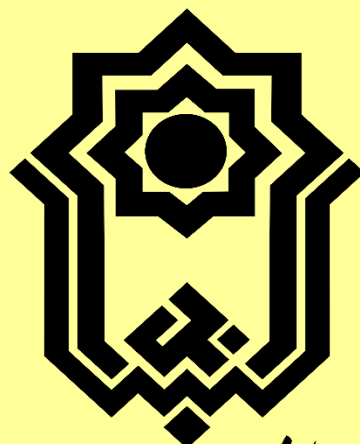


به نام یزدان



دانشگاه بوعلی سینا

عنوان پروژه :

پروژه تخصصی کارشناسی واحد ساختمان داده

سامانه مدیریت حمل و نقل (سمنحشت)

نام دانشجو:

حسین فاضل 40112358028

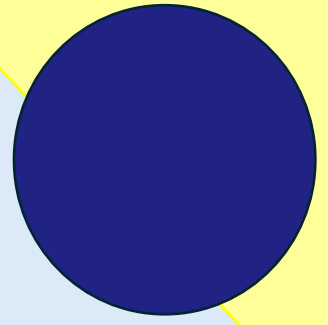
مدیسا خبازی 40112358013

محمد امین شهابی 40112507010

استاد مربوطه :

الهام افشار

بهمن ماه 1402



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



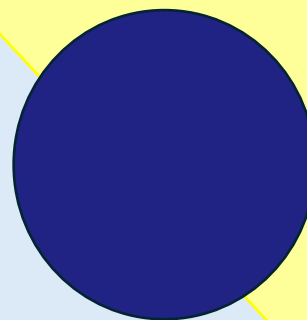
چکیده

پروژه ارائه داده شده درباره نقشه حمل و نقل شبیه ساز شده و یا ساده شده شهر تهران می باشد 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 می شود ، نشان داده شود که هر کدام از این خروجی ها به ترتیب ، کمترین مسافت ، کمترین هزینه و نیز کمترین زمان می باشد.

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 می شود.

این برنامه با زبان سی پلاس پلاس (C++) برنامه نویسی شده است و نیز بخش گرافیکی آن با QT Widget طراحی شده است و همچنین در بستر گیت و یا به صورت حضوری در قالب کار تیمی به اتمام رسیده است.

فهرست مطالب



فصل اول: ساخت گراف و پیاده سازی الگوریتم

5.....	مقدمه
6.....	الگوریتم های مورد استفاده
7.....	علت تشکیل کلاس و عملکرد آنها
21	نمایش خروجی در Console

فصل دوم: نمایش گرافیکی (UI)

22.....	مقدمه
23.....	نمایش خروجی
24	منابع



مقدمه :

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

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

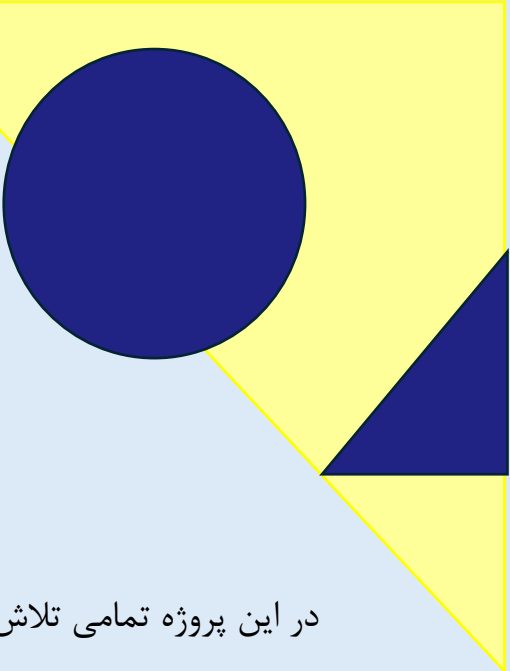


الگوریتم های مورد استفاده

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 می شود.

خروجی اول که شامل بدست آوردن کوتاه ترین مسیر بین دو ایستگاه می باشد که با استفاده از الگوریتم دایجسترا بدست می آید 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 می شود.

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



علت تشکیل کلاس و عملکرد 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 کلاس‌ها و فایل‌های تکست است. (البته فایل‌هایی همچون گزارش کار و ... هم بودند که ذکر نشدند).
فایل‌ها به دو دسته `hpp` و `cpp` و `txt` تقسیم می‌شوند که :

: Hpp

فایل‌هایی که این پیشوند را دارا هستند شامل تعریف کلیت‌ابع و کلاس‌ها هستند!

: Cpp

فایل‌هایی 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 ناخوانایی کد اضافه شدند که دارای محتوایی چون ایستگاه‌ها و... هستند که جلوتر به نحوه خواندن آنها به طور کامل 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 اصلی 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 می دهد.

