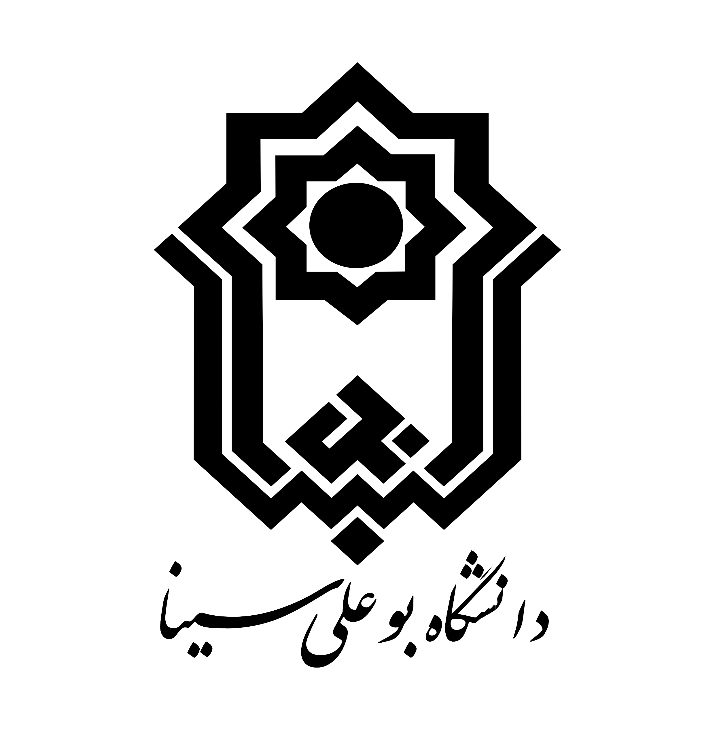
به نام یزدان



عنوان پروژه :

پروژه تخصصی کارشناسی واحد ساختمان داده

سامانه مدیریت حمل ونقل(سمنحشت)

نام دانشجو:

حسین فاضل 40112358028

مدیسا خبازی 40112358013

محمد امین شهابی40112507010

استاد مربوطه :

الهام افشار

بهمن ماه 1402

بسم الله الرحمن الرحیم

چکیده

پروژه ارائه داده شده درباره نقشه حمل و نقل شبیه ساز شده و یا ساده شده شهر تهران می‏باشد که با استفاده از سه وسیله نقلیه عمومی اتوبوس ، مترو و تاکسی و خطوط مشخص شده هر کدام از وسایل نقلیه باید سه خروجی مورد نظر که با دادن ورودی ها به عنوان مبدأ ، مقصد و ساعت شروع حرکت مشخص می‏شود ، نشان داده شود که هر کدام از این خروجی ها به ترتیب ، کمترین مسافت ، کمترین هزینه و نیز کمترین زمان می‏باشد.

همچنین در هر سه خروجی مورد نظر زمان پیمایش هر یک از خروجی ها به همراه مسیر طی شده آنها به نمایش گذاشته شده و محاسبه می‏شود.

برنامه نویسی شده است و نیز بخش(C++)این برنامه با زبان سی پلاس پلاس

طراحی شده است و همچنین در بستر گیت و یا به QT Widgetگرافیکی آن با

صورت حضوری درقالب کار تیمی به اتمام رسیده است.

فهرست مطالب

فصل اول: ساخت گراف و پیاده سازی الگوریتم

مقدمه.............................................................................................................................5

الگوریتم های مورد استفاده........................................................................................6

علت تشکیل کلاس و عملکرد آنها.............................................................................7

............................................................................. 21 Console نمایش خروجی در

) UI فصل دوم: نمایش گرافیکی (

مقدمه...............................................................................................................................22

نمایش خروجی...............................................................................................................23

منابع ........................................................................................................................................ 24

مقدمه :

در این پروژه وظیفه‏ی ما پیدا کردن بهترین مسیر بر حسب مسافت، زمان و هزینه، بین دو ایستگاه است.

