Dopravná infraštruktúra je neoddeliteľnou súčasťou hlavného mesta najmä vtedy, ak mesto nemá vybudovanú podzemnú dráhu. MHD slúži všetkým ľudom, najmä tým, ktorí nevlastnia osobné vozidlá, sú zdravotne znevýhodnení, ako aj účastníkom cestovného ruchu.

(nespomínať podzemnú dráhu, druhá veta vyznieva zle a my chceme viac upozorniť na to že MHD je pre všetkých)

Dopravná infraštruktúra je neoddeliteľnou súčasťou mobility mesta. Mestská hromadná doprava slúži všetkým ľudom žijúcim v meste, tým čo dochádzajú do mesta za rôznym cieľom, ako aj účastníkom cestovného ruchu.

Navyše minimalizuje počet vozidiel v meste, čím odbremeňuje mesto od dopravných zápch. Prispieva k ekologickej preprave v meste a v niektorých prípadoch môže byť mestská hromadná doprava najrýchlejším a najekonomickejším spôsobom prepravy cestujúcich.

Aby bolo cestovanie verejnou dopravou komfortné a cestujúci viac motivovaní využívať tento spôsob prepravy, kľúčovú rolu zohráva presnosť spojov, ich vysoká frekvencia, hustota rozmiestnenia zastávok, minimálna nutnosť peších presunov medzi zastávkami, pohodlnosť cestovania, **informovanosť o optimálnych cestách** a mnohé iné faktory.

**Cieľom práce je ~~Študent informatiky je schopný~~** prispieť k zefektívneniu MHD vytvorením aplikácie,

ktorá bude ponúkať čo najpresnejšie vyhľadávanie ciest a okolitých zastávok, aby tak

prispel k celkovému sprehľadneniu mestskej hromadnej dopravy.

Aktuálne sú dostupné mnohé aplikácie, ktoré poskytujú spomínané funkcionality. Okrem možnosti vyhľadávania ciest ponúkajú aj priamu kúpu cestovného lístka, sledovanie pohybu vozidiel v reálnom čase, ako aj informáciu o možnosti prestupu na prímestskú dopravu. Avšak vyhľadávanie je realizované nad statickými cestovnými poriadkami. Dopravný podnik Bratislava disponuje informáciami o aktuálnych polohách vozidiel, z čoho vie následne určiť meškanie jednotlivých spojov. Tieto informácie sú zobrazené na elektronických tabuliach na vybraných zastávkach. ~~Prečo teda nevyužiť tieto dáta na spresnenie vyhľadávania ciest?~~

Rozhodli sme sa vytvoriť aplikáciu, ktorá bude pri vyhľadávaní ciest prihliadať na aktuálnu dopravnú situáciu. Zároveň chceme vytvoriť túto aplikáciu pre široké spektrum cestujúcich.

(nerozmyslela som si to ja, na základne zadania sme sa rozhodli vytvoriť aplikáciu)

Na základe zadania sme sa rozhodli vytvoriť aplikáciu, ktorá bude prihliadať na aktuálnu dopravnú situáciu a bude slúžiť pre široké spektrum cestujúcich.

Vieme, že existujú cestujúci, ktorí preferujú minimálny počet prestupov, iní hľadajú najrýchlejšiu cestu do cieľa a sú aj takí, ktorí potrebujú na svoj presun nízkopodlažné vozidlá. Preto táto aplikácia bude poskytovať aj alternatívne optimálne cesty.

Práca popisuje priebeh vývoja aplikácie a je rozdelená do štyroch kapitol. V prvej kapitole sú zhrnuté poznatky, ktoré sme nadobudli štúdiom zaujímavých vedeckých článkov, ako aj skúmaním funkcií už existujúcich aplikácií na vyhľadávanie ciest v bratislavskej mestskej hromadnej doprave.

V literatúre sa vyhľadávanie ciest často spája s Dijkstrovým algoritmom. Spomíname aj algoritmus A\* spolu s jeho rôznymi optimalizáciami a modifikáciami. Jedna z modifikácií sa dokonca zaoberá časovými dátami a počíta s multimodalitou dopravnej siete. Pri grafových algoritmoch sú často spomínané aj minimalizácie prehľadávaného priestoru, ktoré by mohli zrýchliť beh algoritmu.

