



ساختمان‌های داده  
سامانه مدیریت حمل و نقل شهروندی تهران

ریحانه السادات خسروی  
فاطمه دماوندی  
زینب اسلامی شکیب

استاد مهندس الهام افشار :

پاییز 1402

## فهرست مطالب

|    |   |
|----|---|
| 3  | مقدمه.....                              |
| 5  | ساختار پروژه.....                       |
| 7  | نحوه ی دقیق پاسخگویی به هر درخواست..... |
| 7  | کوتاهترین مسافت.....                    |
| 7  | کوتاهترین مسافت.....                    |
| 7  | هزینه.....                              |
| 9  | پایاده سازی.....                        |
| 11 | لینک گیت‌هاب.....                       |
| 11 | لینک‌های کمکی.....                      |

## مقدمه

### ساختمان های داده

درس ساختمان داده (Data Structures) یکی از مهم‌ترین و پایه‌ای‌ترین دروس مهندسی نرم‌افزار است که در فهم و تحلیل مسائل مختلف و طراحی الگوریتم‌های بهینه اهمیت زیادی دارد. در این درس، مفاهیم و اصول مربوط به ساختمان‌های داده و الگوریتم‌ها بررسی می‌شوند.

در این درس، دانشجویان با مباحثی همچون لیست‌ها، صف‌ها، درخت‌ها، گراف‌ها و هشتتیل‌ها آشنا می‌شوند. آن‌ها یاد می‌گیرند که چگونه اطلاعات را در یک ساختمان منطقی و بهینه ذخیره و مدیریت کنند و همچنین چگونه با استفاده از الگوریتم‌های مناسب این داده‌ها را پردازش کنند.

علاوه بر این، درس ساختمان داده به دانشجویان کمک می‌کند تا مفاهیم اصلی همچون جستجو، مرتب‌سازی، ادغام و جداسازی داده‌ها را بیاموزند و اهمیت انتخاب و استفاده از ساختمان داده مناسب برای هر مسئله را درک کنند.

در گزارش پروژه درس ساختمان داده، می‌توانید به توضیح مفاهیم اصلی مطرح شده در درس، مثال‌های عملی و کاربردی از این ساختمان‌ها در حل مسائل، بررسی الگوریتم‌های مرتبط و نحوه پیاده‌سازی آن‌ها در زبان برنامه‌نویسی مورد استفاده، و ارزیابی کارایی و کاربردی بودن این ساختمان‌ها در پروژه‌های واقعی پرداخته و نتایج و یافته‌های خود را به دقت گزارش کنید.

### الگوریتم دایجسترا

الگوریتمی است که برای پیدا کردن کوتاه‌ترین مسیر (Dijkstra's Algorithm) الگوریتم دایجسترا این گراف، ممکن است نشان‌گر شبکه جاده‌ها یا موارد دیگری بین دو گره در گراف به کار می‌رود. مطرح الگوریتم دایجسترا در سال ۱۹۵۶، توسط دانشمند کامپیوتری با نام ادسخر ویبه دیکسترا باشد. الگوریتم دایجسترا دارای انواع گوناگونی است و سه سال بعد، منتشر شد.

الگوریتم اصلی، کوتاه‌ترین مسیر بین دو گره را پیدا می‌کند؛ اما نوع متداول‌تر این الگوریتم، یک گره یکتا را به عنوان گره مبدا (آغازین) در نظر می‌گیرد و کوتاه‌ترین مسیر از مبدا به دیگر گره‌ها در گراف را با ساختن درخت کوتاه‌ترین مسیر پیدا می‌کند.

برای یک گره مبدا داده شده، الگوریتم، کوتاه‌ترین مسیر بین آن گره و دیگر گره‌ها را پیدا می‌کند. همچنین، الگوریتم دایجسترا برای پیدا کردن کوتاه‌ترین مسیر از یک گره یکتا به گره مقصد یکتای دیگری به کار می‌رود؛ برای انجام این کار، الگوریتم هنگامی که کوتاه‌ترین مسیر از مبدا به مقصد را پیدا کند، متوقف می‌شود.

### علت استفاده از الگوریتم استفاده شده در این پروژه

در این پروژه هدف پیدا کردن بهترین مسیر بین مبدا و مقصد در شهر تهران از طریق مترو، تاکسی، اتوبوس یا بی‌آرتی است.

