg
73 # l~story:   [Array] enthält die Linien auf dem bisherigen Weg
74 # r:        Rekursionstiefe
75
76 # aktuelle Haltestelle aus der History ermitteln
77 $stop = $history[count($history)-1];
78
79 if($advanced_output)
80     echo "\n", r($r), "aktuelle Haltestelle: $stop\n";
81
82 # Checke für jede Linien, ob man von der aktuellen Haltestelle damit fahren kann
83 foreach($linien as $n => $linie) {
84     $f = fahre_mit_linie($linien, $linie, $stop, $position);
85
86     if($advanced_output)
87         echo "\n", r($r), " Fahren (von $stop) mit Linie $n möglich? ",
88             $f ? 'ja' : 'nein', "\n";
89
90     # temporäre Auslagerung
91     $temp_history = $history;
92     $temp_linien_history = $linien_history;
93
94     if($f) {
95         # Fahren mit der aktuellen Linie möglich
96
97         # Position der Haltestelle in $c speichern
98         $c = 0;
99
100        foreach($linie['stops'] as $next_stop => $time) {
101
102            if($c > $position) {
103                # Haltestelle liegt auf der Strecke der Linie ab aktueller Haltestelle
104
105                # Linie in Linien-History speichern
106                $temp_linien_history[] = $n;
107
108
109                if($advanced_output)
110                    echo r($r), "    Stop $next_stop [History: ",
111                        implode(', ', $temp_history), "]\n";
112
113                if($next_stop == $ziel) {
114                    # Ziel erreicht
115
116                    if($advanced_output)
117                        echo r($r), "        --> Ziel erreicht!\n";
118
119                    # Haltestelle in History schreiben
120                    $temp_history[] = $next_stop;
121
122                    # temporäre Daten in Verbindungsarray speichern
123                    $verbindungen[] = array(
124                        'haltestellen' => $temp_history,
125                        'linien' => $temp_linien_history
126                    );
127                    # Abbruch
128                    break;
129                }
130                elseif(in_array($next_stop, $temp_history)) {
131                    # Haltestelle lag bereits auf dem bisherigen Weg
132
133                    if($advanced_output)
134                        echo r($r),
135                            "        --> Haltestelle lag bereits auf bisherigem Weg\n";
136
137                    # Abbruch
138                    break;
139                }
140
141                # Haltestelle in History schreiben
```

```
142         $temp_history[] = $next_stop;
143
144         # rekursiver Funktionsaufruf; finde den Weg zum Ziel
145         finde_weg($linien, $ziel, $temp_history, $temp_linien_history, $r+1);
146         break;
147     }
148
149     # Position der Haltestelle inkrementieren
150     $c++;
151 }
152 }
153 }
154
155 }
156
157
158 # Zeit der Abfahrt in $von und der Ankunft in $nach ermitteln mit Linie $n ab $jetzt
159 function zeit($jetzt, $von, $nach, $n, $i) {
160     global $linien, $warten, $verbindungen;
161     # jetzt:      "aktuelle" Uhrzeit
162     # von:        Haltestelle von der gefahren wird
163     # nach:       Haltestelle zu der gefahren wird
164     # n:          Nummer der Linie
165     # i:          ID der aktuellen Verbindung
166     # warten:     die Zeit, die bei evtl. Warten auf Anschluss am Folgetag entsteht
167
168     $linie = $linien[$n];
169     $fahrtzeit = 0;
170
171     # 1. Abfahrt der Linie $n von Starthaltestelle
172     $abfahrtszeit = $linie['first_h']*60 + $linie['first_m'];
173
174     if($von == $linie['start']) {
175         # Abfahrt von Starthaltestelle der Linie; keine weiteren Berechnungen nötig
176
177         # Bus fährt erst später als gewünscht
178         if($jetzt < $abfahrtszeit) {
179             $jetzt = $abfahrtszeit;
180         }
181         $fahrtzeit = current($linie['stops']);
182     }
183     else {
184         # Abfahrt erfolgt nicht von der Starthaltestelle der Linie
185
186         $u = false;
187         foreach($linie['stops'] as $stop => $t) {
188
189             # Fahrtzeit auslesen
190             if($u) {
191                 $fahrtzeit = $t;
192                 break;
193             }
194
195             $abfahrtszeit += $t+1; # inkl. je 1 Minute Wartezeit!