Zmieňujeme sa aj o takej literatúre, ktorá by nám mohla pomôcť s vyhľadávaním alternatívnych

ciest. Spomedzi všetkých algoritmov na vyhľadávanie ciest nás napokon najviac zaujal negrafový algoritmus RAPTOR, čo je skratkou pre Round bAsed Public Transit Optimal Router. Ako vyplýva už z názvu, je určený na vyhľadávanie ciest v sieti verejnej dopravy. V tejto kapitole popisujeme jeho základnú verziu, ale aj vylepšenú verziu, ktorá sa zaoberá vyhľadávaním alternatívnych ciest. Výhodou algoritmu je, že je prispôsobiteľný tak, aby spracovával aj dynamické dáta.

(skrátiť, spomenúť grafové-negrafové, že existuje raptor).

Zameriavame sa na literatúru, ktorá spomína vyhľadávanie ciest v hromadnej doprave a ktorá rieši niektoré z problémov: multimodalita dopravy, dynamické dáta alebo alternatívne cesty. Existuje veľa grafových riešení, ale natrafili sme aj na negrafové riešenie vyhľadávania ciest.

Druhou kapitolou je kapitola Návrh, v ktorej vysvetľujeme, ako sme sa rozhodli vytvoriť našu aplikáciu z pohľadu funkcií, vyhľadávacieho algoritmu, dát a architektúry. Popisujeme funkcie, ktoré bude aplikácia cestujúcim ponúkať, pričom sme sa inšpirovali existujúcimi aplikáciami a vlastnými skúsenosťami. V tejto kapitole objasňujeme výber algoritmu na vyhľadávanie ciest. Vybraným algoritmom je RAPTOR algoritmus, ktorý modifikujeme, aby sme dosiahli požadované funkcie aplikácie. Ďalej po podrobnej analýze približujeme návrh spracovania dát a popisujeme dáta statických cestovných poriadkov uložených vo formáte GTFS. Dopravný podnik Bratislava nám poskytol aj dáta o meškaní vozidiel. Opisujeme, ako ukladáme dáta do databázy a do dátovej

štruktúry pre algoritmus. V neposlednom rade popisujeme aj návrh architektúry systému. (spomenúť, že sme získali dynamický pohľad, trochu skrátiť)

Druhou kapitolou je kapitola Návrh, v ktorej vysvetľujeme, ako sme sa rozhodli vytvoriť našu aplikáciu z pohľadu funkcií, vyhľadávacieho algoritmu, dát a architektúry. Objasňujeme výber RAPTOR algoritmu a navrhujeme jeho modifikácie na dosiahnutie požadovaných funkcií. Opisujeme dáta získané od Dopravného podniku Bratislava, ako aj návrh ich ukladania do databázy a dátovej štruktúry pre algoritmus. Navrhujeme ako sa vysporiadať s dynamickými dátami a približujeme návrh klient-server architektúry.

V kapitole Implementácia približujeme technológie použité pri vývoji aplikácie. Spomíname riešenie problémov vzniknutých pri importovaní a spracovávaní statických dát a dát o meškaní. Popisujeme, akým spôsobom je implementovaná dátová štruktúra a ako do nej ukladáme dáta o meškaní vozidiel. Pre správne fungovanie aplikácie bolo potrebné simulovať čas v minulosti, keďže poskytnuté dáta mali platnosť v roku 2018. Ďalej ukazujeme implementáciu niektorých kľúčových častí RAPTOR algoritmu. Na ukážku poskytujeme časti kódu, ktorými sme obohatili algoritmus, aby prihliadal na

preferencie zadané používateľom. Približujeme taktiež problémy, ktoré vznikali pri implementácii

na klientskej strane.

(spomenut techonologie, klient server, time simulator)

V kapitole Implementácia popisujeme technológie použité pri vývoji aplikácie. Čo sa týka serverovej strany, spomíname implementáciu vyhľadávacieho algoritmu, dátovej štruktúry a importovania dát. Ďalej opisujeme ako prebiehal vývoj vybraných častí na klientskej strane. Keďže platnosť cestovných poriadkov a dát o meškaní vozidiel je z minulosti, popíšeme ako sme simulovali čas v aplikácii.

