OOP Zápočtový program - Diskrétní simulace metra

Jan Oupický

2018

1 Zadání

Program má za úkol provést diskrétní simulaci metra. Cílem této simulace je zjištění doby cesty ze stanice A do stanice B za různých podmínek, jako různá kapacita souprav, různý počet lidí, různý počet souprav apod.

2 Uživatelský manuál

Spolu s programem je distribuován soubor `stanice.txt`, ve kterém je seznam stanic metra. V základu se v něm nachází seznam stanic pražského metra v požadovaném formátu.

Uživatel si ale může tento soubor upravit podle libosti, akorát musí dodržet následující syntaxi:

- Prázdné řádky jsou ignorovány.
- Řádek, který začíná "#", je komentář. Tento řádek je také ignorován.
- Na každém řádku se může nacházet pouze jedna stanice metra.
- Řádek se stanicí musí mít následující formát:

[`pismeno`], [`nazev`], [`km`], [`je konecna`], [`je prestupni`], [`prestupni pismeno`], kde:

`[pismeno]`: Písmeno linky, na které se daná stanice nachází.
`[nazev]`: Název stanice.
`[km]`: Na kolikátém kilometru od počáteční stanice se stanice nachází. Počáteční stanice má 0. kilometr. Číslo může být i s desetinnou tečkou.
`[je konecna]`: Zda je stanice konečná. 0 = ne, 1 = ano. Tento údaj

není povinný (defaultně je 0)
`[je prestupni]`: Zda je stanice přestupní. 0 = ne, 1 = ano. Tento údaj není povinný (defaultně je 0)
`[prestupni pismeno]`: Písmeno linky, na kterou lze z dané stanice přestoupit. Tento údaj není povinný, pokud je předchozí údaj 0.
Řádky mohou tedy vypadat například takto:

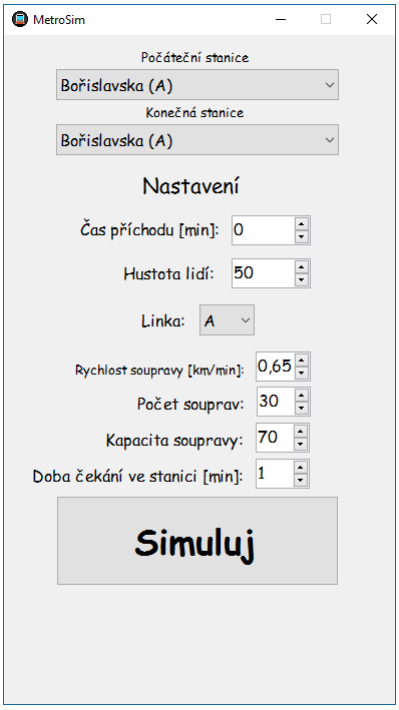
```
C, Háje, 0, 1
B, Českomoravská, 6.4
B, Florenc, 10.9, 0, 1, C
```

Poté již stačí program spustit a vybrat počáteční a konečnou stanici. Uživatel dále může měnit více specifická nastavení:

- „Čas příchodu“: Kdy dorazíme do počáteční stanice od prvního výjezdu souprav. První soupravy totiž vyjedou v čase 0 z konečných stanic. Zbylé soupravy vyjíždějí po 5 minutových intervalech.
- „Hustota lidí“: Kolik lidí příjde každou minutu do metra (každý do náhodné stanice).
- „Linka“: Výběr linky, pro kterou chceme měnit další nastavení.
- „Rychost soupravy“: Rychlost souprav na dané lince v kilometrech za minutu.
- „Počet souprav“: Počet souprav na dané lince. Počet souprav je sudý, jelikož soupravy vyrážejí po párech z konečných stanic.
- „Kapacita soupravy“: Kolik lidí se vejde do jedné soupravy.
- „Doba čekání ve stanici“: Jak dlouho souprava čeká ve stanici v minutách.

Hodnoty „Hustota lidí“ a „Kapacita soupravy“ by měly být upraveny oproti reálným hodnotám. Například v pražském metru ve špičce příjde do metra cca 1200 lidí za minutu. Každá souprava má kapacitu zhruba 1400. Základní nastavení je odvozeno od těchto údajů. Nechceme simulovat všech 1200 různých lidí za minutu ale stačí nám zhruba $\frac{1}{20}$, tedy „Hustota lidí“ = 50 (lehce pod špičkou) a „Kapacita soupravy“ = 70.

Po nastavení stačí stisknout tlačítko „Simuluj“ a počkat na výsledek, který se objeví pod tlačítkem. Uživatel po skončení jedné simulace může změnit nastavení a simulaci opakovat opětovným stiskem tlačítka.