این پروژه با استفاده از الگوریتم دیجسترا کوتاهترین مسیر بین مبدا و مقصد موردنظر را با در نظر گرفتن معیارهای زمان، مسافت و هزینه محاسبه می‌کند و نمایش می‌دهد.

## ساختمان‌های داده‌ی استفاده شده در این پروژه

در الگوریتم دیجسترا که برای پیدا کردن کوتاهترین مسیر در گرافها استفاده می‌شود، از مفاهیم و ساختمان‌های داده مختلفی استفاده می‌شود:

### 1. صف اولویت‌دار (Priority Queue):

صف اولویت‌دار یک ساختمان داده است که این امکان را فراهم می‌کند که اجزای آن براساس اولویت‌هایشان مرتب شوند، به طوری که همیشه عنصر با بالاترین اولویت (کوچکترین یا بزرگترین مقدار) را در بالای صف داشته باشیم. در الگوریتم دیجسترا، از صف اولویت‌دار برای نگهداری و مدیریت نودهایی که باید بررسی شوند و همچنین به ترتیب اهمیتشان مدیریت می‌شوند، استفاده می‌شود.

### 2. مپ (Map):

مپ یک ساختمان داده است که از جفت‌های مقادیر "کلید-مقدار" برای نگهداری داده‌ها استفاده می‌کند. در الگوریتم دیجسترا، از مپ برای نگهداری اطلاعات لازم در مورد هر نود و همچنین مسافت کمینه تا آن نود استفاده می‌شود.

### 3. آرایه‌ی پویا (Dynamic Array):

آرایه‌ی پویا یک ساختمان داده است که به صورت پویا تغییر اندازه می‌دهد و این امکان را فراهم می‌کند که اطلاعات را به صورت مکرر و سریع موجودیت‌های آن اضافه و حذف کنیم. در الگوریتم دیجسترا، از آرایه‌ی پویا برای نگهداری مسافت‌های کمینه تا هر نود استفاده می‌شود و بر اساس نتایج به‌روز شده تثبیت می‌شود.

### 4. لیست (برای لیست مجاورت):

لیست، یک ساختمان داده خطی است که اجازه می‌دهد چندین مقدار (آیتم) را در یک نوع داده ذخیره کنیم. در الگوریتم دیجسترا، از لیست مجاورت برای نگهداری همسایه‌های هر نود و وزن‌های مرتبط با آن‌ها استفاده می‌شود.

### 5. گراف:

گراف یک ساختمان داده است که از رئوس (نودها) و یال‌ها (یال‌ها) بین آن‌ها تشکیل شده است و روابط و تعاملات بین اجزای گراف را نشان می‌دهد. الگوریتم دیجسترا بر روی گراف‌ها اعمال می‌شود تا کوتاهترین مسیر بین دو رأس مشخص شده را پیدا کند.

## ساختار پروژه

در این پروژه از الگوریتم دایجسترا استفاده کردیم و برای درخواست‌های مختلف گراف ایجاد شده است. برای ذخیره هر گراف از یک فایل تکست استفاده می‌کنیم.

برای هر گراف یک فایل تکست ایجاد شده است که یال‌ها در این فایل ذخیره می‌شود و هنگامی که اطلاعات را از روی فایل می‌خوانیم تعدادی نود ایجاد می‌کنیم که شامل اسم مبدا هستند که به یک مقصد مرتبط می‌شود. هر نود شامل اطلاعاتی است که به چه مبدهایی متصل است.

الگوریتم دایجسترا برای کوتاهترین مسافت بین گره‌ها در گراف‌ها به کار می‌رود. در این الگوریتم، تنها یال‌هایی با کمترین وزن انتخاب می‌شوند تا مسیر کوتاهترین بین گره‌های مورد نظر محاسبه شود. علاوه بر این، از مپ برای نگهداری اطلاعات لازم در مورد هر گره و همچنین برای ذخیره‌سازی مسافت کمینه تا هر گره استفاده می‌شود.

برای محاسبه هزینه و هزینه‌های مرتبط با مسافت‌های طی شده، از یک گراف جدید استفاده می‌شود. در این گراف، هزینه‌های مربوط به استفاده از وسایل نقلیه مختلف مانند اتوبوس، مترو و تاکسی در نظر گرفته می‌شود. برنامه‌ی پیاده‌سازی شده می‌تواند نوع حمل‌ونقل را تشخیص داده و هزینه‌ها مرتبط با آن‌ها را محاسبه کند. این امکان در اعتبارگیری و تصمیم‌گیری در مورد مسیریابی و تعیین بهترین مسیر بسیار موثر است.