V poslednej kapitole spomíname testovacie dáta, ktoré sme vytvorili na overenie správnosti vyhľadávacieho algoritmu. Na ukážkach vyhľadaných ciest približujeme, ako sa algoritmus postupne vyvíjal a podľa toho sa menil zoznam vyhľadaných ciest.

(Toto predlžiť. Testovanie navrhovanej funkcionality najskôr na testovacej vzorke. Potom na reálnych dátach. Spomenúť že demonštrujem konkrétnymi ukážkami funkcionalitu algoritmu.)

V poslednej kapitole spomíname testovacie dáta, ktoré sme vytvorili na prvotné overenie správnosti algoritmu. Konkrétnymi ukážkami demonštrujeme schopnosť algoritmu vyhľadávať cesty na reálnych dátach od Dopravného podniku Bratislava. Ukážeme, ako algoritmus pri vyhľadávaní prihliada na meškanie vozidiel alebo na zadané používateľské preferencie. V neposlednom rade predvedieme, ako algoritmus hľadá alternatívne optimálne cesty.

Podarilo sa nám navrhnúť a naimplementovať aplikáciu na vyhľadávanie optimálnych ciest v MHD Bratislava. Dominantnou funkciou aplikácie je, že prihliada na aktuálnu dopravnú situáciu a pri vyhľadávaní ciest zohľadňuje aj meškanie vozidiel. Keďže cestujúci majú rôzne preferencie a optimálna cesta môže byť pre každého iná, aplikácia neponúka používateľovi len jednu cestu, ale viacero alternatívnych optimálnych ciest. Používateľ môže zvoliť jeho aktuálnu polohu ako začiatočný bod. V prípade, že používateľ pozná názvy zastávok, môže ich vybrať zo zoznamu ako začiatočnú a cieľovú zastávku, prípadne si ich môže zvoliť z mapy. Okrem iného aplikácia ponúka aj zobrazenie

všetkých liniek a ich trás.

Podarilo sa nám navrhnúť a naimplementovať aplikáciu na vyhľadávanie optimálnych ciest v mestskej hromadnej doprave Bratislava. Dominantnou funkciou aplikácie je, že prihliada na aktuálnu dopravnú situáciu a pri vyhľadávaní ciest zohľadňuje aj meškanie vozidiel. Vyhľadávací algoritmus beží na serverovej strane, pričom operuje na stále aktuálnej dátovej štruktúre.

Keďže cestujúci majú rôzne preferencie a optimálna cesta môže byť pre každého iná, aplikácia neponúka používateľovi len jednu cestu, ale viacero alternatívnych optimálnych ciest.

Pri vyhľadávaní ciest používateľ zadáva požadovaný začiatočný, konečný bod cesty a dátum a čas začiatku vyhľadávania, pričom predvolený je aktuálny dátum a čas. Aplikácia ponúka aj rozšírené vyhľadávacie parametre. Používateľ má teda možnosť nastavenia maximálneho počtu prestupov, minimálneho času na prestup medzi jazdami, maximálneho času pešieho presunu, ako aj vyhľadanie len nízkopodlažných spojov. V prípade, že používateľ upraví rozšírené vyhľadávacie parametre, algoritmus vyhľadá len cesty vyhovujúce zadaným parametrom. Zmenené hodnoty preferencií sa ukladajú do zariadenia a sú použité aj pri nasledujúcich vyhľadávaniach. Používateľ môže vidieť v histórii vyhľadávania 5 posledných záznamov (začiatočná a konečná zastávka).

Aplikácia poskytuje nielen tie optimálne cesty, ktoré začínajú najbližšie od zadaného času a dátumu, ale aj nasledujúce v chronologickom časovom slede. Vždy zobrazí aspoň 5 ciest na jednu stránku. Následne môže používateľ zvoliť voľbu pre vyhľadanie ďalších nasledujúcich ciest.

Používateľ má možnosť zvoliť jeho aktuálnu polohu ako začiatočný bod. Konečný bod môže byť len zastávka. V prípade, že používateľ pozná názvy zastávok, môže ich vybrať zo zoznamu ako začiatočnú a konečnú zastávku. Ak pomenovanie jednotlivých zastávok nepozná, môže si ich zvoliť priamo z mapy. Okrem iného, aplikácia ponúka aj zobrazenie všetkých liniek a ich trás. Trasu linky si vie používateľ zobraziť na mape.