Obrázek 1: GUI programu

3 Jak program funguje (algoritmus)

Program je vlastně obyčejná diskrétní simulace, tudíž je algoritmus velice jednoduchý na vysvětlení. Zhruba funguje takto:

- Simulujeme jízdu souprav mezi stanicemi. Soupravy vyrážejí z konečných stanic po párech na každé lince v čase 0. Pokud je na lince více souprav (což je obvyklý případ), tak další pár vyráží po 5 minutách.
- Každou minutu nagenerujeme „Hustota lidí“ počet lidí, kteří pojedou z náhodné stanice A do náhodné stanice B.
- V čase „Čas příchodu“ vytvoříme hlavního pasažéra, který bude absolvovat trasu dle nastavení.
- Každý pasažér se snaží co nejkratší cestou dostat do své cílové stanice. Cestu má určenou tímto algoritmem:
    1. Pokud je cílová stanice na stejné lince jako aktuální stanice, kde se pasažér nachází, tak pasažér nastoupí do soupravy, která jede správným směrem, a nevystoupí, dokud nedorazí do cíle.

2. Pokud neplatí 1., tak pasažér dojede do nejbližší přestupní stanice na danou linku, na které se nachází cílová stanice. Poté pokračuje krokem 1.

- Simulace končí, když hlavní pasažér dorazí do cílové stanice.

Jelikož je generování ostatních pasažérů náhodné (odkud kam jedou), tak se výsledky různých simulací mohou výrazně lišit. Proto program provede v základním nastavení 16 simulací se stejným nastavením a zprůměruje výsledek.

4 Implementace

Skoro každá diskrétní simulace se skládá ze 4 hlavních tříd (`Udalost`, `Kalendar`, `Model`, `Proces`). Postupně si je všechny probereme.

4.1 Třída `Udalost`

Tato třída v sobě neobsahuje žádnou logiku programu. Obsahuje v sobě pouze informace o dané události v simulaci. Tyto infomace jsou uloženy ve 3 promměných:

- `int kdy`: Kdy se daná událost stane.
- `Proces kdo`: Který proces tuto událost zpracuje.
- `TypUdalosti co`: Identifikátor typu události. `TypUdalosti` je `enum`.

V programu se vyskytují tyto typy událostí (obsah `TypUdalosti`):

- `prichodDoStanice`: Tato událost je zpracována třídou `Pasazer` (o ní později). Označuje událost, když pasažér příjde do stanice metra.
- `prijezdDoStanice`: Tato událost je zpracována třídou `Souprava` (o ní později). Označuje událost, když souprava přijede do stanice metra.
- `vyjezdZeStanice`: Tato událost je zpracována třídou `Souprava` (o ní později). Označuje událost, když souprava odjiždí ze stanice metra.
- `spawnSouprav`: Tato událost je zpracována třídou `Spawner` (o ní později). Označuje událost, kdy vyráží nové soupravy metra z konečných stanic.

4.2 Třída `Kalendar`

Třída `Kalendar` je vcelku malá jednoduchá třída. Stará se pouze o uchovávání výše zmíněných událostí a slouží jako interface pro práci s tímto seznamem.

Seznam událostí je realizován třídou `List<Udalost>`. Pro manipulaci s kalendářem existují metody `void pridejUdalost(Udalost udalost)`, `Boolean jePrazdny()` a `Udalost vratNejaktulanejsi()`. Co tyto metody dělají, snad plyne z názvu. Jediná otázka může nastat ohledně toho, co je to „nejaktuálnější“ událost. Nejaktuálnější událost je ta, která má nejnižší hodnotu `cas`. Pokud je více takových, tak se vybere ta s nejnižším indexem v seznamu.

4.3 Třída `Proces`

Tato abstraktní třída je předkem všech tříd, které zpracovávají události. Obsahuje pouze 2 veřejné proměnné:

- `string id`: identifikátor procesu
- `Model model`: odkaz na třídu `Model` (o ní později).

Poslední součástí této třídy je abstraktní metoda `void zpracuj(Udalost udalost)`.

V programu se používají pouze 3 třídy, jejichž předkem je třída `Proces`.

4.3.1 Třída `Souprava`

Jak již plyne z názvu třídy, toto je třída pro soupravu metra. Obsahuje informace o rychlosti, kapacitě, lince, aktuální stanici apod. dané soupravy. Obsahuje metody:

- `void jedDoDalsiStanice()`: Tato metoda spočítá dobu jízdy do následující stanice a vytvoří novou událost `prijezdDoStanice`.
- `void vystupovani()`: Tato metoda naplánuje událost `prichodDoStanice` příslušným pasažérům.
- `void nastupovani()`: Tato metoda přidá do seznamu příslušné pasažéry, pokud je volné místo.
- `void zpracuj(Udalost udalost)`: Tato metoda zpracovává 2 typy událostí:
  - `prijedzDoStanice`: Zavolá metodu `void vystupovani()` a naplánuje událost `vyjezdZeStanice`.