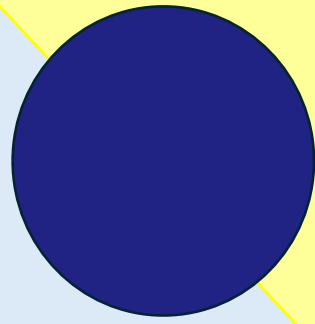
*این تابع در زمان پر کردن ماتریس مجاورت گراف مد نظر(مالی) مورد استفاده قرار

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 استفاده قرار گرفته است!

*این کلاس در تابع `get_min` هم استفاده شده!

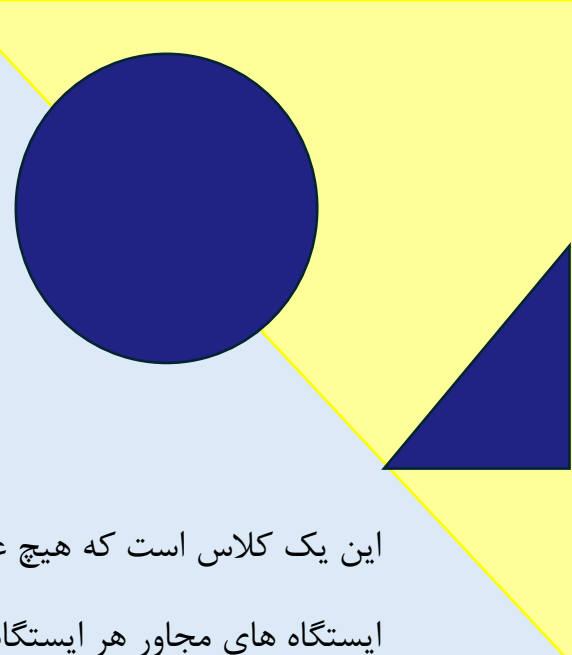
*این کلاس در پیدا کردن بهترین مسیر از لحاظ زمانی نقش بسزایی را ایفا میکند!

: 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 برمی گرداند.



: 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 هزینه 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 edge (Struct) است که در میان تمامی ایستگاه های مجاور ایستگاه فعلی کم هزینه ترین را باز می گرداند.

: 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 می رود را باز گرداند.



: 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` , `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 و اگر نبود و فاصله ای که گرفته `dist[]` کوچک تر از کوچک ترین عضو قبلی (`min_index`) بود عوض جدید رو به عنوان کوچک ترین عضو قرار میدهد و فاصله دریافتی رو با کم ترین فاصله جا به جا میکند. و به عنوان خروجی کوچک ترین عضو (`min_index`) را باز میگرداند.



: stations

این مپ یکی از اعضای خصوصیۀ کلاس تهران میباشد و علت استفاده از آن، پیمایش سریع آن O(1) است.

هر عضو این مپ دارای دو بخش عددی و اسمی است که جایگاه عددیۀ ایستگاه و اسم ایستگاه در آن‌ها مشخص شده است.

: Linemap

*برای متوجه شدن بخش‌های جلوتر لازم است که این تابع و کاربرد آن هرچه سریع‌تر شفاف شود!

این مپ یکی از اعضای خصوصیۀ 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 ایستگاه‌های داخل آن لاین می‌باشد.

*کاربرد این مپ در پیدا کردن بهترین زمان است!

: get_value

*برای متوجه شدن بخش‌های جلوتر لازم است که این تابع و کاربرد آن هرچه سریع‌تر شفاف شود!



هیولای Find_shortest_path :

این تابع یکی از خروجی های مستقیم را با کمک تابع `print_shortest_path` به ما می دهد که جلوتر با آن آشنا خواهیم شد.

بررسی محدوده ورودی:

در ابتدا، تابع محدوده ورودی را بررسی می کند تا اطمینان حاصل شود مبدا و مقصد در بازه ایستگاه های معتبر (از 0 تا 59) قرار دارند.

ایجاد آرایه ی `save_directions`:

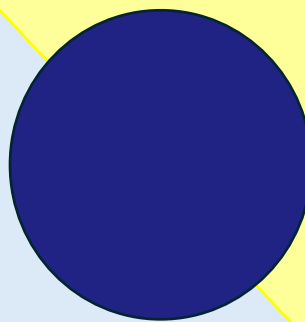
یک آرایه از نوع `save_directions` با نام `dist[V]` (که `V` برابر با 59 است) ایجاد می شود. این آرایه برای ذخیره ی اطلاعات مسافت و مسیر استفاده می شود.