Momentálne má aplikácia v sebe naprogramovaný simulátor času, aby sme mohli vyhľadávať cesty v reálnom čase a prihliadať tak na meškania vozidiel, aj keď máme k dispozícii iba dáta z minulosti.

~~Pred samotným návrhom a vývojom aplikácie sme študovali literatúru, ktorá sa venovala riešeniu rôznych otázok vznikajúcich pri hľadaní ciest vo verejnej doprave. Kľúčovým problémom bol najmä výber algoritmu na vyhľadávanie ciest. Vo fáze štúdia odbornej literatúry sme sa stretli najmä s grafovými riešeniami hľadania ciest. Našli sme aj rôzne modifikácie a optimalizácie týchto algoritmov. My sme sa však zaujímali najmä o také algoritmy, ktoré sa dokážu vysporiadať s časovo závislými dátami, rôznymi typmi dopravných prostriedkov (módmi) a alternatívnymi cestami. Pri tvorení~~

~~požiadaviek na aplikáciu sme sa inšpirovali podobnými existujúcimi aplikáciami, ktoré okrem vyhľadávania spojov ponúkajú rôzne iné funkcionality. O tom sa zmieňujeme v kapitole Východiská.~~

~~Napokon sme sa rozhodli pre algoritmus, ktorý využíva fakt, že mestská doprava jazdí po vopred určených trasách. Týmto algoritmom je RAPTOR algoritmus, ktorého výhodou je, že je pomerne jednoducho prispôsobiteľný. Navrhli sme zmeny ako upraviť algoritmus, aby vedel pracovať s dátami poskytnutými Dopravným podnikom Bratislava. Ďalej sme navrhli dátovú štruktúru pre algoritmus, ktorá udržuje dáta statických cestovných poriadkov ako aj dáta o meškaní vozidiel. Bolo nutné navrhnúť ďalšie zmeny v algoritme, aby zohľadňoval pri vyhľadávaní zadané používateľské preferencie. Navrhli sme filtre, ktoré minimalizujú množinu všetkých ciest na množinu optimálnych ciest. Vo fáze implementácie sme natrafili na niekoľko problémov s dátami, keďže nevyhovovali~~

~~forme, ktoré RAPTOR algoritmus vyžadoval. Museli sme sa vysporiadať s veľkosťou dátovej štruktúry pre algoritmus, ako aj s neaktuálnymi dátami. Vytvorili sme simulátor času, keďže dáta sú z roku 2018 a my potrebujeme simulovať vyhľadá- vanie v aktuálny deň. Na klientskej strane vznikol problém so zobrazením zastávok na mape tak, aby zobrazenie bolo čo najviac používateľsky prívetivé.~~

Aplikáciu sme testovali v jednotlivých fázach vývoja. Algoritmus sme stále obohacovali

o nové vylepšenia. Začali sme vyhľadávaním nad statickými cestovnými poriadkami,

pokračovali sme prispôsobením algoritmu a dátovej štruktúry, aby prihliadal na

meškania spojov a poskytoval cesty zodpovedajúce aktuálnej dopravnej situácii. Ďalej

sme vytvorili a aplikovali filtre na cesty tak, že výsledkom sú vždy optimálne cesty.

V poslednom kroku sme prispôsobili algoritmus, aby pri vyhľadávaní vedel zohľadniť

používateľské kritériá. Pre potreby testovania, ale aj vylepšovania algoritmu sme si

vytvorili testovacie dáta. Správnosť vyhľadaných ciest tak mohla byť kontrolovaná.

(aplikácia bola najskôr otestovaná na zmenšenej sade dát, aby bolo možné manuálne porovnať správnosť algoritmu, potom otestovaná aj na dátach od DPB z roku 2018). Testovali sme v jednotlivých fázach vývoja.

Testovanie prebiehalo na reálnej sade dát od Dopravného podniku Bratislava. Dáta statických cestovných poriadkov, ako aj dáta o meškaní vozidiel sú z roku 2018. Keďže sme nevedeli vyhodnocovať správnosť vyhľadaných ciest, najskôr sme si vytvorili zmenšenú sadu dát. Správnosť vyhľadaných ciest tak mohla byť kontrolovaná manuálne už vo fáze vývoja.