برای حل کردن بخش های زمان، مسافت و هزینه، از الگوریتم دایجسترا برای پیدا کردن کمترین مقدار استفاده شده. البته در بخش های مختلف این الگوریتم دستخوش یکسری تغییرات جزئی شده است اما پایه تمام آن ها دایجسترا است.

برنامه‏ی ما به عنوان ورودی اول تعداد کاربرها سپس از هر کاربر ساعت شروع، مبدا و مقصد را می‏گیرد و نتیجه هر بخش(مسافت، هزینه و زمان) را به عنوان خروجی نشان می‏دهد.

الگوریتم های مورد استفاده

در این پروژه ما با استفاده از الگوریتم های مختلف به خصوص الگوریتم دایجسترا میتوانیم تمامی موارد مورد نیاز برای پیاده سازی خروجی خواسته شده را بدست بیاوریم که این امر با استفاده از

ساخت ماتریس مجاورت با محتوای فاصله مسیر ها میسر می‏شود.

خروجی اول که شامل بدست آوردن کوتاه ترین مسیر بین دو ایستگاه می‏باشد که با استفاده از الگوریتم دایجسترا بدست می‏اید و همچنین زمان مصرف شده در طی این مسیر نیز با استفاده از تابع calc\_time و نیز شی ساخته شده از کلاس تایم بدست آمده و به نمایش داده می‏شود.

همچنین برای بدست آوردن کمترین هزینه که با استفاده از ماتریسی که با قیمت همه ایستگاه ها پر میشود بدست امده و زمان صرف شده هم به نمایش گذاشته می‏شود و در نهایت برای بدست آوردن کمترین زمان که همان با استفاده از ماتریس مجاورت کوتاه ترین مسیر صورت می‏گیرد. زمان صرف شده هر مسیر را با الگوریتم دایجسترا بدست می‏آوریم.

علت تشکیل کلاس وعملکرد آن‏‏ها‏

در این پروژه تمامی تلاش خود را کرده‏ایم که از اضافه کردن فایل‌های اضافه

جلوگیری کنیم و در نهایت به 14 فایل که شامل کلاس ها و فایل های تکست است.(البته

فایل هایی همچون گزارش کار و ... هم بودند که ذکر نشدند.)

فایل‌ها به دو دستهhpp و cppو txt تقسیم می شوند که :

Hpp :

فایل‌هاییکه که این پیشوند را دارا هستند شامل تعریف کلیتوابع و کلاس ها هستند!

Cpp :

فایل هایی که این پیشوند را دارند شامل تعریف جز به جز توابع اند و همه عملیات در این بخش انجام می‌شود.

Txt :

این فایل‌ها برای جلوگیری از پیچیدگی و ناخوانی کد اضافه شدند که دارای محتوایی چون ایستگاه ها و... هستند که جلوتر به نحوه خواندن آنها به طورکامل اشاره شده است .

3i8:

3i8 به ترجمه ملالغطیه سمنحشت به عنوان اسکلت اصلی برنامه است که وظایف آن در

کلاس ها و توابع مختلف آن پخش شده است.

این فایل دارای دو بخش hpp و cpp است و همانطور که بالاتر اشاره شد هرکدام بار

خاصی را به دوش میکشند.

این فایل شامل یک کلاس به نام تهران است که محور اصلیه برنامه است و توابع زیادی را

در خود جای داده که در ادامه به آنها میپردازیم:

Struct Node\_p :

این struct یکی از مهم ترین و کاربردی ترین استراکت های استفاده شده در این پروژه

است که سه متغیر را در خود جای می‌دهد.

\*این تابع در زمان پر کردن ماتریس مجاورت گراف مد نظر(مالی) مورد استفاده قرار

گرفته است.

(همانطور که میدانید خط های ما به دو دسته تقسیم میشوند “L” ها که مختص به تاکسی و

مترو هستند و “b”ها که مختص اتوبوس هستند.)

Class Price :

\*تمام اعضای این کلاس عمومی هستند و این تابع هیچ متغیر خصوصی ندارد!

\*این کلاس در زمان پر کردن ماتریس مجاورت مدنظر(مالی) مورد استفاهد قرار گرفته

است!

\*این کلاس در تابع get\_min هم استفاده شده!

