Semestrálny projekt z predmetu Programovanie(3)

Dávid Števaňák

January 2023

Úvod

Cieľom projektu bolo naprogramovať zjednodušenú verziu dopravnej siete mesta. Využili sme nasledovné predpoklady:

- Každá zastávka má jedinečný názov (začínajúci veľkým alebo malým písmenom) a príslušný index
- Každá linka má jedinečné číslo
- Linka môže byť len typu `BUS, TRAM` alebo `METRO`
- Linka nemusí nutne byť okružná, t.j. môže ísť len jedným smerom
- Každá linka jazdí pravidelne v dvoch intervaloch - počas pracovných dní a počas víkendov začínajúc o polnoci (00:00). Abstrahovali sme od rôznych časov v nočných a denných hodinách, zároveň sme ale povolili autobusom jazdiť aj v takom intervale, ktorý nedelí 60 (napríklad 22) a prípadná nezrovnalosť pri prechode na nasledujúci deň je ošetrená podmienkou, že každý autobus začína o polnoci (teda ak by posledný spoj vyrazil o 23:59 nasledujúci vyrazí o polnoci bez ohľadu na interval).

1 Pomocné triedy

Definujeme dve pomocné triedy:

- trieda `Time` slúži na ukladanie a jednoduché pracovanie s inštanciami času, využíva pritom dve premenné typu `int`: `hours, minutes`. Metódy sú popísané nižšie.
- trieda `Exception` je jednoduchá trieda na vracanie výnimiek v projekte a obsahuje len jednu premennú `std::string mess` ktorá reprezentuje chybovú správu danej vyvolanej výnimky. Jej konštruktor berie ako parameter `std::string message` a presunie ju to premennej `mess`. Druhá metóda vracia premennú `mess`.

```
Time(): hours(0), minutes(0){};
    //konstruktor bez parametrov nainicializuje premenne na 0

bool Time::setTime(int hours0, int minutes0);
    //nastavi cas pre pre hodnoty hours v intervale <0,24) a minutes <0,60) a vrati
    true inak nastavi obe na 0 a vrati false

Time(int hours0, int minutes0);
    //s vyuzitim funckie setTime nastavi hodnoty

Time::getTime();
    //vrati std::string reprezentaciu casu vo formate <hours>:<minutes> vyplnene
    nulami (t.j. 03:34, 01:03)

Time::getTimePair();
    //vrati std::pair obsahujuci hodiny a minuty

//scitavanie a odcitavanie casu s maximom 23:59 a minimom 00:00 (t.j. 23:55 + 00:10 =
     00:05 a podobne 00:05 - 00:10 = 23:55)
//porovnavanie funguje intuitivne
```

2 Trieda BusStop

Táto trieda slúži na uchovávanie údajov o jednotlivých zastávkach. Obsahuje členské premenné

- `int number` - číslo zastávky
- `std::string name` - názov zastávky
- `std::vectorint lines`, ktorá uchováva všetky linky, ktoré prechádzajú danou zastávkou (reprezentované ako `int`).

Jej metódy:

```
BusStop(): number(-1), name("");
    //vytvori prazdnu zastavku so zapornym cislom ako indikator nenastavenej zastavky

BusStop(const std::string& stop_name, int num);
    //skontroluje stop_name podla nasledovnych kriterii
    // - nemoze byt prazdne
    // - musi zacinat velkym alebo malym pismenom
    //ak nesplna, vyhodi vynimku s chybovou spravou "Wrong bus stop name - must be of non-empty length and start with a letter"
    //skontroluje, ci cislo zastavky je vacsie ako 0 - ak nie vyhodi vynimku "Wrong bus stop number"
    // ak splna vyssie podmienky, nastavi premenne

bool BusStop::addLine(int line_num);
    //ak line_num je mensie rovne ako 0 tak vyhodi vynimku so spravou "Incorrect line number";
    // ak sa line_num nenachadza v std::vector<int> lines tak ju prida na koniec a vrati true; ak sa tam nachadza nic nespravi a vrati false

bool BusStop::removeLine(int line_num);
    // skontroluje ci je std::vector<int> lines neprazdny - ak je prazdny vrati vynimku so spravou "No line to delete"
    // inak skontroluje ci je line_num vo vectore - ak ano vymaze ju a vrati true inak nespravi nic a vrati false

bool BusStop::changeName(const std::string &new_name);
    //zmeni meno zastavky - podla rovnakych podmienok ako v konstruktore, ak zle tak vracia rovnaku vynimku ako v konstruktore

bool BusStop::changeStopNum(int new_num);
    //zmeni cislo zastavky - rovnako ako pri nastavovani v  konstruktore

std::string BusStop::getBSlines() const;
    //vrati string reprezentaciu jednotlivych krizujucich liniek v tvare <line_num>,<line_num>,...

int BusStop::numberOfLines() const {return lines.size();}
    //vrati pocet prechadzajucich liniek;

std::string BusStop::getName() const { return name; };
    //vrati nazov zastavky

int BusStop::getStopNumber() const {return number; };
    //vrati cislo zastavky

std::vector<int> BusStop::getCrossingLines(){return lines;};
    //vrati std::vector<int> prechadzajucich liniek
```