Na reálnej sade dát sme testovali, ako sa množina vyhľadaných ciest mení pri zmene používateľských preferencií. Vyhľadané cesty sa menia aj podľa toho, či prihliadame na meškanie vozidiel. Cesty sú optimálne podľa takých pravidiel, ako sme určili pri návrhu aplikácie.

Aplikácia má niekoľko častí, ktoré by bolo možné v budúcnosti vylepšiť. Počas implementácie sme objavili dáta, ktoré sa nám nepodarilo zanalyzovať. Išlo o dáta, ktoré predstavujú časy odchodu zo zastávok po 23:59 hodine. Tieto dáta sme odignorovali a z toho dôvodu nemusí správne fungovať vyhľadávanie na prelome dní.

Ďalšou možnosťou vylepšenia by bolo zvážiť, v akej fáze vyhľadávania ciest je najvhodnejšie filtrovať vyhovujúce cesty na optimálne. V tomto riešení zapracovávame filtre na vyhľadané cesty po zbehnutí algoritmu. Stálo by za zváženie, či nie je efektívnejšie ich filtrovať už počas behu algoritmu.

(pre nás bolo výhodnejšie vidieť všetky cesty a potom vymýšlať a upravovať filtre. Teraz vak už vieme aké filtre potrebujeme a preto by sa dalo filtovať už počas vyhľadávania ciest. Pri porovnaní by sme zistili, či sa neodfiltovali tie cesty, ktoré mohli byť vyhodnotené ako optimálne + porovnanie času behu algoritmu)

Ďalšou možnosťou vylepšenia by bolo zvážiť, v akej fáze vyhľadávania ciest je najvhodnejšie filtrovať vyhovujúce cesty na optimálne. Aby sme používateľovi ponúkli alternatívne optimálne cesty algoritmus hľadá všetky cesty zo začiatočnej zastávky do konečnej zastávky. Následne na tieto cesty aplikujeme filtre a získavame množinu optimálnych ciest. Stálo by za zváženie a overenie, či nie je efektívnejšie ich filtrovať už počas behu algoritmu.

V neposlednom rade by mohlo byť vyhľadávanie paralelizované na viacerých CPU

jadrách. Linky v dátovej štruktúre sú zapísané nezávisle a pri vyhľadávaní môže každé

jadro spracovávať inú podmnožinu liniek zvlášť.

V kapitole Testovanie ukážeme schopnosť aplikácie vyhľadávať optimálne cesty s prihliadnutím

na meškanie. Správnosť vyhľadaných ciest sa dokazuje ťažko, keďže nemáme k dispozícii porovnateľné dáta. Existujúce aplikácie vyhľadávajú cesty bez prihliadnutia na meškanie. My máme k dispozícii dáta s platnosťou cestovných poriadkov v minulosti a spomínané aplikácie neumožňujú vyhľadávanie v ďalekej minulosti. Tým pádom nevieme porovnať ani vyhľadané cesty bez prihliadnutia na meškanie.

Vyhľadávací algoritmus sme testovali postupne. Najskôr algoritmus hľadal len jednu najkratšiu cestu nad statickými cestovnými poriadkami. Potom sme algoritmus vylepšili, aby prihliadal aj na meškanie vozidiel. Následne sme upravili vyhľadávanie, aby výsledná cesta nebola len jedna najkratšia, ale viacero optimálnych. Na záver sme upravili algoritmus, aby prihliadal aj na používateľské preferencie zadané pri vyhľadávaní.

Aplikácia používateľovi ponúka aj nasledujúce cesty tak, aby na stránke bolo aspoň 5 vyhľadaných ciest zoradených od času príchodu na začiatočnú zastávku vzostupne. Z toho dôvodu zbehne vyhľadávací algoritmus minimálne jedenkrát po jednom spustení hľadania ciest.

## Testovacie dáta

Dáta získané od Dopravného podniku Bratislava sú príliš rozsiahle a kontrola algoritmu človekom s týmito dátami bola nemožná. Vytvorili sme preto aj testovaciu sadu dát, nad ktorými sme spúšťali vyhľadávanie. Testovacia sada je tvorená zmenšeným počtom zastávok, liniek a jázd. Pre lepší prehľad v testovacích dátach sme nakreslili schému liniek a vygenerovali cestovný poriadok. Schému môžeme vidieť na obrázku. Algoritmus, ktorý vyhľadával nad testovacími dátami, kontrolovali traja cestujúci. Podľa ich názorov a skúseností sme vytvárali a upravovali filtre. Filtre slúžia na minimalizovanie množiny všetkých ciest, ktoré vedú zo skupiny začiatočných zastávok do skupiny konečných zastávok. Všetky vyfiltrované cesty však musia byť optimálne.