برای اجرای بهینه‌تر الگوریتم، از یال‌های جدید با وزن‌های متفاوت برای گراف استفاده می‌شود. این تغییرات در وزن‌ها به عنوان پارامترهایی مانند مسافت، سرعت وسیله نقلیه و هزینه‌های مرتبط در نظر گرفته می‌شود. از این روش برای افزایش دقت و کارایی در تعیین بهترین مسیر و مدت زمان لازم برای مسافت‌های مختلف بهره می‌بریم.

در نهایت، با ایجاد یال‌های جدید و مدیریت مناسب آن‌ها، الگوریتم دایجسترا قادر خواهد بود تا مسیریابی و محاسبه کوتاهترین مسیر با در نظر گرفتن هزینه‌ها و زمان صرف شده را به صورت دقیق و کارآمد انجام دهد.

\*\*\*توضیحات:

- به جهت این که برنامه ی تیم ما بر این بود که گرافیک داشته باشیم، همه ی الگوریتم ها طوری نوشته شده بودن که خروجی شون فعلا جداگانه داخل ترمینال نشون داده بشه و هرکدوم توی فایلی

ذخیره بشن و بعد از اون فایل برای گرافیک خوانده بشن و سپس تو محیط گرافیکی به نمایش در بیان... اما متاسفانه گرافیک نرسید! و من(خسروی) مجبور شدم دقیقا دقیقه ی ۹۰ ددلاین پروژه رو دوباره توی یه فایل جمع کنم که بتونم حداقل ورودی و خروجی بگیرم و بدم به کاربر و همه ی الگوریتم ها نشون داده بشن.

- در نتیجه فایل آپلود شده ی پروژه شامل فایل `maigapp.cpp` هست که همه ی الگوریتم ها باهم ران می شن و همینطور اینپوت مخصوص و فایل سی پی پی مخصوص هر الگوریتم که جداگانه ران گرفته بشن هم جواب دارن...

## نحوه ی دقیق پاسخگویی به هر درخواست

### کوتاهترین مسافت

برای پیدا کردن کوتاهترین مسافت نوع وسیله نقلیه و زمان صرف شده اهمیت ندارد پس فقط یال‌هایی با کمترین وزن انتخاب می‌شوند به عبارت دیگر الگوریتم دایجسترا روی گراف ساده مپ اجرا می‌شود.

### کوتاهترین مسافت

برای پیدا کردن کوتاهترین مسافت نوع وسیله نقلیه و زمان صرف شده اهمیت ندارد پس فقط یال‌هایی با کمترین وزن انتخاب می‌شوند به عبارت دیگر الگوریتم دایجسترا روی گراف ساده مپ اجرا می‌شود.

### هزینه

برای پیاده سازی مدل با در نظر گرفتن هزینه‌ها باید از یک گراف جدید استفاده کنیم که در آن باید هزینه صرف شده برای مسافتی که طی می‌کنیم محاسبه شود. همچنین باید در نظر گرفت که هزینه صرف شده برای استفاده از مترو و اتوبوس متفاوت می‌باشد. برنامه قابلیت تشخیص و تفاوت نوع‌های حمل‌ونقل از جمله مترو، اتوبوس، و تاکسی را دارد و بر اساس نوع حمل‌ونقل، وزن و هزینه‌های مرتبط با آن‌ها را محاسبه می‌کند. علت استفاده از یک گراف جدید برای این بخش این است که ما می‌توانیم با تغییر وزن‌های یال این گراف، الگوریتم را خیلی راحت‌تر اجرا کنیم؛ اما مشکلی که ممکن است پیش آید این است که حین تغییر پیدا کردن/نکردن خطوط تاکسی و مترو و یا حتی اتوبوس، به سختی می‌توانیم این روند افزایش و یا تشخیص هزینه ی درست را انجام دهیم.  
راه حل:

می‌توانیم تمام نود های روی یک خط را با یکدیگر مرتبط (یک یال مستقیم) در نظر بگیریم؛ چرا که تعداد ایستگاه‌های بین دو ایستگاه مدنظر ما بر روی آن خط اهمیتی نداشته و تنها ماندن بر روی همان خط کافی می‌باشد.