3 Trieda BusLine

Trieda `BusLine` slúži na uchovávanie informácii o jednotlivých linkách. Používa pritom nasledovné premenné:

- `std::vector<std::pair<BusStop*, int» stops` slúži na uchovávanie zastávok v linke v poradí, v akom linka premáva. Druhým parametrom v `std::pair` je `int`, ktorý reprezentuje vzdialenosť v minútach od predošlej zastávky.

- `int interval_workdays, interval_weekends` reprezentujú minútový interval, v ktorom linka premáva počas pracovných dní alebo víkendov
- `int line_num` je jedinečné číslo linky
- `bool status` symbolizuje, či je linka v prevádzke alebo nie
- `Type type` hovorí o type vozidla - jeden z množiny `{BUS, TRAM, METRO}`
- `direction` hovorí, akým smerom premáva linka (-1 znamená smer <- a teda počiatočná zastávka je na konci vektora, 0 znamená <->, čiže obojsmerne a 1 reprezentuje ->, čiže v smere vektora. Bližsie vysvetlené pri metóde `changeDirection`.)

Jej metódy rozdelíme do dvoch častí

3.1 Prípravné metódy

```
BusLine::BusLine():line_num(-1), direction(-2){};

BusLine::BusLine(int number, Type type1, int direct);
    //ak number je cislo vacsie ako 0 a direct je jedno z cisiel -1,0,1 tak nastavi tieto premenne
    // pri nespravnom number vyhodi vynimku "Incorrect line number";
    // pri nespravnom direct vyhodi chybu "Unknown direction - use values -1,0,1 only!"

BusLine::BusLine(int number, Type type1, int workdays_interval, int weekends_interval, int direct);
    //pre number a direct platia rovnake podmienky a spravanie ako pri predoslom konstruktore
    // pre oba intervaly plati, ze musia byt z rozmedzia minimalnych a maximalnych intervalov definovanych ako const int v side_classes.h, inak vyhodi Exception so spravou bud OutsideWeekendInterval alebo OutsideWorkdayInterval
    // inak nastavi jednotlive premenne

bool BusLine::changeDirection(int new_direct);
    // funkcia dostava ako input new_direct z {-1,0,1} - osetri, ak nepatri vyhodi Exception so spravou WrongDirection definovane v side_classes.h
    // ak je new_direct rovnaka ako aktualna tak len vrati false
    // inak ak je ina tak skontroluje ci je 0 - v tom pripade len prepise jej hodnotu
    // ak je new_direct v {-1,1} tak potom reverzne vektor zastavok v linke (teda prehodi jeho poradie zastavok), zapise new_direct to direction a vrati true

int BusLine::getDirection();
    //vrati aktualny smer

bool BusLine::isLineInOrder();
    // vrati, ci je linka v prevadzke

std::string BusLine::getStopsString();
    // vrati string reprezentaciu zastavok v nasledovnom tvare
    // - ak je smer 0 bude pouzivat znak <-minutes->
    // - inak pouziva znak -minutes->
    // vysledny string je v tvare <stopName> (<)-minutes-> <stopName> ...

std::vector<std::pair<BusStop*,int>> &BusLine::getStopVector() { return stops;};
    //vrati referenciu na vektor obsahujuci smernik na zastavky a ich vzdialenosti v minutach

bool BusLine::changeStatus();
    // zmeni aktualny status na true/false len ak su splnene nasledovne podmienky:
    // - cislo linky je > 0, oba intervaly su > 0
    // a vrati true
    // ak nie je splnena nejaka z podmienok tak povoli zmenit status na false a vrati true
    inak len vrati false

bool BusLine::setIntervalWeekends(int wknd_int);
bool BusLine::setIntervalWorkdays(int work_int);
//nastavia interval podla podmienok ako v konstruktore s rovnakym spravanim

bool BusLine::changeLineNum(int new_line_num);
    //zmeni cislo linky s rovnakym spravanim ak ov konstruktore
```

```
47
48 std::string BusLine::getTimetable() const;
49     //vracia string rozvrhu prichodov a odchodov linky vo formate
50     //Pondelok az Piatok
51     //<hodina> | <minuta> <minuta> ...
52     //...
53     //Vikendy
54     //<hodina> | <minuta> <minuta> ...
55     //...
56     // pricom hodina <0,24); a minuta <0,60)
57
58 void BusLine::timetableToFile(const std::string& file) const;
59     //zapise vystup z getTimetable() do suboru
60
61 int getLineNum() const {return line_num;};
62
63 Type getLineType() const {return type;};
64
65 int getIntervalWorkdays() const {return interval_workdays;};
66
67 int getIntervalWeekends() const{return interval_weekends;};
68
69 BusStop* getLastStop() const {return stops.back().first;};
70     //vrati smernik na poslednu zastavku linky
71
72 BusStop* getFirstStop() const {return stops.front().first;};
73     //smernik na prvu zastavku linky
```