196
197
198             if($stop == $von) {
199
200                 # Bus fährt erst später als gewünscht
201                 if($jetzt < $abfahrtszeit) {
202                     $jetzt = $abfahrtszeit;
203                 }
204
205                 # bei nächstem (= letztem) Schleifendurchlauf Fahrtzeit auslesen
206                 $u = true;
207             }
208         }
209     }
210 }
211
212 # letzte Abfahrt an diesem Tag berechnen
```

```
213     $letzte = $abfahrtszeit + $linie['last_h']*60 + $linie['last_m'] -
214         ($linie['first_h']*60 + $linie['first_m']);
215
216     if($jetzt > $letzte) {
217         # letzte Abfahrt verpasst(!); erste Abfahrt am Folgetag nehmen!
218         $warten += 24*60-$jetzt+$abfahrtszeit;
219         $jetzt = $abfahrtszeit;
220         $verbindungen[$i]['doku'][] = "ACHTUNG: Anschluss erst am Folgetag!";
221     }
222
223     # Abfahrtszeit evtl. korrigieren
224     $abfahrtszeit = $abfahrtszeit+
225         ceil(($jetzt-$abfahrtszeit)/$linie['intervall'])*$linie['intervall'];
226     $ankunftszeit = $abfahrtszeit + $fahrtzeit;
227
228     return array($abfahrtszeit, $ankunftszeit);
229 }
230
231
232 # Tagesminuten in "normalem" HH:mm-Format ausgeben
233 function mat($zeit) {
234
235     $h = floor($zeit/60);
236     $m = $zeit%60;
237     return sprintf("%02d:%02d", $h, $m);
238 }
239 }
240
241
242 # zeitlich minimale Verbindung[en] heraussuchen
243 function verbindung_ankunft($verbindungen) {
244     asort($verbindungen);
245
246     # i: ID der Verbindung
247     foreach($verbindungen as $i => $ankunft) {
248         if($ankunft == current($verbindungen)) {
249             $schnellste_verbindungen[] = $i;
250         }
251     }
252     return $schnellste_verbindungen;
253 }
254
255
256 # kostengünstigste Verbindung[en] heraussuchen
257 function verbindung_preis($verbindungen) {
258     asort($verbindungen);
259
260     # i: ID der Verbindung
261     foreach($verbindungen as $i => $preis) {
262         if($preis == current($verbindungen)) {
263             $guenstigste_verbindungen[] = $i;
264         }
265     }
266     return $guenstigste_verbindungen;
267 }
268
269
270 echo "Start: $start; Ziel: $ziel\n";
271 echo "Gewünschte Startzeit: ", sprintf("%02d:%02d", $h,$m), "\n\n";
272
273 # Verbindungen [am Anfang leer]
274 $verbindungen = array();
275
276
277 # Finde den Weg/ die Wege vom Start zum Ziel
278 finde_weg($linien, $ziel, array($start), array(), 0);
279
280 if($advanced_output)
281     echo "\n\nEs wurden ", count($verbindungen), " verschiedene Wege von $start",
282         " nach $ziel ermittelt.\n\n";
```

```
283
284
285 # Die Ankunft [und Preise] der einzelnen Verbindungen berechnen
286 foreach($verbindungen as $i => $verbindung) {
287
288     # Startzeit
289     $zeit = $h*60 + $m;
290     $warten = 0;
291     $start_zeit = $zeit;
292
293     for($k = 0; $k < count($verbindung['linien']); $k++) {
294         # Wege zwischen den einzelnen Stationen
295
296         $von = $verbindung['haltestellen'][$k];
297         $nach = $verbindung['haltestellen'][$k+1];
298         $linie = $verbindung['linien'][$k];
299
300         $zeit = zeit($zeit, $von, $nach, $linie, $i);
301
302         $abfahrtszeit = $zeit[0];
303         $ankunftszeit = $zeit[1];
304
305         # Linienwechsel oder Einstieg an dieser Starthaltestelle
306         if($linie <> $verbindung['linien'][$k-1] ||
307             $verbindung['haltestellen'][$k] ==
308                 $linien[$verbindung['linien'][$k]]['start']) {
309             # einsteigen
310             $verbindungen[$i]['einstiege']++;