به همین دلیل مثلاً تمام نودهای خط قرمز رنگ با یکدیگر مرتبط بوده و فایل ورودی تکست که وظیفه ی نگهداری اطلاعات گراف شهر را دارد، یال‌های جدیدی که ایستگاه‌های بین دو ایستگاه مدنظر حذف شده اند را نیز ذخیره می‌کند.

باید توجه داشت که تعداد یال‌ها مهم نیست و در نهایت به تعداد همان ۵۹ ایستگاه، ۵۹ نود (ورتکس) مختلف خواهیم داشت و تنها بر روی یال‌ها مقایسه‌ها صورت می‌گیرند.

زمان

بزرگترین چالش پیاده سازی این قسمت و این درخواست، بررسی مدت زمان پیاده و سوار شدن و یا عوض کردن خطوط بوده که باید با در نظر گرفتن این ها، باز هم مسیری را انتخاب می کردیم که در کوتاه ترین زمان ممکن بتواند به مقصد برسد.

به عنوان راه حل، می توانستیم مجدداً یال هایی با وزن هایی جدید برای گراف تشکیل دهیم. به این صورت که:

الف) وسیله ی نقلیه را عوض کرده ایم:

الف-۱- سوار اتوبوس می شویم:

هزینه ی سوار شدن بر اتوبوس را به وزن جدید یال (مسافت ضرب در سرعت اتوبوس) (وسیله ی نقلیه ی (نقلیه ی آن یال) اضافه می کنیم.

الف-۲- سوار تاکسی می خواهیم بشویم:

هزینه ی سوار شدن بر تاکسی را به وزن جدید یال (مسافت ضرب در سرعت تاکسی) (وسیله ی نقلیه ی آن یال) اضافه می کنیم.

الف-۳- سوار مترو می خواهیم بشویم:

هزینه ی سوار مترو شدن را به وزن جدید یال (مسافت ضرب در سرعت مترو- که در واقع همان مسافت می شود - (وسیله ی نقلیه ی آن یال) اضافه می کنیم.

ب) وسیله ی نقلیه ثابت مانده:

ب-۱- خط تغییر کرده:

تنها هزینه ی تغییر خط به آن یال (هزینه ی زمانی یا وزن هر یال را برابر با ضرب مسافت در سرعت وسیله ی نقلیه در نظر گرفته ایم) اضافه می کنیم.

ب-۲- خط تغییر نکرده: وزن یال تنها به ضرب مسافت در سرعت وسیله ی نقلیه تغییر خواهد کرد.

الگوریتم چگونه از این یال های جدید استفاده می کند؟

دیگر لازم نیست نگران این مورد باشیم که سوار و پیاده شدن ها و یا خط عوض کردن ها چگونه خواهند شد... وزن های جدید و یال های جدید تشکیل شده (که در فایل تکست اینپوت جدیدی درست شده) (کد درست کردن آن فایل زده شده - دستی نوشته نشده! اگر مپ بخواد تغییر کند همه ی اینپوت ها را نیز می توان مجدداً در چند لحظه تغییر داد) و نگهداری می شوند) یک گراف نو با رعایت نکات مربوط به زمان تشکیل خواهند داد که دایجسترا روی آن گراف (با همان تعداد ۵۹ نود یا ورتکس) اعمال خواهد شد.



## پیاده سازی

1. Enum (TransportType) برای نوع وسیله نقلیه

Subway انواع مختلف وسایل نقلیه مانند، Bus و. تعریف شده اند Enum به عنوان یک Taxi

2. در گراف (Vertex) ساختار برای نمایش یک راس

آن (neighbors) تعریف شده است که شامل نام راس و همسایگان Vertex یک ساختار به نام است.

این ساختار برای راحتی کار به صورت پویتری تعریف شده تا دسترسی به المان های آن راحت تر باشد.