– `vyjezdZeStanice`: Zavolá metody `void nastupovani()` a `void jedDoDalsiStanice()`.

4.3.2 Třída `Pasazer`

Asi druhá nejhlavnější třída (po třídě `Model`) v celém programu. Obsahuje informace a metody o příslušném pasažérovi (například aktuální stanici). Důležité metody jsou 2:

- `void setPristiStanice()`: Pomocí výše zmíněného algoritmu na-staví pasažérovi jeho příští stanici.
- `void zpracuj(Udalost udalost)`: Tato metoda zpracovává pouze jednu událost `prichodDoStanice`. Zjistí, jestli aktuální stanice je konečná. Pokud ano, tak smaže pasažéra, pokud ne, tak zavolá metodu `void setPristiStanice()`. Poté ještě zjistí, zda se pasažér nachází na přestupní stanici. Pokud ano, tak naplánuje novou událost `prichodDoStanice` do druhé přestupní stanice za určitou dobu, která je určena parametrem `int DOBA_PRESTUPU` ve třídě `StaniceLoader` (o ní později). Pokud ne, tak zavolá metodu `void zaradNaNastupiste(Pasazer p, Stanice pristiStanice)`, která je metodou třídy `Stanice` (také později).

4.3.3 Třída `SpawnerSouprav`

Tato třída je vytvořená pouze z důvodů jednoduché implementace spawnování (vytváření) nových souprav. Jen implementuje metodu `void zpracuj(Udalost udalost)`, která zpracovává událost `spawnSouprav`, při níž volá metodu `void spawniCastSouprav()` třídy `Model`.

4.4 Třída `Model`

Jak již bylo řeceno, toto je nejdůležitější třída v celém v programu. Jejím účelem je uchování odkazů na ostatní třídy a celkový „management“ simulace. V celém programu se pro výpočet používá několik instancí této třídy. Pro další výpočty se tato třída neničí, ale pouze se resetuje pomocí níže zmíněných metod.

Důležité proměnné:

- `SeznamStanic seznamStanic`: Odkaz na třídu `SeznamStanic` (později), ve které jsou uloženy všechny stanice metra načtené ze souboru `stanice.txt`.
- `SortedList<int, Pasazer> seznamPasazeru`: Seznam všech pasažérů v simulaci.

- `SortedList<string, Souprava> seznamSouprav`: Seznam všech souprav v simulaci
- `Kalendar kalendar`: Odkaz na třídu `Kalendar`.
- `Nastaveni nastaveni`: Odkaz na třídu `Nastaveni`, která uchovává informace o daném nastavení modelu zadaném v GUI.
- `MainGUI gui`: Odkaz na třídu `MainGUI`, která se stará o uživatelské rozhraní programu (o ní později).
- `float cas`: Aktuální čas celé simulace.

Důležité metody:

- `void reset()`: Tato metoda připraví model na novou simulaci. Například vytvoří nový kalendář, seznam pasažérů, seznam souprav a vyresetuje všechny stanice.
- `void nactiNastaveni(Nastaveni n)`: Tato metoda načte nastavení z GUI, vytvoří hlavního pasažéra a zavolá metody `void naplanujSpawnSouprav()` a `void spawniOstatniPasazery()`.
- `Pasazer vygenerujPasazera(int casPrichodu)`: Tato metoda vygeneruje pasažéra metra, který jede z náhodné stanice do jiné náhodné stanice.
- `void spawniOstatniPasazery()`: Tato metoda spočítá počet pasažérů pro dané časové rozmezí určené konstantou `int SPAWN_LIDI_MEZICAS = 30`, které je potřeba vygenerovat. Pro každého zavolá metodu `Pasazer vygenerujPasazera(float casPrichodu)`, kde `casPrichodu` je náhodný čas v daném časovém rozmezí. Stručně řečeno, každých 30 minut naplánuje událost `prichodDoStanice` pro určitý počet nových pasažérů určených nastavením „Hustota lidí“.
- `void spawniCastSouprav()`: Jak již bylo řečeno, tato metoda je volána při zpracování událost `spawnSouprav`. Tato metoda pro každou linku vytvoří nový pár souprav (každá jede z jiné konečné stanice).
- `void spocitej()`: Tato metoda se stará o vybírání událostí z kalendáře a volá jejich zpracování, díky tomu aktualizuje proměnnou `cas`. Také každých `int SPAWN_LIDI_MEZICAS = 30` volá metodu `void spawniOstatniPasazery()`. Všechno probíhá ve while loopu, dokud není kalendář prázný nebo hlavní pasažér nedorazil do cíle. Po skončení while loopu zavolá metodu `void finished(float vysledek)` třídy `MainGUI`.