311             $verbindungen[$i]['doku'][] = "Einstieg in den Bus der Linie ".$linie
312                 ." in $von um ".mat($abfahrtszeit)
313                 ." ($k == 0 ? ' (Start)' : '');
314         }
315
316         # Linienwechsel oder Einstieg an dieser Starthaltestelle
317         if($linie <> $verbindung['linien'][$k+1] ||
318             $verbindung['haltestellen'][$k+1] ==
319                 $linien[$verbindung['linien'][$k+1]]['start']) {
320             # aussteigen
321             $verbindungen[$i]['doku'][] = "Ausstieg aus dem Bus der Linie ".$linie
322                 ." in $nach um ".mat($ankunftszeit)
323                 ." ($k == count($verbindung['linien'])-1 ? ' (Ziel)' : '');
324         }
325
326         $zeit = $ankunftszeit;
327     }
328
329     # Fahrtkosten berechnen
330     $taler = 1; # 1 Taler pro Einstieg (anpassbar)
331     $kosten = $verbindungen[$i]['einstiege']*$taler;
332     $verbindungen[$i]['doku'][] = "Fahrtkosten: $kosten Neckarmonische"
333         ." ($kosten == 1 ? 'r' : '')." Taler";
334
335     # "Wartezeit" zur Zeit hinzuzählen (bei ggf. Warten auf Anschluss am Folgetag)
336     $zeit = $zeit+$warten;
337
338     $verbindungen_preis[$i] = $kosten;
339     $verbindungen_ankunft[$i] = $zeit;
340 }
341
342 # erweiterte dokumentierende Ausgabe
343 if($advanced_output) {
344     echo "\$verbindungen:\n";
345     print_r($verbindungen);
346
347     echo "\n\$verbindungen_preis:\n";
348     print_r($verbindungen_preis);
349     echo "\n\$verbindungen_preis [nach Sortierung]:\n";
350     asort($verbindungen_preis);
351     print_r($verbindungen_preis);
352 }
```

```
352     echo "\n\$verbindungen_ankunft:\n";
353     print_r(\$verbindungen_ankunft);
354     echo "\n\$verbindungen_ankunft [nach Sortierung]:\n";
355     asort(\$verbindungen_ankunft);
356     print_r(\$verbindungen_ankunft);
357
358     echo "\n";
359 }
360
361 # schnellste Verbindung[en]
362 echo "-----\n";
363
364 \$schnellste_verbindungen = verbindung_ankunft(\$verbindungen_ankunft);
365 if(count(\$schnellste_verbindungen) > 1) {
366     # keine eindeutig schnellste Verbindung [-> mehrere zeitgleiche]
367
368     foreach(\$schnellste_verbindungen as \$verbindung) {
369         \$p[\$verbindung] = \$verbindungen_preis[\$verbindung];
370     }
371
372     # günstigste unter den schnellsten Verbindungen suchen
373     \$guenstigste_verbindungen = verbindung_preis(\$p);
374
375     if(count(\$guenstigste_verbindungen) > 1) {
376         # keine eindeutig günstigste schnellste Verbindung
377         # [-> mehrere gleich günstige schnellste Verbindungen]
378         echo "Günstigste schnellste Verbindungen:";
379         foreach(\$guenstigste_verbindungen as \$i) {
380             echo "\n ", implode("\n ",
381                 \$verbindungen[\$guenstigste_verbindungen[0]]['doku']), "\n";
382         }
383     }
384     else {
385         # eindeutig günstigste schnellste Verbindung
386         echo "Günstigste schnellste Verbindung:\n ";
387         echo implode("\n ", \$verbindungen[\$guenstigste_verbindungen[0]]['doku']);
388     }
389 }
390 else {
391     # eindeutig schnellste Verbindung
392     echo "Schnellste Verbindung:\n ";
393     echo implode("\n ", \$verbindungen[\$schnellste_verbindungen[0]]['doku']);
394 }
395
396
397
398 # günstigste Verbindung[en]
399 echo "\n\n-----\n";
400
401 \$preislichgleiche_verbindungen = verbindung_preis(\$verbindungen_preis);
402 if(count(\$preislichgleiche_verbindungen) > 1) {
403     # keine eindeutig günstigste Verbindung [-> mehrere gleich günstig]
404