\*این کلاس در پیدا کردن بهترین مسیر از لحاظ زمانی نقش بسزایی را ایفا میکند!

Price\_edge :

این یک متغیر از نوع وکتور و یک شیع از استراکت Node\_p است و مقادیر تمامی یال

های متصل به یک ایستگاه را در خود داراست.

get\_min :

این تابع وظیفه پیدا کردن کمترین هزینه در بین یال های متصل به یک ایستگاه را داراست.

و به عنوان خروجی یک ایستگاه را برمی‌گرداند.

Class Node\_sp :

این یک کلاس است که هیچ عضو خصوصی ندارد.این کلاس در نظر گرفته شد تا بین

ایستگاه های مجاور هر ایستگاه هم هیزنه ترین وسیله و مسیر را پیدا کند.

\*کاربرد:پیدا کردن بهترین هزینه!

dist\_edge :

این تابع شامل تمام ایستگاه های مجاور ایستگاه فعلی است.

get\_min\_dist :

این تابع یک شی از ساختار( Struct) edge است که در میان تمامی ایستگاه های مجاور

ایستگاه فعلی کم هزینه ترین را بازمی‌گرداند.

get\_veicle(string name) :

این تابع هم مانند تابع قبلی یک شی از struct edge است چرا که احتیاج داریم تا داخل هر

عضو چند شاخصه قرار دهیم.

این تابع یک نام وسیله میگیرد تا اسم ایستگاهی که با این وسیله میرود را بازگرداند.

Class Tehran :

این کلاس مختص به شهر تهران است و گراف بر مبنای این شهر تشکیل شده!

این کلاس یکسری توابع بسیار مهم در خود جای داده تا ما بتوانیم با استفاده از آنها عملیات

مورد نظر خود را انجام دهیم. در اینجا میخواهیم به آنها بپردازیم:

گراف شهر را با استفاده از تابع readfile رسم می‏کنیم.تابع readfile با خواندن دو فایل

bus.txt &amp; line.txt اطلاعات را بدست می‌آورد و آنها را به طور مداوم در Tehran

می‌ریزد و به آنها مقدار می‌دهد.

تابع Mindistance :

این تابع وظیفه پیدا کردن کوچک ترین عضو را دارد.

این تابع با گرفتن یک شی از struct save\_directions و یک متغیر bool

sptSet(shortest path tree set),در یک حلقه شامل تمامی ایستگاه ها می‌چرخد و

چک میکند که آیا درخت بدست آمده درخت کمینه پوشا است یا نه واگر نبود و فاصله ای که گرفتهdist[] کوچیک‌تر از کوچیک‌ترین عضو قبلی(min\_index) بود عوض جدید رو به عنوان کوچک‌ترین عضو قرار میده و فاصله دریافتی رو با کم‌ترین فاصله جا به جا میکند. و به عنوان

خروجی کوچک ترین عضو(min\_index) را باز میگرداند.

stations :

این مپ یکی از اعاضای خصوصیه کلاس تهران میباشد و علت استفاده ازآن, پیمایش سریع

آن O(1)است.

هر عضو این مپ دارای دو بخش عددی و اسمی است که جایگاه عددیه ایستگاه و اسم

ایستگاه در آن‌ها مشخص شده است.

Linemap :

\*برای متوجه شدن بخش های جلوتر لازم است که این تابع و کاربرد آن هرچه سریع تر

شفاف شود!

این مپ یکی از اعاضای خصوصیه کلاس تهران میباشد و علت استفاده از آن,پیمایش سریع

آن O(1)است.

هر عضو این مپ دارای دو بخش اسمی و وکتور اسم هاست که بخش اول اسم لاین

مربوطه و بخش وکتور مربوط به ایسستگاه‌های داخل آن لاین می‌باشد.

\*کاربرد این مپ در پیدا کردن بهترین زمان است!

get\_value :

\*برای متوجه شدن بخش های جلوتر لازم است که این تابع و کاربرد آن هرچه سریع تر

شفاف شود!.

هیولای Find\_shortest\_path :

این تابع یکی از خروجی های مستقیم را با کمک تابع print\_shortest\_path به ما

می‏دهد که جلوتر با آن آشنا خواهیم شد.

