# Od soumraku do úsvitu

KIV/PPA2 - Semestrální práce


| | |
|---|---|
| Jméno: | Tomáš Valenta |
| Osobní číslo: | A04179 |
| E-mail: | valentat@students.zcu.cz |
| Datum: | 28. dubna 2005 |

# 1 Zadání

## 1.1 Zdroje

Byl zadán problém č. 10187 z databáze ACM. Původní anglické zadání je dostupné na http://online-judge.uva.es/p/v101/10187.html. Přeložené zadání pak lze nalézt na stránkách předmětu PPA2:
http://www.kiv.zcu.cz/~netrvalo/vyuka/ppa2/cviceni/semestralka.html

## 1.2 Text zadání

Vladimir has white skin, very long teeth and is 600 years old, but this is no problem because Vladimir is a vampire. Vladimir has never had any problems with being a vampire. In fact, he is a very successful doctor who always takes the night shift and so has made many friends among his colleagues. He has a very impressive trick which he shows at dinner partys: He can tell tell blood group by taste. Vladimir loves to travel, but being a vampire he has to overcome three problems.

- First, he can only travel by train because he has to take his coffin with him. (On the up side he can always travel first class because he has invested a lot of money in long term stocks.)
- Second, he can only travel from dusk till dawn, namely from 6 pm to 6 am. During the day he has to stay inside a train station.
- Third, he has to take something to eat with him. He needs one litre of blood per day, which he drinks at noon (12:00) inside his coffin.

You should help Vladimir to find the shortest route between two given cities, so that he can travel with the minimum amount of blood. (If he takes too much with him, people will ask funny questions like "What do you do with all that blood?")

### 1.2.1 Input Specification

The first line of the input will contain a single number telling you the number of test cases. Each test case specification begins with a single number telling you how many route specifications follow. Each route specification consists of the names of two cities, the departure time from city one and the total travelling time. The times are in hours. Note that Vladimir can't use routes departing earlier than 18:00 or arriving later than 6:00. There will be at most 100 cities and less than 1000 connections. No route takes less than one hour and more than 24 hours. (Note that Vladimir can use only routes with a maximum of 12 hours travel time (from dusk till dawn).) All city names are shorter than 32 characters. The last line contains two city names. The first is Vladimir's start city, the second is Vladimir's destination.

### 1.2.2 Output Specification

For each test case you should output the number of the test case followed by "Vladimir needs # litre(s) of blood." or "There is no route Vladimir can take."

#### 1.2.3 Sample Input

```
2
3
Ulm Muenchen 17 2
Ulm Muenchen 19 12
Ulm Muenchen 5 2
Ulm Muenchen
10
Lugoj Sibiu 12 6
Lugoj Sibiu 18 6
Lugoj Sibiu 24 5
Lugoj Medias 22 8
Lugoj Medias 18 8
Lugoj Reghin 17 4
Sibiu Reghin 19 9
Sibiu Medias 20 3
Reghin Medias 20 4
Reghin Bacau 24 6
Lugoj Bacau
```

#### 1.2.4 Sample Output

```
Test Case 1.
There is no route Vladimir can take.
Test Case 2.
Vladimir needs 2 litre(s) of blood.
```

# 2 Vypracování

## 2.1 Součásti

- Vladimir.tex – zdrojový soubor dokumentace
- Vladimir.pdf – přeložený soubor dokumentace
- Main.java – zdrojový soubor projektu v jazyce Java
- dusk.in – vzorový vstup
- dusk.out – schválený výstup

## 2.2 Popis algoritmu

Program po startu načte počet řešených problémů a po načtení každého ihned vypíše řešení. Pro každý problém se načítají na každém řádku jména 2 stanic a 2 časové údaje. Nejprve program ověří, zda stanice již nebyly načteny do spojového seznamu. Pokud ne, přidá je tam. Každá stanice je reprezentována jako instance třídy `Station`. Ta obsahuje další spojový seznam objektů třídy `Connection`, kde jsou uloženy načtené časové údaje a cílová stanice. Nakonec si program zaznamená do proměnných `from` a `to` počáteční a cílovou stanici.