## Ukážky vyhľadávania ciest

Na záver sme testovali na plnej sade dát. V nasledujúcich ukážkach znázorníme, ako sa menila množina vyhľadaných ciest v závislosti od vývoja algoritmu.

### Meškanie

V tejto ukážke môžeme pozorovať, ako sa zmenila vyhľadaná cesta v závislosti od meškania. Na obrázku môžeme vidieť riadok z dát o meškaní vozidiel, ktorý poukazuje na to, že v deň 5.2.2018 v čase 9:59 nadobudla linka 5 na zastávke Karlova Ves meškanie 1 minútu. Vyhľadávanie sme spustili 5.2.2018 v čase 10:00. Deň 5.2.2018 je v konfigurácii nastavený ako aktuálny dátum.

Hľadáme cestu s vyhľadávacími parametrami na obrázku 4.1(a). Najskôr algoritmus hľadal len nad statickými cestovnými poriadkami. Našiel preto jednu najrýchlejšiu cestu (obrázok 4.1(b)). Po tom, ako sme v algoritme začali prihliadať na meškanie vozidiel, zmenil sa aj výsledok vyhľadávania (obrázok 4.1(c)).

Jazda linky 5 príde na zastávku Botanická záhrada s meškaním a z toho dôvodu sa mení aj prestupná zastávka medzi jazdami. Predvolený minimálny čas na prestup medzi linkami je 1 minúta. Keďže jazda linky 5 má meškanie, cestujúci by nemal žiadny čas na prestup na zastávke Vysoká, Tchibo Outlet. Prestupnou zastávkou sa preto stala zastávka Poštová - Martinus.

Všimnime si ešte označenie meškania. Meškajúca jazda má zafarbené časy červenou farbou. Po kliknutí na detail (obrázok 4.1(d)) vidíme aj hodnotu meškania jazdy. Časy z druhej jazdy sú čiernou farbou, pretože táto jazda ešte nevyrazila zo začiatočnej zastávky v čase vyhľadávania.

Obr. 4.1: Testovanie meškania

### Alternatívne cesty

Následne sme vylepšili vyhľadávanie tak, aby aplikácia cestujúcemu ponúkla vždy alternatívne

optimálne cesty. Prihliadame už aj na meškanie spojov. Spustili sme vyhľadávanie s parametrami na obrázku 4.2(a). Doteraz aplikácia vracala iba jednu najrýchlejšiu cestu a so zadanými parametrami by bola vyhľadaná cesta na obrázku 4.2(b).

Po vylepšení vyhľadávania nám aplikácia ponúkla množinu ciest, ktoré sú na obrázku 4.2(c). Vidíme prvú najrýchlejšiu cestu s 2 prestupmi, druhú pomalšiu s jedným prestupom a tretiu s najhorším časom príchodu do cieľa bez nutnosti prestupu.

Vo filtrovacom mechanizme ciest je podmienka, že najrýchlejšiu cestu zobrazujeme vždy. Cesty z neskoršími príchodmi do cieľa zobrazujeme len vtedy, ak majú menej prestupov. Avšak nemôžu prísť do cieľa s veľkým oneskorením po najrýchlejšej ceste. Rozdiel v časoch musí byť menší alebo rovný, ako rozdiel počtu prestupov vynásobený 𝑚 minútami. V projekte sme si určili 𝑚 = 5 minút. Najrýchlejšou cestou je prvá cesta na obrázku 4.2(c). Rozdiel v prestupoch prvej a druhej cesty je 1 a rozdiel v časoch príchodu do cieľa sú 3 minúty (3 minúty ≤ 1 × 5 minút). Rozdiel v prestupoch medzi

prvou a treťou cestou sú 2 prestupy a rozdiel v časoch príchodu do cieľa je 8 (8 minút ≤ 2 × 5 minút).

Na obrázkoch 4.2(b) a (c) si môžeme všimnúť zafarbené časy zelenou farbou. Čo znamená, že jazda linky už vyrazila z prvej zastávky a nemá žiadne meškanie. V detaile prvej cesty (obrázok 4.2(d)) môžeme vidieť označenie "včas" pri nemeškajúcej jazde.