بررسی محدوده ورودی:

در ابتدا، تابع محدوده ورودی را بررسی می‌کند تا اطمینان حاصل شود مبدا و مقصد در

بازه ایستگاه‌های معتبر (از 0 تا 59) قرار دارند.

ایجاد آرایه‌ی save\_directions:

یک آرایه از نوع save\_directions با نام dist[V] (که V برابر با 59 است) ایجاد

می‌شود. این آرایه برای ذخیره‌ی اطلاعات مسافت و مسیر استفاده می‌شود.

آماده‌سازی وضعیت درخت پوشای کمینه:

یک آرایه به نام sptSet با اندازه 59 ایجاد می‌شود و با مقدار پیش‌فرض false پر

می‌شود. این آرایه برای نشان دادن وضعیت درخت پوشای کمینه استفاده می‌شود.!

تنظیم اطلاعات ایستگاه مبدا:

وزن مسافت ایستگاه مبدا (src) به صفر تنظیم می‌شود و اسم ایستگاه مبدا به مسیر افزوده

می‌شود.

حلقه اصلی:

یک حلقه for از 0 تا V-1 (که در اینجا V برابر با 59 است) اجرا می‌شود. در هر

مرحله، تابع minDistance فراخوانی می‌شود تا کمترین ایستگاه جهت به‌روزرسانی

مسافت انتخاب شود. سپس، وضعیت این ایستگاه در درخت پوشای کمینه به True تغییر

می‌کند.

استفاده از sptSet:

مشاهده می‌شود که اگر راس v در مجموعه‌ی sptSet نباشد و مسافت فعلی تا v به مسافت

بی‌نهایت نزدیک نباشد و مسافت تا راس فعلی (u) به مسافت بی‌نهایت نزدیک نباشد و

مسافت از u به v بهتر از مسافت قبلی باشد، آنگاه مسافت و مسیر به‌روزرسانی می‌شود.

به‌روزرسانی مسافت و مسیر:

مسافت v به عنوان مجموع مسافت تا u و مسافت از u به v به‌روزرسانی می‌شود.

همچنین، مسیر جاری به v تا u کپی شده و مسیر جدید از u به v اضافه می‌شود.

Print\_shortest\_path :

چاپ مسیر:

این تابع به کاربر اطلاعات مربوط به مسافت کل مسیر و جزئیات مسیر (ایستگاه‌ها و

وسایل نقلیه مورد استفاده)ارایۀ می‌دهد.

حلقه بررسی ایستگاه‌ها :

یک حلقه for بر اساس ایستگاه‌های مسیر اجرا می‌شود تا اطلاعات هر بخش از مسیر

چاپ شود.

تشخیص نوع وسیله نقلیه:

بر اساس اطلاعات دریافتی از Line\_vehicle، نوع وسیله نقلیه (تاکسی، مترو یا اتوبوس)

تشخیص داده و چاپ می‌شود.

محاسبه زمان بین ایستگاه‌ها:

از تابع calc\_time برای محاسبه زمان بین ایستگاه‌ها استفاده می‌شود و زمان به صورت

تجمعی محاسبه شده و چاپ می‌شود.

چاپ زمان رسیدن:

در انتها، زمان رسیدن به مقصد نیز چاپ می‌شود.

Complete\_matrix\_p :

پرکردن ماتریس مجاورت گراف هزینه برعهده این تابع این تابع است و با چک کردن

بخش اسمی linemap اتوبوس یا لاین بودن آن را مشخص میکند.

check\_lineیکی از متغیرهای این تابع میباشد که به صورت پایه با مقدار صفر تایین شده

تا زمانی که در linemap لاین تشخیص داده شد برنامه بتواند تاکسی را از مترو جدا کند

و مقادیر را در price\_edge از کلاس price بریزد!

حال که گراف هزینه هم تشکیل شد زمان پیدا کردن بهترین مسیر از لحاظ مالی است.

Find\_best\_cost :

یک نکته بسیار مهم در رابطه با این تابع این است که این تابع برپایه الگوریتم

دایجسترا(dijksrta) نوشته شده است.

