LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

TRANSPORTS

PROJEKTĒJUMA APRAKSTS

Autors:
Jēkabs Jaunarājs

Rīga 2016

Uzdevuma prasības

Kompānija FastService nolēma uzlabot pasūtījumu izpildes ātrumu un savu transporta līdzekļu izmantošanas efektivitāti. Tika nolemts maršrutu plānošanai nopietnāk izmantot datorus, kurus līdz šim viņi šādam uzdevumam nebija izmantojuši.
Tā kā pašā FastService nebija labu IT speciālistu pat lai precīzi noformulētu savas prasības, tad viņi palīdzību meklēja vienā no labākajām IT risinājumu projektēšanas kompānijām DSuP II.
FastService savas vēlmes noformulēja ar dažu principu palīdzību:
1. Pilsētai, kurā mēs strādājam, ir karte. Līdz ar to datoram vajadzētu to ņemt vērā, t.i. jābūt iespējai to ievadīt datorā;
2. Pilsētā ielas ir orientētas tikai Dienvidu-Ziemeļu un Rietumu-Austrumu virzienā;
3. Kartē ir atzīmētas klientu atrašanās vietas un FastService autoparka atrašanās vieta (uz ielas starp diviem krustojumiem);
4. Jābūt iespējai norādīt, ka dotajā brīdī kāds ceļa posms ir slēgts remonta dēļ vai atkal brīvs pēc remonta beigām;
5. Jābūt iespējai atzīmēt, ka iela ir vienvirziena vai divvirzienu (pēc noklusēšanas visas ielas ir divvirzienu);
6. Dispečers datorā ievada sarakstu ar apmeklējamiem klientiem un vēlas saņemt optimālo (īsāko) brauciena maršrutu (visi reisi sākas un beidzas autoparkā; šoferim maršruts pa ielām norādīts saprotamā un ērtā veidā).
DSuP II vadītājs apsolīja, ka viņa darbinieki veiks problēmas analīzi un izveidos projektējumu risinājumam. Tika garantēts, ka risinājums būs labs un to sapratīs jebkura puslīdz nopietna programmatūras izstrādes kompānija.
Lai iegūtu labāku rezultātu, DSuP II vadītājs lika katram kompānijas sistēmanalītiķim izveidot savu risinājuma versiju, lai vēlāk no tiem izveidotu gala variantu.

Ieejas fails

Programmas ieejas fails ir *.zip fails, kurš sevī satur vairākus *_street.json failus, katrs *_street.json fails sevī satur informāciju par vienu pilsētas ielu.

*_street.json faila shēma:

```
{
  "$schema": "http://json-schema.org/schema#",
  "title": "Street",
  "type": "object",
  "required": ["id", "name"],
  "properties":{
    "id": {
      "type": "number",
      "description": "Street identifier"
    },
  "name": {
      "type": "string",
      "description": "Name of the street"
    },
  "direction": {
      "type": "string",
      "description": "North-South or East-West or South-North or West-East"
  },
  "driveDirection": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "All allowed riving directions on street"
  },
  "intersections": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "All intersecting streets to current street in South-North or West-East directions"
  }
  }
}
```

*_street.json faila piemērs:

```
{
  "id": 31425,
  "name": "Gustava iela",
  "direction": "North-South",
  "driveDirection": [
    "North-South",
    "South-North"
  ],
  "intersections":[
    "Stuxnet iela",
    "Skynet iela",
    "Neo iela",
    "Stormtrooper iela"
  ]
}
```

Papildus, *.zip fails satur arī *_location.json failus, kuros ir atzīmētas klientu atrašanās vietas un FastService autoparka atrašanās vietas:

*_location.json faila shēma:	*_location.json faila piemērs:

<pre>{
  "$schema": "http://json-schema.org/schema#",
  "title": "Location",
  "type": "object",
  "required": ["id", "locationType", "name1", "name2", "name3"],
  "properties":{
  "id":{
  "type": "number",
  "description": "unique ID for location"
  },
  "locationType":{
  "type": "string",
  "description": "Client location or fastService location"
  },
  "name1": {
  "type": "string",
  "description": "Name of the street the location is on"
  },
  "name2": {
  "type": "string",
  "description": "Name of the intersecting street in South or West direction from location"
  },
  "name3": {
  "type": "string",
  "description": "Name of the intersecting street in North or East direction from location"
  }
  }
}</pre>
	<pre>{
	  "id": 2345,
	  "locationType": "Client location",
	  "name1": "Alfreda iela"
	  "name2": "Gunara iela",
	  "name3": "Henrija iela"
	}</pre>

Papildus, *.zip fails satur arī *_road_works.json failus, kuros ir atzīmētas ceļu remontu vietas.