Obr. 4.2: Testovanie alternatívnych ciest

### Meškanie a alternatívne cesty

Pomocou nasledujúcej ukážky znázorníme, aký vplyv má meškanie na zobrazenie optimálnych

ciest. Na obrázku 4.3(a) vidíme vyhľadávacie parametre. Najskôr sme definovali v systéme aktuálny deň 4.2.2018, aby získané meškania pri vyhľadávaní 5.2.2018 nemali vplyv na vyhľadávanie. Aplikácia ponúkla jednu vyhovujúci cestu a žiadne alternatívne optimálne cesty (obrázok 4.3 (b)).

Potom sme zmenili dátum v systéme na 5.2.2018 a čas 6:00. Teraz už algoritmus prihliada na meškania, ktoré vznikli do 6:00 hodiny. Jazda linky 5 má minútové meškanie (označené červenou farbou). Zaujímavé však je, že pribudla aj alternatívna cesta, ktorá nevyžaduje začiatočný peší presun. V predchádzajúcom hľadaní bola táto cesta odfiltrovaná. Potrebný peší presun na začiatku tiež penalizujeme ako prestup.

V prípade 4.3(b) bola priama cesta odfiltrovaná, pretože rozdiel v časoch príchodu do cieľa je väčší ako rozdiel v počte prestupov vynásobených 5 minútami (6 minút ≤ 1×5 minút). V prípade 4.3(c) je už rozdiel v časoch rovných 5 minút. Preto cesta s menším počtom prestupov nebola vyfiltrovaná.

Obr. 4.3: Testovanie meškania a alternatívnych ciest

### Používateľské preferencie

V aplikácii môže používateľ nastaviť okrem základných vyhľadávacích parametrov aj ďalšie parametre. Nachádzajú sa vo vysúvacej lište pod povinnými vyhľadávacími parametrami.

Ďalej sme upravili algoritmus, aby počas vyhľadávania ciest prihliadal na zadané prídavné parametre. Na obrázku 4.4(a) môžeme vidieť zadané parametre vyhľadávania. Všimnime si najmä, že maximálny počet prestupov je prednastavený na hodnotu 3 a začiatočný bod nie je zastávka, ale aktuálna poloha používateľa.

Vyhľadané cesty (obrázok 4.4(b)) sú dve. Jedna cesta obsahuje 1 prestup a druhá neobsahuje žiadny. Po zmene vyhľadávacích parametrov (obrázok 4.4(c)) je vyhľadaná cesta už len jedna – tá bez prestupu (obrázok 4.4(d)).

Obr. 4.4: Testovanie obmedzenia počtu prestupov

Na ďalšom obrázku ukážeme, ako sa algoritmus vysporiada s obmedzením dĺžky pešieho presunu. Po zadaní parametrov na obrázku 4.5(a) aplikácia ponúkla cesty na obrázku 4.5(b). Tu vidíme ponuku dvoch optimálnych ciest. Prvá cesta je najrýchlejšia, ale obsahuje dva pešie presuny. Druhá cesta obsahuje len jeden peší presun. Následne sme zmenili vo vyhľadávacích parametroch hodnotu maximálnej dĺžky pešieho presunu na 4 minúty (obrázok 4.5(c)). Na obrázku 4.5(d) môžeme vidieť vyhľadanú jednu cestu. Najrýchlejšia cesta bola vyfiltrovaná, keďže obsahovala 6 – minútový peší presun.

Obr. 4.5: Testovanie obmedzenia dĺžky peších presunov

V ďalšej ukážke znázorníme fungovanie nastavenia minimálneho času na prestup. Na obrázku 4.6(a) je hodnota minimálneho času na prestup nastavená na hodnotu 1. Vyhľadaná cesta (obrázok 4.6(b)) vyžaduje 1 prestup a čas na prestup medzi linkami je 1 minúta. Po tom, ako sme zmenili hodnotu minimálneho času na prestup na hodnotu 3 vo vyhľadávacích parametroch (obrázok 4.6(c)), dostávame inú vyhľadanú cestu s jedným prestupom (obrázok 4.6(d)). Na prestup medzi zastávkami má cestujúci 4 minúty.