این تابع یکی از خروجی های مستقیم را با کمک تابع print\_best\_cost به ما میدهد که

جلوتر با آن آشنا خواهیم شد.

بررسی محدوده ورودی:

در ابتدا، تابع محدوده ورودی را بررسی می‌کند تا اطمینان حاصل شود مبدا و مقصد در

بازه ایستگاه‌های معتبر (از 0 تا 59) قرار دارند.

ایجاد آرایه‌ی save\_directions:

یک آرایه از نوع save\_directions با نام dist[V] (که V برابر با 59 است) ایجاد

می‌شود. این آرایه برای ذخیره‌ی اطلاعات مسافت و مسیر استفاده می‌شود.

آماده‌سازی وضعیت درخت پوشای کمینه:

یک آرایه به نام sptSet با اندازه 59 ایجاد می‌شود و با مقدار پیش‌فرض false پر

می‌شود. این آرایه برای نشان دادن وضعیت درخت پوشای کمینه استفاده می‌شود.!

تنظیم اطلاعات ایستگاه مبدا:

وزن مسافت ایستگاه مبدا (src) به صفر تنظیم می‌شود و اسم ایستگاه مبدا به مسیر افزوده

می‌شود.

حلقه اصلی:

یک حلقه for از 0 تا V-1 (که در اینجا V برابر با 59 است) اجرا می‌شود. در هر

مرحله، تابع minDistance فراخوانی می‌شود تا کمترین ایستگاه جهت به‌روزرسانی

مسافت انتخاب شود. سپس، وضعیت این ایستگاه در درخت پوشای کمینه به True تغییر

می‌کند.

استفاده از sptSet:

مشاهده می‌شود که اگر راس v در مجموعه‌ی sptSet نباشد و هزینه فعلی تا v به هزینه

بی‌نهایت نزدیک نباشد و هزینه تا راس فعلی (u) به هزینه بی‌نهایت نزدیک نباشد و هزینه

از u به v بهتر از هزینه قبلی باشد، آنگاه هزینه و مسیر به‌روزرسانی می‌شود.

به‌روزرسانی هزینه و مسیر:

هزینه v به عنوان مجموع هزینه تا u و هزینه از u به v به‌روزرسانی می‌شود. همچنین،

مسیر جاری به v تا u کپی شده و مسیر جدید از u به v اضافه می‌شود.

و در نهایت اطلاعاتی که ذخیره شده از ایستگاه مبدا و مقصد را برمیگرداند.

Print\_best\_price :

چاپ مسیر:

این تابع به کاربر اطلاعات مربوط به مسافت کل مسیر و جزئیات مسیر (ایستگاه‌ها و

وسایل نقلیه مورد استفاده)ارایۀ می‌دهد.

حلقه بررسی ایستگاه‌ها :

یک حلقه for بر اساس ایستگاه‌های مسیر اجرا می‌شود تا اطلاعات هر بخش از مسیر

چاپ شود.

تشخیص نوع وسیله نقلیه:

بر اساس اطلاعات دریافتی از Line\_vehicle، نوع وسیله نقلیه (تاکسی، مترو یا اتوبوس)

تشخیص داده و چاپ می‌شود.

محاسبه زمان بین ایستگاه‌ها:

از تابع calc\_time برای محاسبه زمان بین ایستگاه‌ها استفاده می‌شود و زمان به صورت

تجمعی محاسبه شده و چاپ می‌شود.

چاپ زمان رسیدن:

در انتها، زمان رسیدن به مقصد نیز چاپ می‌شود.

Calc\_time :

این تابع با گرفتن مبدا,مقصد,لاین قبلی,وسیله در حال حاضر و یک شی از کلاس Time

سعی دارد تا در هر مرحله زمان را محاسبه و برگرداند. به نحوی که برای ساعات شلوغی

و عوض کردن وسیله و ... شرایط خود را اعمال میکند , تشخیص میدهد و در متغیر

speed میریزد.