*_road_works.json faila shēma:	*_road_works.json faila piemērs:

<pre>{
  "$schema": "http://json-schema.org/schema#",
  "title": "Location",
  "type": "object",
  "required": ["id", "name1", "name2", "name3"],
  "properties":{
   "id":{
    "type": "number",
    "description": "unique ID for location"
   },
   "name1": {
    "type": "string",
    "description": "Name of the street having roadworks"
   },
   "name2": {
    "type": "string",
    "description": "Name of the intersecting street in South or West direction from location"
   },
   "name3": {
    "type": "string",
    "description": "Name of the intersecting street in North or East direction from location"
   }
  }
}</pre>
	<pre>{
	   "id": 2345,
	   "name1": "Alfreda iela"
	   "name2": "Gunara iela",
	   "name3": "Henrija iela"
	}</pre>

Rezultātu fails

Rezultāti tiek apkopoti directions_<startlocation>_<endlocation>.txt failā un satur instrukcijas pagriezieniem x;y ielu krustojumos:

directions_<startlocation>_<endlocation>.txt faila shēma:
Directions to get to <start_location> to <end_location>: at <street A> and <street B> intersection, <turn (Right or Left)> or <drive straight> at <street C> and <street D> intersection, <turn (Right or Left)> or <drive straight> at <street E> and <street F> intersection, <turn (Right or Left)> or <drive straight> You have arrived to <endlocation> To return to <startlocation>, follow directions: at <street A> and <street B> intersection, <turn (Right or Left)> or <drive straight> at <street C> and <street D> intersection, <turn (Right or Left)> or <drive straight> at <street E> and <street F> intersection, <turn (Right or Left)> or <drive straight> You have returned to <startlocation>

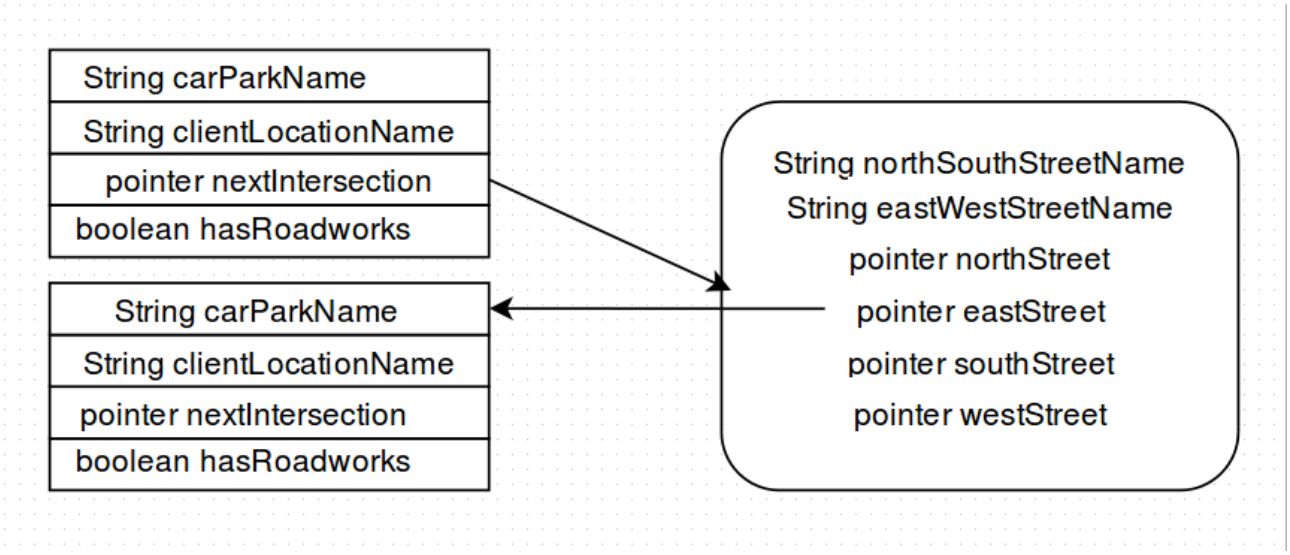
Iespējamie risinājumi un to pamatojums

Izvēlētā datu struktūra

Izvēlētā datu struktūra ir grafs, ar divu veidu mezgliem.