405     foreach(\$preislichgleiche_verbindungen as \$verbindung) {
406         \$p[\$verbindung] = \$verbindungen_ankunft[\$verbindung];
407     }
408
409     # schnellste unter den günstigsten Verbindungen suchen
410     \$schnellste_verbindungen = verbindung_ankunft(\$p);
411     if(count(\$schnellste_verbindungen) > 1) {
412         # keine eindeutig schnellste günstigste Verbindung
413         # [-> mehrere gleich schnelle günstigste Verbindungen]
414         echo "Günstigste schnellste Verbindungen:";
415         foreach(\$schnellste_verbindungen as \$i) {
416             echo "\n ", implode("\n ",
417                 \$verbindungen[\$guenstigste_verbindungen[0]]['doku']), "\n";
418         }
419     }
420     else {
421         # eindeutig schnellste günstigste Verbindung
422         echo "Schnellste günstigste Verbindung:\n ";
```

```
423     echo implode("\n ", $verbindungen[$schnellste_verbindungen[0]]['doku']);
424     }
425 }
426 else {
427     # eindeutig günstigste Verbindung
428     echo "Günstigste Verbindung:\n ";
429     echo implode("\n ", $verbindungen[$preislichgleiche_verbindungen[0]]['doku']);
430 }
431
432 ?>
```

form.php

```
1 <?
2 include('data.inc.php');
3
4 $haltestellen = array();
5 foreach($linien as $linie) {
6     $haltestellen[] = $linie['start'];
7     $haltestellen = array_merge($haltestellen, array_keys($linie['stops']));
8 }
9 $haltestellen = array_flip($haltestellen);
10 $haltestellen = array_flip($haltestellen);
11 asort($haltestellen);
12
13 ?>
14 <form action="calc.php" method="post">
15 <table>
16 <tr>
17     <td>Startort</td>
18     <td><select name="start">
19 <?php
20 foreach($haltestellen as $haltestelle) {
21     echo " <option>$haltestelle</option>\n";
22 }
23 ?>
24 </select></td>
25 </tr>
26 <td>Zielort</td>
27 <td><select name="ziel">
28 <?php
29 foreach($haltestellen as $haltestelle) {
30     echo " <option>$haltestelle</option>\n";
31 }
32 ?>
33 </select></td>
34 </tr>
35 </tr>
36 <tr>
37     <td>Startzeit</td>
38     <td><select name="h">
39 <?php
40 if(empty($h)) { $h = date("H"); }
41 for($h_form = 0; $h_form <= 23; $h_form++) {
42     echo " <option", $h_form == $h ? " selected" : '', ">";
43     printf("%02d", $h_form);
44     echo "</option>\n";
45 }
46 ?> </select> : <select name="m">
47 <?php
48 if(empty($i)) { $m = date("i"); }
49 for($m_form = 0; $m_form <= 59; $m_form++) {
50     echo " <option", $m_form == $m ? " selected" : '', ">";
51     printf("%02d", $m_form);
52     echo "</option>\n";
53 }
54 ?> </select></td>
55 </tr>
56 <tr>
57     <td>&nbsp;</td>
58     <td><input type="submit" value="Verbindungen ermitteln"></td>
59 </tr>
60 </table>
61 </form>
```

### ***Kommentar***

Bei der Umsetzung der Aufgabe kamen mir ein paar Gedanken, die zu Kritik und Verbesserungsvorschlägen am Nahverkehrssystem anregen.

#### **Problem 1**

Herr Hämmerle, der in J. wohnt und in F. zur Arbeit geht, wird beispielsweise kaum den Nahverkehr nutzen. Für ihn stellt es zwar kein Problem dar mit dem Bus zu seinem Arbeitsplatz zu gelangen, zurück wird das allerdings um vielfaches komplizierter. Die Rückfahrt wird für ihn deutlich ungünstig: sie fällt für ihn auf jeden Fall deutlich länger aus! Er wird also eher zum Auto greifen. Der Verbesserungsvorschlag von meiner Seite lautet nun, dass die Strecken der Linien nicht nur in eine Richtung, sondern auch in die andere Richtung befahren werden, um gerade solche Probleme zu vermeiden. Dann müsste beispielsweise Herr Hämmerle aus J. auch nicht mehr durch halb Neckarmonien fahren, um seine Tante und seinen Onkel in L. zu besuchen.