Obr. 4.6: Testovanie minimálneho času na prestup

V poslednej ukážke dokážeme fungovanie vyhľadávania s prihliadnutím na obmedzenie len nízkopodlažných vozidiel. Na obrázku 4.7(a) vidíme zobrazené parametre vyhľadávania a na obrázku (b) vidíme vyhľadanú cestu. Cesta obsahuje 2 prestupy a je zložená z troch jázd. Len pre jednu z jázd tejto cesty platí, že má priradené nízkopodlažné vozidlo. Po zmene vyhľadávacích parametrov (obrázok 4.7(c)) je vyhľadaná iná cesta (obrázok 4.7(d)). Táto cesta obsahuje 2 jazdy, pričom obe jazdy majú priradené nízkopodlažné vozidlo.

Obr. 4.7: Testovanie hľadania len nízkopodlažných vozidiel

### Nasledujúce cesty

Používateľovi poskytujeme nielen tie optimálne cesty, ktoré začínajú najbližšie od zadaného času a dátumu, ale aj nasledujúce v chronologickom časovom slede. Na jednu stránku aplikácia zobrazí aspoň 5 ciest. Pre všetky tieto cesty platí, že sú optimálne. Vyhľadávací algoritmus sa pre jedno vyhľadanie spustí najmenej jeden krát. Na obrázku (a) môžeme vidieť vyhľadávacie parametre a na obrázku (b) a (c) je rozdelený zoznam vyhľadaných ciest. Vyhľadávací algoritmus bol v tomto prípade na pozadí spustený 3 krát. Po prvom zbehnutí algoritmu bola vyhľadaná 1. cesta, po druhom spustení boli vyhľadané ďalšie 3 optimálne cesty a na posledné spustenie bola vyhľadaná posledná cesta.

## Výpočtové časy

V nasledujúcej tabuľke ukážeme výpočtový čas jednotlivých vyhľadávaní. Testovanie prebiehalo na zariadení s procesorom Intel® Core ™ i7-7700 HQ CPU @ 2,80GHz.

|  |  |
| --- | --- |
|  | Výpočtový čas (ms) |
| Meškanie | 562 |
| Alternatívne cesty | 702 |
| Meškanie + alternatívne cesty | 691 |
| Maximálny počet prestupov | 625 |
| Maximálny čas na peší presun | 646 |
| Minimálny čas na prestup | 681 |
| Len názkopodlažné vozidlá | 721 |
| Nasledujúce cesty | 1871 |

## Zhrnutie

V ukážkach sme priblížili funkciu vyhľadávania ciest v mestskej hromadnej doprave Bratislava. Použili sme pri tom dáta statických cestovných poriadkov od Dopravného podniku Bratislava s platnosťou pre rok 2018. Využili sme aj dáta o meškaní vozidiel zo dňa 5.2.2018. Výsledné cesty, ktoré aplikácia ponúkla boli podľa očakávania s výnimkou vyhľadávania na prelome dní.

V ukážkach sme použili také príklady, pri ktorých môžeme jednoznačne vidieť zmenu množiny vyhľadaných ciest pri vývoji algoritmu, ako aj pri zmene používateľských preferencií. Používateľské preferencie sú po nainštalovaní aplikácie prednastavené na hodnoty zaznamenané v tabuľke:

|  |  |
| --- | --- |
| Názov preferencie | Preddefinovaná hodnota |
| Maximálny počet prestupov | 2 |
| Maximálny čas pešieho presunu | 8 min |
| Minimálny čas na prestup | 1 min |
| Len nízkopodlažné vozidlá | nie |

Po tom, ako používateľ zmení hodnotu niektorej z preferencií, hodnota zostane zapamätaná aj pre nasledujúce vyhľadávania.

Aplikácia má aj iné funkcionality, ktoré sme testovali z hľadiska používateľského rozhrania: vybranie zastávok z mapy, zobrazenie zoznamu vyhľadaných ciest a detailu cesty, ako aj zobrazenie liniek a ich trás.

Testujúcimi osobami boli traja cestujúci, ktorí pravidelne využívajú verejnú dopravu a  podobné aplikácie. Na základne ich návrhov sme pridali ešte históriu vyhľadávania, v ktorej sa udržuje 5 posledných záznamov začiatočnej a konečnej zastávky.