- Pirmais mezglu veids glabā informāciju par krustpjumu (ielu nosaukumi un norādes uz citiem mezgliem (skaits ne vairāk kā 4))
- Otrais mezglu veids glabā speciālu informāciju (ceļa darbi, klienta vai autoparka atrašanās vieta) par ceļa posmu vienā virzienā starp diviem krustojumiem

Iespējamās datu struktūras pamatfragments:



Starp katriem diviem krustojumiem ir viens (vienvirziena ielas) vai divas (divvirzienu ielas) objekts, kas satur informāciju par klientu lokācijām un autoparkiem, kā arī slēdzi, vai attiecīgajā posmā notiek ceļu darbi. Ja posmā nav autoparks vai klienta lokācija, mainīgajiem “carParkName” un “clientLocationName” tiek piešķirtas null vērtības.

Iespējamais algoritms

Iespējamais algoritms īsāko ceļu no sākuma posma starp krustojumiem uz mērķa posmu starp krustojumiem atrod sekojošā veidā:

1. Salīdzina starta adreses un finiša adreses ielu nosaukumus, ja tie ir vienādi un vēlamajā virzienā atļauts braukt un visos ceļa posmos starp adresēm nav remontdarbu, brauc pa esošo ielu taisni vēlamajā virzienā, līdz sasniedz finiša adresi.
2. Citā gadījumā pārskata katru starta adreses ielas krustojumu, lai pārliecinātos, vai starta adreses iela krusto finiša adreses ielu.
    - Ja nosacījums izpildās un ceļa posmos no starta adreses līdz ielu krustojumam un no krustojuma līdz mērķa adresei atļauts braukt vēlamajā virzienā un šajā virzienā nav ceļa darbu, braukt no starta adreses līdz krustojumam ar mēķa adreses ielu, veikt pagriezienu uz mērķa adreses ielu un braukt pa šo ielu līdz sasniegra mērķa adrese
3. Citā gadījumā braukt pa starta adreses ielu atļautajā virzienā Ziemeļu vai Austrumu virzienā,

pirmajā krustojumā nogriezties pa labi un atkārtot algoritmu no 1. soļa.

Lai izpildītu maršruta plānošanu, tiek veikta īsākā ceļa meklēšana no starta pozīcijas līdz finiša pozīcijai un pēcāk tiek veikta meklēšana no finiša pozīcijas atpakaļ uz starta pozīciju.

Realizācija un tās novērtējums

Citam programmētājam, atkarībā no pieredzes, šis risinājums var likties viegls līdz vidēji sarežģīts, līdz ar to realizācija varētu prasīt salīdzinoši neilgu laiku.

Programmas izpildes laiks atkarīgs pirmkārt no tai atvēlētajiem datora resursiem un otrkārt no tai padoto ieejas datu apjoma un treškārt no izpildes veida zemā līmenī (izpildes laiks virtuālajā mašīnā būs ilgāks kā tad, ja kods būs kompilēts mašīnkodā).

Programmas atmiņas patēriņš tieši atkarīgs no padoto ieejas datu apjoma.

Secinājumi

Izvēlētais risinājums ir atbilstošs uzdevuma nosacījumiem, bet nav izmaināms, ja mainās programmatūras prasības, piemēram kartē parādās ceļš, kurš neseko ziemeļu – dienvidu, rietumu-austrumu virzienam.

Risinājums neņem vērā atļauto braukšanas ātrumu dažādiem ceļiem, kas ir būtiski, jo īsākais ceļš kilometros ne obligāti nozīmē īsāko ceļa veikšanas laiku. Tāpat netiek ņemta vērā dažādu ceļu caurlaidība, jeb sastrēgumu biežums. Netiek ņemts vērā diennakts laiks, jo atkarībā no diennakts laika veidojās sastrēgumi un īsākais ceļš vairs nav ātrākais ceļš.