2. درست کردن فایل های input جدید برای محاسبه ی بهترین زمان و بهترین قیمت:

ما MakingMoneyInputFile و MakingTimeInputCost در فایل های مختلفی به نام های جدید بر اساس گراف اصلی شهر مان درست می کنیم که یال های آن وزن های جدیدی txt دو فایل بر اساس معیار های گفته شده برای هر درخواست خواهند داشت

3. خواندن اطلاعات گراف از فایل:

همچنین اطلاعات گراف از یک فایل خوانده شده و گراف ساخته شده و رؤس به آن اضافه می شوند

برای تعیین نوع وسیله نقلیه استفاده شده است Enum TransportType از

input.txt

|                               |                           |                           |    |      |      |
|-------------------------------|---------------------------|---------------------------|----|------|------|
| esktop/4022/samhanshat/vscode | ari'ati                   | Rahahan                   | 5  | BUS  | GRAY |
| 2                             | Rahahan                   | Shahrak-e_Shari'ati       | 5  | BUS  | GRAY |
| 3                             | Meydan-e_Hazrat-e_ValiAsr | Mirdamad                  | 11 | BUS  | GRAY |
| 4                             | Mirdamad                  | Meydan-e_Hazrat-e_ValiAsr | 11 | BUS  | GRAY |
| 5                             | Tajrish                   | Shahid_Sadr               | 4  | BUS  | GRAY |
| 6                             | Shahid_Sadr               | Tajrish                   | 4  | BUS  | GRAY |
| 7                             | Tajrish                   | Mirdamad                  | 8  | BUS  | GRAY |
| 8                             | Mirdamad                  | Tajrish                   | 8  | BUS  | GRAY |
| 9                             | Kouhsar                   | Kashani                   | 10 | TAXI | PINK |
| 10                            | Kashani                   | Kouhsar                   | 10 | TAXI | PINK |
| 11                            | Yadegar-e_Emam            | Boostan-e_laleh           | 8  | TAXI | PINK |
| 12                            | Boostan-e_laleh           | Yadegar-e_Emam            | 8  | TAXI | PINK |
| 13                            | Yadegar-e_Emam            | Kashani                   | 6  | TAXI | PINK |
| 14                            | Boostan-e_laleh           | Meydan-e_Hazrat-e_ValiAsr | 2  | TAXI | PINK |
| 15                            | Meydan-e_Hazrat-e_ValiAsr | Haftom-e_Tir              | 2  | TAXI | PINK |
| 16                            | Meydan-e_Hazrat-e_ValiAsr | Boostan-e_laleh           | 2  | TAXI | PINK |
| 17                            | Haftom-e_Tir              | Meydan-e_Hazrat-e_ValiAsr | 2  | TAXI | PINK |
| 18                            | Haftom-e_Tir              | Emam_Hosseini             | 5  | TAXI | PINK |
| 19                            | Emam_Hosseini             | Haftom-e_Tir              | 5  | TAXI | PINK |
| 20                            | Emam_Hosseini             | Meydan-e_Shohada          | 2  | TAXI | PINK |

تصویر 1: نگهداری گراف اصلی شهر و نقشه و همچنین کوتاه ترین مسافت

4. الگوریتم دیجسترا برای یافتن کوتاهترین مسیر:

الگوریتم دیجسترا برای پیدا کردن کوتاهترین مسیر از یک نقطه مبدا به یک نقطه مقصد پیاده‌سازی شده است.

ابتدا یک صف با اولویت بر اساس فاصله از مبدأ ایجاد می‌شود و سپس برای هر نود مجاور، فاصله جدید و اولویت صف بروزرسانی می‌شود.

### 5. تابع اصلی (main):

در تابع اصلی، ابتدا فایل input.txt خوانده شده و گراف بر اساس آن ساخته می‌شود. سپس مبدأ و مقصد برای پیدا کردن مسیر کوتاه تعیین شده و الگوریتم دیجسترا فراخوانی می‌شود.

در نهایت با اجرای برنامه، کوتاهترین مسیر با اطلاعات مربوط به وسایل نقلیه مختلف از نقطه مبدا به نقطه مقصد محاسبه شده و نمایش داده می‌شود. اگر مسیری برای رسیدن از مبدأ به مقصد وجود نداشته باشد، برنامه اطلاع مناسب را چاپ می‌کند.