4.5 Ostatní třídy

4.5.1 Třída `MainGUI`

Tato třída obsahuje logiku celého uživatelského rozhraní a podle toho aktualizuje nastavení modelu. Dále se stará o simulaci více instancí třídy `Model` najednou. Jak již bylo řečeno, aby výsledky simulace byly přesnější, provádí se více simulací při stejném nastavení, z důvodů náhodného generování ostatních pasažérů.

Počet celkových simulací je určen konstantou `int POCET_MODELU_CELKEM` a konstanta `int POCET_MODELU_PAR` určuje, kolik simulací beží paralelně. V základním nastavení je `int POCET_MODELU_CELKEM = 16` a `int POCET_MODELU_PAR = 4`. Volání simulací probíhá tak, že v jednu chvíli se simuluje `int POCET_MODELU_PAR` modelů najednou (každý v jiném vlákně). Počká se, až všechny z `int POCET_MODELU_PAR` nejsou dokonečné, a poté se volá další série. O toto se starají metody `void spustSeriiVypoctu()` a již výše zmíněná `void finished(float vysledek)`. Tyto konstanty lze samozřejmě libovolně měnit, akorát je potřeba dbát na jejich dopad, co se týče doby výpočtu. Zřejmě hodnota `int POCET_MODELU_PAR` musí dělit hodnotu `int POCET_MODELU_CELKEM`.

4.5.2 Třídy `Nastaveni` a `NastaveniLinky`

Třída `Nastaveni` v sobě obsahuje seznam tříd `NastaveniLinky`. Samotná třída `Nastaveni` obsahuje údaje o nastavení společném pro celý model, naopak třída `NastaveniLinky` obsahuje pro každou linku její speciální nastavení určené uživatelem.

4.5.3 Třida `Stanice`

Tato třída obsahuje informace o dané stanici a jejím okolí. Například její název, kilometr, na kterém se nachází.

Důležité promměné:

- `Queue<Pasazer> nastupisteVice`: Obsahuje seznam lidí na nástupišti. „Vice“ určuje směr nástupiště ve smyslu toho, že ty stanice v tomto směru mají větší vzdálenost od konečné stanice s kilometrem 0.
- `Queue<Pasazer> nastupisteMene`: To samé jako proměnná výše, akorát opačný směr.
- `List<Stanice> sousedi`: Obsahuje seznam sousedících stanic.

Důležité metody:

- `void najdiSousedy(SortedList<string, Stanice> seznamStanic)`: V seznamu stanic najde sousední stanice a uloží je do `List<Stanice> sousedi`.
- `void zaradNaNastupiste(Pasazer p, Stanice pristi)`: Daného pasažéra zařadí na příslušné nástupiště podle toho, kam chce jet.

4.5.4 Třida `SeznamStanic`

Tato třída obsahuje seznam stanic pro model, ale také i speciální seznamy, jako seznam konečných stanic. Je to taková pomocná třída, která obsahuje informace o všech stanicích metra.

4.5.5 Třida `StaniceLoader`

Tato třída slouží jako parser souboru `stanice.txt` a vytváří z něj seznam stanic, který se poté ukládá do třídy `SeznamStanic`. Zde se nachází konstanta `int DOBA_PRESTUPU`, která je základně nastavena na hodnotu 2.

4.5.6 Třida `CustomCBItem`

Tato třída je vytvořena pro uložení více údajů v itemu třídy `ComboBox`, která se používá v uživatelském rozhraní pro výběr stanic.

5 Výsledky simulací

Výsledky simulací jsou po zadaní správných parametrů velice blízké realitě.

Například při měření doby cesty ze stanice Českomoravská do stanice Křižíkova v čase příchodu nastaveném na 500 min, je doba cesty spočítána na 7,4 minuty. Což odpovídá reálné hodnotě ze stránky `https://jizdnirady.idnes.cz`, dle které cesta trvá zhruba 5-6 minut, a to se nepočítá čekání ve stanici na soupravu metra.

Nebo cesta ze stanice Malostranská do stanice Hloubětín by měla trvat podle simulace 29,9 minut. A dle IDOSu má trvat 29 minut.

6 Závěr

Výsledky simulací se někdy výrazně liší kvůli náhodnému generování lidí. Dále kvůli tomu, že se například neberou v potaz různé změny počtu souprav,

jejich rychlosti v různých částech dne. Na druhou stranu, pokud jsou správně navrženy vstupní hodnoty, tak výsledky simulací odpovídají realitě.

7 Zdroje

[1] *IDOS.cz*. URL: https://jizdnirady.idnes.cz/pid/spojeni/hhh.