Semanhasht :

این فایل ,فایل مدیریت برنامه است و از تمام توابع و کلاس‌ها خروجی هارا میگیرد و به

طور منظم آنها را به عنوان خروجی نشان می‌دهد.(این فایل مدیریت استان مربوطه و

درخواست هارا برعهده دارد.)

پینوشت : با توجه به استفاده از رابط گرافیکی این فایل تا حدی غیرقابل استفاده می شود.

Time :

این قسمت از کد یک کلاس به نام به نام Time است که وظیفه آن مدیریت زمان است.

این کد شامل توابع مختلفی است که برای کنترل صحت ساعت, دقیقه و نیمه‌ی روز

(AM,PM) است که در صورت هرگونه مشکل در ورودی های ساعت خروجیه خاص

invalid argument را برنامه باز میگرداند و کاربر را از مشکل با خبر میکند.

این کلاس شامل توابع print,oprator ها هم هست که وظیفه oprator ها وابسته به

چیزی است که به کلاس اضافه میکنند. مثلا oprator + وظیفه جمع زدن دقیقه هارا

برعهده داراست و oprator &gt;&gt; است که با استفاده از این اوپریتور ما مقادیری را

درساعات ترافیک و ... میریزیم.

Vehicle :

تابع machine یک ورودی ازنوع استرینک میگیرد که این ورودی همان وسیله نقلیه است

و با توجه به آن اسم در فایل ها به دنبال اسم فایلی مشابه با ورودی همراه با پسوند “v”

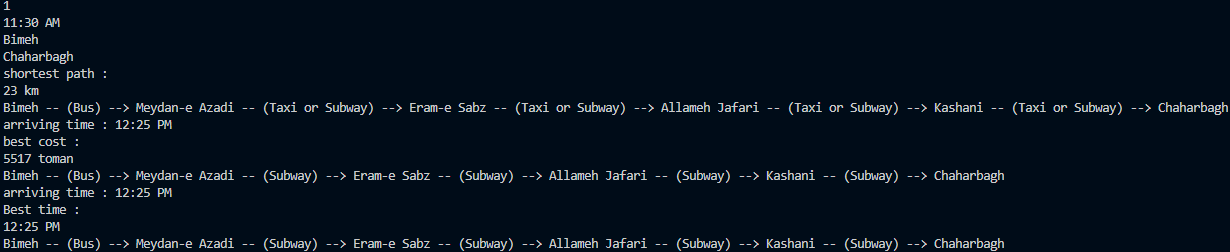
می‌گردد و زمانی که آن را پیدا کرد به ترتیب خاصی که در فایل ها ذکر شده به آن وسیله

نقلیه مقدار میدهد.

دو تابع دیگر در این کلاس نیز وظیفه چک کردن ساعت ترافیک برای حرکت یا عوض

کردن لاین ها را دارا هستند.

نمایش خروجی در کنسول:



مقدمه :

ما با استفاده از Qwidget شکل و نقشه گرافیکی خود را انجام داده ایم که در صفحه نخست صفحه نمایش طراحی شده توسط اعضای تیم به نمایش گذاشته می‏شود که با استفاده از دکلمه start به صفحه اصلی نقشه سمنحشت اشاره می‏شود.

بر روی هر نود از نقشه push bottun قرار داده شده است که با کلیک بر روی هر یک از آن‏ها به قسمت تابع مربوطه آن ارجا می‏کند و تنظیمات آن فراخوانی می‏شود.

و همچنین در Qbox پایین صفحه میتوان زمان و نود مبدا و مقصد خود را جهت اطمینان چک کنیم و بعد از انتخاب یکی از دکمه های کوتاه ترین مسافت کمترین هزینه و کمترین زمان مسیر مربوطه با رنگ سبز ویا قرمز نشان داده می‏شود و هزینه ویا مسافت آن به نمایش داده می‏شود.

نمایش خروجی گرافیکی



A diagram of a network

Description automatically generated

منابع

* لینک های استفاده شده :

[www.geeksforgeeks.org](http://www.geeksforgeeks.org)

[www.stackoverflow.com](http://www.stackoverflow.com)

* لینک گیت هاب :

[https://github.com/Hossein-Fazel/Semanhasht](https://github.com/Hossein-Fazel/Semanhasht.git)