| inputtime.txt > data |                           |                           |    |        |      |  |
|----------------------|---------------------------|---------------------------|----|--------|------|--|
| 1                    | Shahrak-e_Shari'ati       | Rahahan                   | 20 | BUS    | GRAY |  |
| 2                    | Meydan-e_Hazrat-e_ValiAsr | Mirdamad                  | 44 | BUS    | GRAY |  |
| 3                    | Tajrish                   | Shahid_Sadr               | 16 | BUS    | GRAY |  |
| 4                    | Tajrish                   | Mirdamad                  | 32 | BUS    | GRAY |  |
| 5                    | Kouhsar                   | Kashani                   | 20 | TAXI   | PINK |  |
| 6                    | Kouhsar                   | Kashani                   | 18 | SUBWAY | PINK |  |
| 7                    | Kashani                   | Kouhsar                   | 18 | SUBWAY | PINK |  |
| 8                    | Yadegar-e_Emam            | Boostan-e_laleh           | 16 | TAXI   | PINK |  |
| 9                    | Yadegar-e_Emam            | Boostan-e_laleh           | 16 | SUBWAY | PINK |  |
| 10                   | Boostan-e_laleh           | Yadegar-e_Emam            | 16 | SUBWAY | PINK |  |
| 11                   | Yadegar-e_Emam            | Kashani                   | 14 | SUBWAY | PINK |  |
| 12                   | Boostan-e_laleh           | Meydan-e_Hazrat-e_ValiAsr | 4  | TAXI   | PINK |  |
| 13                   | Boostan-e_laleh           | Meydan-e_Hazrat-e_ValiAsr | 10 | SUBWAY | PINK |  |
| 14                   | Meydan-e_Hazrat-e_ValiAsr | Haftom-e_Tir              | 4  | TAXI   | PINK |  |
| 15                   | Meydan-e_Hazrat-e_ValiAsr | Haftom-e_Tir              | 10 | SUBWAY | PINK |  |
| 16                   | Meydan-e_Hazrat-e_ValiAsr | Boostan-e_laleh           | 10 | SUBWAY | PINK |  |
| 17                   | Haftom-e_Tir              | Meydan-e_Hazrat-e_ValiAsr | 10 | SUBWAY | PINK |  |
| 18                   | Haftom-e_Tir              | Emam_Hosseini             | 10 | TAXI   | PINK |  |
| 19                   | Haftom-e_Tir              | Emam_Hosseini             | 13 | SUBWAY | PINK |  |
| 20                   | Emam_Hosseini             | Haftom-e_Tir              | 13 | SUBWAY | PINK |  |

تصویر 2: یال‌های گراف ذخیره شده برای محاسبه‌ی زمان و پیدا کردن بهترین زمان ممکن

```

MoneyCostInput.txt
1  Shahrak-e_Shari'ati Rahahan 2250    BUS GRAY
2  Rahahan Shahrak-e_Shari'ati 2250    BUS GRAY
3  Shahrak-e_Shari'ati Meydan-e_Hazrat-e_ValiAsr 2250    BUS GRAY
4  Meydan-e_Hazrat-e_ValiAsr Shahrak-e_Shari'ati 2250    BUS GRAY
5  Shahrak-e_Shari'ati Mirdamad 2250    BUS GRAY
6  Mirdamad Shahrak-e_Shari'ati 2250    BUS GRAY
7  Shahrak-e_Shari'ati Tajrish 2250    BUS GRAY
8  Tajrish Shahrak-e_Shari'ati 2250    BUS GRAY
9  Shahrak-e_Shari'ati Shahid_Sadr 2250    BUS GRAY
10 Shahid_Sadr Shahrak-e_Shari'ati 2250    BUS GRAY
11 Shahrak-e_Shari'ati Tajrish 2250    BUS GRAY
12 Tajrish Shahrak-e_Shari'ati 2250    BUS GRAY
13 Shahrak-e_Shari'ati Mirdamad 2250    BUS GRAY
14 Mirdamad Shahrak-e_Shari'ati 2250    BUS GRAY
15 Rahahan Meydan-e_Hazrat-e_ValiAsr 2250    BUS GRAY
16 Meydan-e_Hazrat-e_ValiAsr Rahahan 2250    BUS GRAY
17 Rahahan Mirdamad 2250    BUS GRAY
18 Mirdamad Rahahan 2250    BUS GRAY
19 Rahahan Tajrish 2250    BUS GRAY
20 Tajrish Rahahan 2250    BUS GRAY

```

تصویر 3: یال های گراف ذخیره شده برای محاسبه ی هزینه ی مالی بین هر دو ایستگاه

لینک گیت هاب:

[github](#)

لینک های کمکی:

[dijkstras-algorithm](#)

[csv file management in c++](#)

[unordered map](#)

[reading and writing in csv files in c++](#)