3.2 Pokračovanie

Nasledovné prvé 2 metódy vkladajú alebo vymazávajú zastávku z linky, pričom prepisujú jej vzdialenosti podľa daných parametrov. Posledné dve metódy slúžia na vyhľadanie najskoršieho odchodu linky z danej počiatočnej zastávky do danej cieľovej zastávky pre zadaný čas.

```
1
2 bool BusLine::addStop(int position, BusStop &new_stop, int mins_from_prev, int
      mins_to_next);
3     //dostane position, na ktore miesto zastavka new_stop patri a tiez prislusne
      vzdialenosti, od predchadzajucej a nesldujucej zastavky
4     //posisiton musi byt v rozmedzi <-1,stops.size()> a minuty musia splnat podmienky
       ako v konstruktore, taktiez zastavka nemoze byt nainicializovana na nespravne
      udaje(-1, prazdny nazov atd)- ak nie tak funkcia skonci  a vrati false
5     // ak je position -1 alebo stops.size() tak prida zastavku na koniec vektora -
      vyuziva teda iba mins_from_prev, mins_to_next su irelevantne kedze zastavka je na
       konci
6     // ak je position 0 tak zastavku prida na zaciatok - mins_from_prev su teda
      irelevantne lebo new_stop je prva
7     // inak ak je position niekde medzi (0, stops.size()) tak ju vlozi na danu
      poziciu a prida vzdialenosti v minutach mins_from_prev a mins_to_next (napr.
      new_stop, mins_from_prev = 4, mins_to_next = 5, position = 1 a stops je vector
      Stop1 -2-> Stop2 tak zmeni na Stop1 -4-> new_stop -5-> Stop2)
8     //a vrati true
9
10 bool BusLine::removeStop(BusStop &stop_rem, int mins_prev_to_next);
11     //vymaze zastavku z vektora zastavok a nahradi interval medzi tymito zastavkami
      hodnotou mins_prev_to_next
12     //ak stops je prazdny vektor alebo zastavka nie je vo vektore vrati false
13     //ak je stop_rem na zaciatku alebo na konci tak ju len vymaze a vrati true
14     // inak ju vymaze z vektora a vzdialnost predoslej a nasledovnej zastavky nastavi
       podla parametra mins_prev_to_next - ak ma hodnotu <= 0 tak to znamena ze ma len
      scitat aktualne vzdialenosti (cize Stop1 -4-> stop_rem -3-> Stop2 sa zmeni na
      Stop1 -7-> Stop2) a vrati true
15     //inak vymaze zastavku a zmeni hodnotu podla mins_prev_to_next (mins_prev_to_next
       = 15, Stop1 -4-> stop_rem -3-> Stop2 sa zmeni na Stop1 -15-> Stop2) a vrati true
16
17 std::vector<std::pair<BusStop, Time>> BusLine::getEarliestFromStop(BusStop &start,
      BusStop &dest, Time &time, bool weekend=false);
18     //funkcia najde najskorsi spoj na linke medzi dvomi zastavkami pre zadany cas
19     // ak linka nie je v prevadzke vyhodi prislusnu vynimku "Line <number> currently
      not in order"
20     //ak jedna z dvojice start, dest nie je v linke vyhodi vynimku so spravou "Given
      {destination, start point} is not in Line number <number>"
21     //inak funkcia vrati vektor dvojic zastavka a cas medzi nimi
22
```

```
std::string BusLine::getEarliestFromStopString(BusStop &stop,BusStop &dest, Time &
    time, bool weekend);
    //funckia vrati string reprezentaciu funckie getEarliestFromStop vo formate
    //(<time>)<stopName> -mins_to_get-> (<time>)<stopName> -mins_to_get-> ...
```