Údaje o spojení jsou ověřovány již při vstupu. Výsledkem vstupu je jakýsi multigraf. Ten je pak prohledáván do šířky. Začne se ve startovní stanici a hledací metoda volá pak sama sebe se stanicemi dosažitelnými z tohoto nádraží a na "hodinách" přibývá čas podle doby jízdy a čekání a ukládá se k jednotlivým stanicím. Pokud program narazí na stanici, kam jsme již dojeli, porovná časový údaj v této stanici s údajem aktuální trasy, lepší zachová a pokračuje s tímto v cestě až do cíle. Pokaždé v poledne je zvětšen čítač litrů. Ten je po posledním dosažení cílové stanice vypsán na výstup.

Program je optimalizován na co nejefektivnější běh na testovacím stroji ACM. Trochu i na úkor přehlednosti.

## 2.3 Popis tříd, metod a polí

### 2.3.1 Station

Slouží na uchování informací o nádraží a průběhu výpočtu. Spravuje spojový seznam stanic.

`String name` - název stanice
`Connection connections` - seznam spojů vedoucích ze stanice
`int duration` - uložení doby jízdy během výpočtu
`Station next` - ukazatel na další nádraží v seznamu

`Station(String name, Station next)` - Vytvoří instanci třídy a připojí ji do seznamu před `next`

### 2.3.2 Connection

Uchovává informace o spoji mezi a spravuje spojový seznam.

`Station target` - cílová stanice
`int from` - hodina odjezdu
`int time` - doba jízdy
`Connection next` - další spojení ze stanice

`Connection(Station target, int from, int time, Connection next)` - Vytvoří instanci třídy se zadanými parametry a zařadí do seznamu před `next`

### 2.3.3 Main

Hlavní třída projektu. Vykonává načítání položek, prohledávání i výstup. Pole i metody jsou staitcké z důvodu výkonu na rototu ACM. Slovo `static` v jejich deklaraci už neuvádím.

`InputStream is` - Vstupní stream do programu. Víceméně k ladícím účelům
`final int BUF_SIZE = 2048` - Velikost vstupního bufferu
`String buf` - vstupní buffer
`int bl` - využití vstupního bufferu
`int gap` - index mezery v bufferu

`int eol` - index odřádkování (`LF`) v bufferu
`int pos` - aktuální pozice v bufferu
`Station stations` - hlava seznamu stanic
`Station from` - startovní stanice
`Station to` - cílová stanice
`int ltrs` - vypočtený počet litrů krve

`void fillBuf()` - načte vstupní buffer
`String inStr()` - vrátí řetězec znaků do dalšího oddělovače
`Station find(String name)` - vrátí objekt nádraží s názvem `name` nebo `null`
`Station add(String name)` - vrátí objekt nádraží s názvem `name`. Pokud ještě není v seznamu, přidá jej.
`void AddConnection()` - načte řádek ze vstupu a zpracuje
`void outln(String s)` - vypíše řádek na výstup
`void inputTest()` - načte celý problém
`void search(Station st, int time, int duration, int litres)` - rekurzivní metoda pro vyhledávání

- `st` - stanice, kde se nacházíme
- `time` - kolik je hodin
- `duration` - jak dlouho už cestujeme
- `litres` - kolik už jsme toho vypili

`void solve()` - spustí řešení a vypíše výsledek
`void main(String[] args)` - hlavní metoda. Načte počet problémů a postupně je všechny zpracuje. V každém nastolí počáteční stav, načte problém (`inputTest()`), vypíše jeho pořadí a zavolá `solve()`.

# 3 Ověření funkčnosti

- Program byl testován na stroji Valladolid ACM Online Judge. To byla ta nejsložitější věc z celé práce, protože stroj si pod pojmem Java představuje jen jakousi "parodii na Javu".
  Mé jméno je v tabulce na http://online-judge.uva.es/cgi-bin/OnlineJudge?ProblemStat:10187.
- Nebo jej možno otestovat na moje jméno nyní na http://acm.uva.es/problemset/submit.php.