آماده سازی وضعیت درخت پوشای کمینه:

یک آرایه به نام `sptSet` با اندازه 59 ایجاد می شود و با مقدار پیش فرض `false` پر می شود. این آرایه برای نشان دادن وضعیت درخت پوشای کمینه استفاده می شود!

تنظیم اطلاعات ایستگاه مبدا:

وزن مسافت ایستگاه مبدا (`src`) به صفر تنظیم می شود و اسم ایستگاه مبدا به مسیر افزوده می شود.



حلقه اصلی:

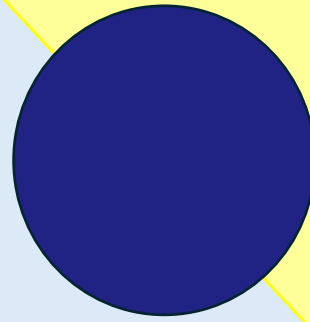
یک حلقه for از 0 تا V-1 (که در اینجا V برابر با 59 است) اجرا می‌شود. در هر مرحله، تابع minDistance فراخوانی می‌شود تا کمترین ایستگاه جهت به‌روزرسانی مسافت انتخاب شود. سپس، وضعیت این ایستگاه در درخت پوشای کمینه به True تغییر می‌کند.

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) ارائه 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:

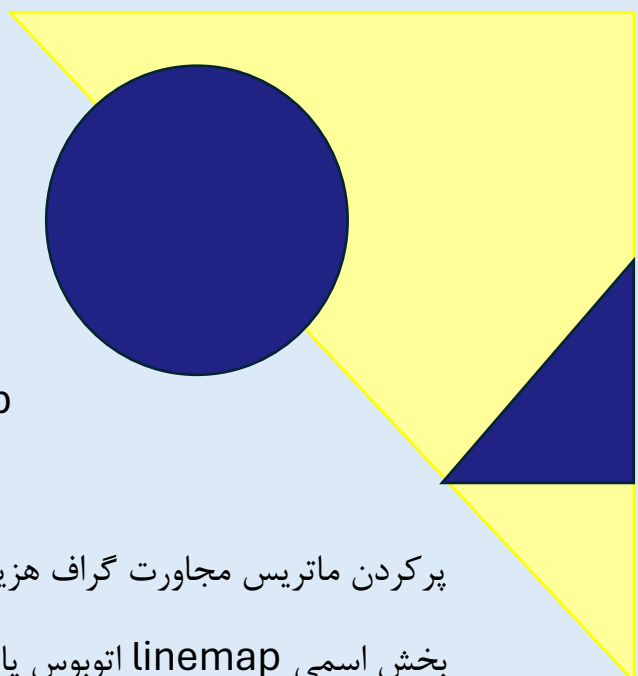
بر اساس اطلاعات دریافتی از **Line_vehicle**، نوع وسیله نقلیه (تاکسی، مترو یا اتوبوس) تشخیص داده و چاپ می‌شود.

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_line` یکی 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.

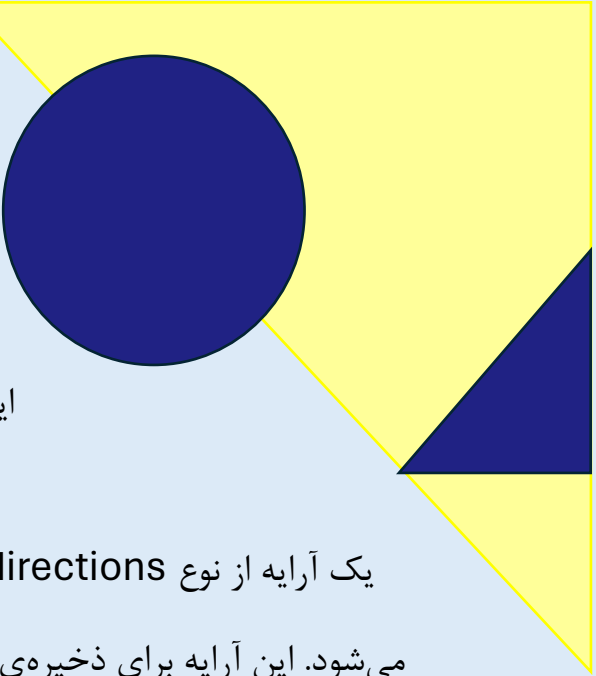
: Find_best_cost

یک نکته بسیار مهم در رابطه با این تابع این است که این تابع برپایه الگوریتم دایجسترا (`dijkstra`) 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 می کند تا اطمینان حاصل شود مبدا و مقصد در بازه ایستگاه های 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.