4 Trieda PTNetwork

Posledná trieda slúži na uchovávanie informácií o celej dopravnej sieti (ďalej len DS) - to robí pomocou dvoch premenných:

- `std::map<int, BusStop> busstops` obsahuje všetky zastávky v DS
- `std::map<int, BusLine> buslines` obsahuje všetky linky v DS

a taktiež na vyhľadávanie spojov medzi dvomi zastávkami.
Aj tu si rozdelíme metódy do dvoch častí

4.1 Manipulácia s inštanciami triedy PTNetwork

V tejto časti sú najmä funkcie manipulujúce s jednotlivými premennými, pridávanie zastávok a liniek ale aj funkcie na načítanie DS zo súboru a zapisovanie do súboru s rovnakým formátom. Zároveň máme aj metódy, ktoré vracajú a zapisujú cestovný poriadok pre jednotlivé zastávky ale aj také, ktoré vracajú usporiadané najbližšie odchody liniek zo zadanej zastávky v zadanom intervale.

```
PTNetwork(): busstops(*new std::map<int, BusStop>) , buslines(*new std::map<int,
    BusLine>){};
PTNetwork(std::map<int,BusStop> stops, std::map<int, BusLine> lines, std::string ) :
    busstops(std::move(stops)), buslines(std::move(lines)) {};
//konstruktory

void PTNetwork::clearAll();
    //vymaze mapy busstops a buslines

~PTNetwork();
    //to co clearAll();

BusStop& PTNetwork::getBusStopById(int id);
    //vrati referenciu na zastavku v DS ak sa v nej nachadza inak vyhodi vynimku so
    spravou "No such stop ID <stopNum> in network"

BusLine& PTNetwork::getBusLineByNum(int linenum);
    //vrati referenciu na linku v DS
    // ak linenum < 0 vyhodi vynimku so spravou "Bus line id out of range"
    // ak taka linka nie je vyhodi vynimku so spravou "No such line with number <
    lineNum> in network"

int PTNetwork::getNumberOfLines() const { return buslines.size();};
int PTNetwork::getNumberOfStops() const { return busstops.size();};

void PTNetwork::readStopsAndLines(const std::string& file);
    // precita subor so zastavkami a spojmi a nacita ho do jednotlivych premennych
    //subor musi byt v nasledovnom formate:
    //<StopNo>;<StopName>
    //...
    //~empty line~
    //<LineNumber>;<IntervalWorkdays>;<IntervalWeekends>;<IsInOrder>;<Direction(-1
    <-, 0 <->, 1 ->)>
    //<StartStopNumber> -> <TimeToGet> -> <StopNumber> ... <EndStopNumber>
    //~empty line~
    // ...
    // ak maju dve zastavky rovnake cislo tak vrati Exception "Multiple Bus Stop
    number definition in file <filename> on stop number <stopNum>" podobne aj pre
    rovnaky nazov vrati  "Multiple Bus Stop name definition in file <file> on stop <
    stopName>"
    //ak je nespravny type v niektorych z liniek vyhodi vynimku "Bus Line <type> must
     be within range 0-2 representing Bus, Tram, Metro respectively"
    //ak parametre niktorej z liniek nie su spravne vyhodi Exception "Bus Line
    parameters not formatted correctly"
    // ak sa v linkach nachadza BusStop ktora nebola medzi vymenovanymi zastavkami
    vrati chybu "Bus Stop number <StopNo> not among list of bus stops"
    // ak maju niekotre linky rovnake number tak vrati chybu "Multiple Bus Line
    number definition in file <file> on Bus Line number <lineNum>"
```

```
37
38
39 void PTNetwork::writeStopsAndLines(const std::string &file);
40     //zapise DS do suboru v rovnakom formate ako ho cita vo funkcii readStopAndLines
       - POZOR nemusi byt rovnaky subor zapisany a precitany lebo poradie liniek nie je
       striktne dane
41
42 std::string PTNetwork::getTimeTableForStop(BusStop &stop);
43     //vrati std::string reprezentujuci rozvrh pre danu zastavku - pre vsetky
       prechadzajuce linky vypise ich trasu a casy odchodov zo zastavky
44     //ak zastavka nie je v DS vrati chybu "Given bus stop <stopName> not in this
       network"
45     //ak niektora linka nie je v prevadzke vo vystupe to vyzera:
46     //==========-<LineNum>-SMER-<lastStop>-==============
47     //  <stop1> -minutes-> <stop2> ....
48     //  Linka momentalne mimo prevadzky
49     //pre ostatne vypisuje rovnako trasu a pod nou
50     //  Pondelok az piatok:
51     //  <hour> | <minute> <minute> ....
52     //  ...
53     //  Vikendy:
54     //  <hour> | <minute> <minute> ...
55     //  ...
56     //...
57
58 void PTNetwork::writeTimeTableForStop(std::string filename, BusStop &stop);
59     //zapise vystup getTimeTableForStop do daneho suboru
60
61 std::vector<std::tuple<Time,int, BusStop>> PTNetwork::getClosestFromStop(BusStop &
       stop, Time time1, Time interval, bool weekend);
62     //vrati usporiadany vektor (od najmensieho casu cakania po najvacsi) casov, linky
        a konecnej zastavky linky pre odchadzajuce linky zo zastavky ktore su v zadanom
       intervale (teda tie linky ktore odchadzaju zo zastavky skor ako time1 + interval)
63
64 std::string PTNetwork::getClosestFromStopString(BusStop &stop, Time time1, Time
       interval, bool weekend);
65     //vrati string reprezentaciu funkcie getClosestFromStop vo formate:
66     //"<lineNum>: o <untilDepartueTIme> minut (odchod-><departureTime>), smer: <
       endStop>\n<lineNum>: o <untilDepartueTIme> minut (odchod-><departureTime>), smer:
        <endStop>\n..."
67     //napr. "29: o 00:14 minut (odchod->00:16), smer: Cintorin Slavicie\n4: o 00:26
       minut (odchod->00:28), smer: Zochova\n"
68
69 int PTNetwork::addStop(BusStop *new_stop);
70     //prida zastavku do siete - ak je uz v sieti taka zastavka s menom vrati jej
       aktualny index/cislo
71     //potom ak je index unikatny tak ju len prida ak ale koliduje s indexom inej
       zastavky tak ho prepise na najvyssi index + 1 a prida zastavku
72
73 bool PTNetwork::addLine(BusLine &new_line);
74     //prida linku do siete (teda aj vsetky jej zastavky - pomocou addStop), ak jej
       cislo nekoliduje s inou zastavkou inak vrati vynimku
75     //"Bus line number <lineNum> already exists with route: <collidingLineRoute>"
```