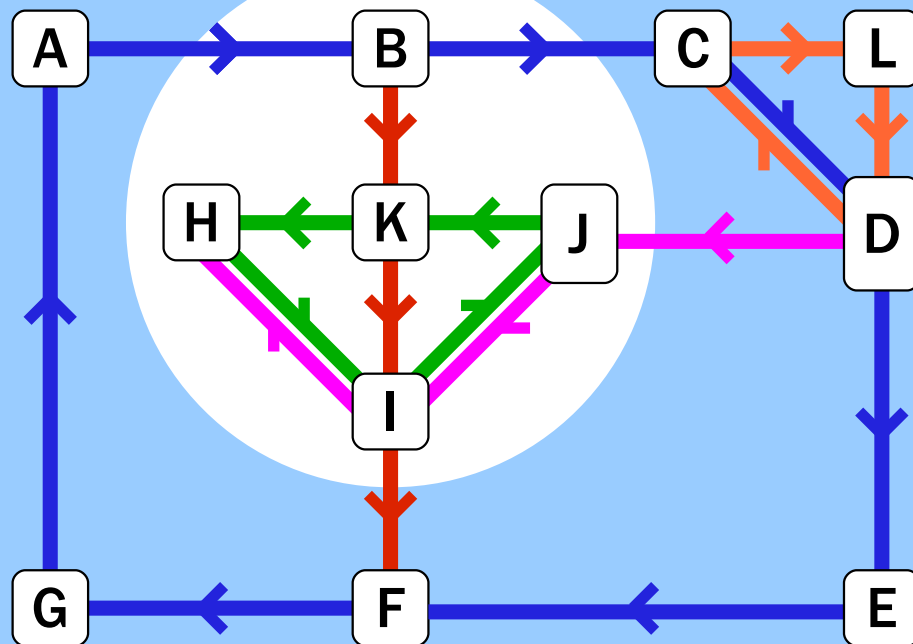
#### **Problem 2**

Bleiben wir bei Herrn Hämmerle. Wenn er zu seiner Tante und zu seinem Onkel nach L. fahren will, muss er immer 5 Neckarmonische Taler zahlen. Das ist ihm zu viel. Doch leider hat er keinen Führerschein und muss so mit dem Nahverkehr zur Arbeit und auch zu seinen Verwandten fahren. Auch andere haben dieses Problem – um von einem Ort zu einem anderen zu kommen müssen sie mit den verschiedensten Linien fahren und für jeden Einstieg zahlen.

Das Zahlen pro Einstiegen ist zwar für den Fürsten von Neckarmonien gewinnbringend, die Bürger könnten es allerdings verärgern, denn schon um von G nach B zu kommen müssen sie zwei mal in einen Bus der Linie 1 (in G und in A) einsteigen. Eventuell wäre der Beruf des Taxifahrers in Neckarmonien ein lukrativer Job! Ich schlage allerdings vor, die Abrechnung anders zu gestalten.

Tages-, Wochen-, Monats- oder Jahreskarten wären bestimmt gerne gesehen. Zudem könnte man Tickets anbieten, die das unbegrenzt lange Fahren an einem Tag (oder einer anderen zeitlichen Begrenzung) ermöglichen. Andernfalls könnte man ein Ticket auch auf eine bestimmte Anzahl von Stationen beschränken und Streifenkarten verkaufen. Es könnte auch eine Unterteilung in zwei Tarifzonen (Innenraum, Außenraum) geben, die folgendermaßen aussehen könnten (vorausgesetzt Neckarmonien sähe geographisch gesehen in etwa so aus):

*siehe nächste Seite*



### Legende



Haltestelle



Fahrtrichtung



Linie 1



Linie 2



Linie 3



Linie 4



Linie 5

**Fazit**

Manche Punkte in Neckarmonien sind einfach nicht gut genug an das System angeschlossen. Glücklicherweise ist mein Skript dynamisch und kann weitere Haltestellen und Buslinien aufnehmen. Die Umgestaltung ist letztendlich aber die Entscheidung des unterstützenden und beratenden Verkehrsministers von Neckarmonien und natürlich beim Fürsten.