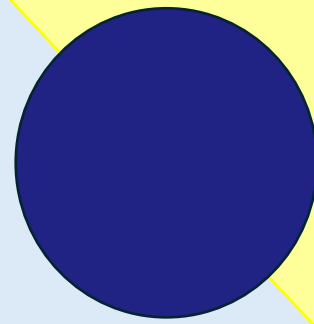
: Print_best_price

چاپ مسیر:

این تابع به کاربر اطلاعات مربوط به مسافت کل مسیر و جزئیات مسیر (ایستگاه‌ها و وسایل نقلیه مورد استفاده) ارائه 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 می شود.

محاسبه زمان بین ایستگاه‌ها:

از تابع `calc_time` برای محاسبه زمان بین ایستگاه‌ها استفاده می شود و زمان به صورت

تجمعی محاسبه شده و چاپ می شود.

چاپ زمان رسیدن:

در انتها، زمان رسیدن به مقصد نیز چاپ می شود.

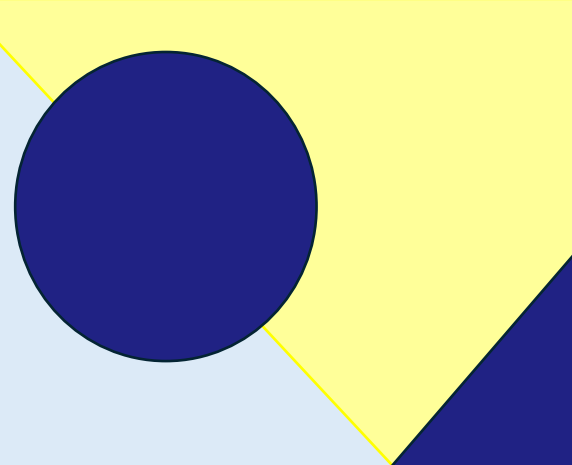
: `Calc_time`

این تابع با گرفتن مبدأ، 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 خروجی‌ها را می‌گیرد و به طور منظم آنها را به عنوان خروجی نشان می‌دهد. (این فایل مدیریت استان مربوطه و درخواست‌ها را برعهده دارد).

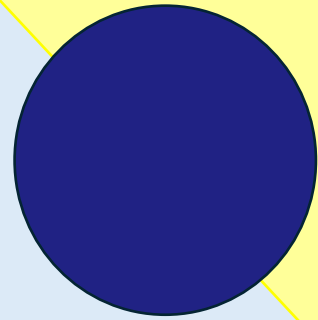
پینوشت : با توجه به استفاده از رابط گرافیکی این فایل تا حدی غیرقابل استفاده می‌شود.

: Time

این قسمت از کد یک کلاس 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 ورودی‌های 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 `operator` ها وابسته به چیزی است که به کلاس اضافه میکنند. مثلاً `operator +` وظیفه جمع زدن دقیقه‌ها را برعهده داراست و `>, >=` است که با استفاده از این اوپراتور ما مقادیری را در ساعات ترافیک و ... میریزیم.



: Vehicle

تابع machine یک ورودی از نوع استرینگ 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” می گردد 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:

```
1
11:30 AM
Bimeh
Chaharbagh
shortest path :
23 km
Bimeh -- (Bus) --> Meydan-e Azadi -- (Taxi or Subway) --> Eram-e Sabz -- (Taxi or Subway) --> Allameh Jafari -- (Taxi or Subway) --> Kashani -- (Taxi or Subway) --> Chaharbagh
arriving time : 12:25 PM
best cost :
5517 toman
Bimeh -- (Bus) --> Meydan-e Azadi -- (Subway) --> Eram-e Sabz -- (Subway) --> Allameh Jafari -- (Subway) --> Kashani -- (Subway) --> Chaharbagh
arriving time : 12:25 PM
Best time :
12:25 PM
Bimeh -- (Bus) --> Meydan-e Azadi -- (Subway) --> Eram-e Sabz -- (Subway) --> Allameh Jafari -- (Subway) --> Kashani -- (Subway) --> Chaharbagh
```

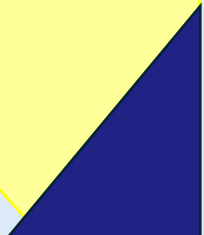


مقدمه :

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 می شود که با استفاده از دکمه start به صفحه اصلی نقشه سمنحشت اشاره می شود.

بر روی هر نود از نقشه push button قرار داده شده است که با کلیک بر روی هر یک از آنها به قسمت تابع مربوطه آن ارجا می کند و تنظیمات آن فراخوانی می شود.

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 می شود و هزینه ویا مسافت آن به نمایش داده می شود.





منابع

• لینک های استفاده شده :

www.geeksforgeeks.org
www.stackoverflow.com

• لینک گیت هاب :

<https://github.com/Hossein-Fazel/Semanhasht>