4.2 Pokračovanie - vyhľadavánie najkratšej cesty

Posledná časť sa zameriava na vyhľadávanie najkratšej trasy (vzhľadom na čas) použitím Dijkstrovho algoritmu na hľadanie "najlacnejšej cesty". Ako reprezentáciu DS sme zvolili `std::map` kde kľúčom je číslo zastávky a hodnota je `std::tuple<int, int,int>` ktorá reprezentuje susednú zastávku (jej číslo), čas, za aký sa do nej dostaneme zo zastávky v kľúči a nakoniec linku, po ktorej do zastávky ideme.

```
1 std::vector<std::tuple<int, int, int>> PTNetwork::whereNext(BusStop &from);
2     //ak je zastavka v DS tak vrati vektor trojic - susedne zastavky, vzdialenost (
      minuty) a spajajuca linka medzi nimi
3     //ak nejaka linka nie je v prevadzke tak sa s nou nepracuje
4     //ak zastavka nie je v DS vrati vynimku "Given stop <stopName> not in network"
5
6  std::map<int, std::vector<std::tuple<int, int, int>>> PTNetwork::createAdjMap();
7     //s vyuzitim whereNext vytvori mapu susednosti kde key je zastavka a value je
      trojica vratena funkciou whereNext
8
9 std::vector<pair<int,int>> PTNetwork::findShortestPath(BusStop &start, BusStop &dest)
      ;
```

```
    //vrati vektor dvojic - prve je zastavka a druha linka na akou sa dostane na nasledujucu zastavku pricom posledna dvojica ma linku = 0
    //ak cesta neexistuje vrati prazdny vektor

std::string PTNetwork::getRoute(BusStop &start, BusStop &end, Time when){
    //vrati string reprezentaciu funkcie findShortestPath
    //ak neexistuje vrati "No path from <startName> to <endName>"
    //"(time length: <totalTime>) <lineNum>(<type>): (<departueTime>)<stopName> -minutes-> (<depTime>)<stopName> -minutes-> ...(<depTime>)<stopName> /-prestup- <lineNume>(<type>) -/ (<departueTime>)<stopName> -minutes-> (<depTime>)<stopName> -minutes-> ...(<depTime>)<stopName> /-prestup- <lineNume> -/ ..."
}
```