بسمه تعالی

- پروژه پایانی درس ساختمان داده‌ها و مبانی الگوریتم‌ها
- پروژه خود را به تحت قالب یک فایل zip با و با نام StudentNumber.zip در سامانه کوئرا آپلود کنید.
- مهلت ارسال پروژه تا ساعت ۱۱:۵۹ روز جمعه مورخ ۶ بهمن ماه ۱۴۰۰ می باشد.
- در صورتی که درمورد این پروژه سوال یا ابهامی داشتید با ایمیل Aut.dsfall1400@gmail.com با تدریس یاران در ارتباط باشید.
- در روزهای ۷ام و ۸ام بهمن‌ماه این پروژه را باید به تدریسیاران تحویل دهید، توجه داشته باشید که مهلت انجام پروژه با توجه به مهلت ثبت نمرات تمدید نمی‌شود.

همچنین خواهشمند است در متن ایمیل به شماره دانشجویی خود اشاره کنید.

محل قرار

بعد از سپری کردن یک ترم سخت در دانشگاه، سینا تصمیم می‌گیرد تا آخر هفته به کافه برود. از آنجایی که وضع مالی سینا در این چند سال بهتر شده است، تصمیم می‌گیرد تا امیرحسین را نیز همراه خود به کافه ببرد. آن ها با یکدیگر تماس می‌گیرند تا محل کافه را مشخص کنند.

سینا کافه‌ایی را که سر کوچه‌شان است در نظر می‌گیرد، اما امیرحسین به دلیل دور بودن محل کافه از خانه‌اش، با انتخاب آن کافه مخالفت می‌کند.

در ادامه امیرحسین کافه‌ایی را که در نزدیکی محله‌اش است انتخاب می‌کند، اما این بار سینا به دلیل دور بودن محل کافه با آن مخالفت می‌کند.

برای مشخص کردن محل قرار، آن ها می‌خواهند کافه‌ایی را انتخاب کنند که مسیر رسیدن هریک به آن کافه منصفانه باشد. محلی را منصفانه می‌گوییم که در آن قدرمطلق اختلاف مسیرهایی که امیرحسین و سینا طی می‌کنند تا به کافه برسند، کمترین مقدار ممکن باشد.

از شما درخواست شده است تا با گرفتن اطلاعاتی راجب تعداد کافه‌ها، مسیرهای بین کافه‌ها و اندازه‌ی آن‌ها و همچنین موقعیت مکانی سینا و امیرحسین، محلی منصفانه برای قرار این دو نفر پیدا کنید.

این کار را در طی سه مرحله قرار است انجام بدهید.

فاز صفرم (آماده‌سازی)

در گام اول به دنبال بررسی کافه‌ها برای به دست آوردن فاصله آن‌ها از افراد هستیم.

از آنجایی که شبیه‌سازی این مسئله با استفاده از گراف انجام می‌شود، استفاده از حلقه‌هایی نظیر for برای پیمایش Node ها ممکن نمی‌باشد؛ بنابراین شما باید الگوریتمی را پیاده‌سازی کنید تا بتوانیم Node های این گراف را پیمایش کنیم. در این فاز می‌خواهیم تا برای پیمایش Node ها از الگوریتم جستجوی عمقی (DFS) استفاده کنیم.

توضیحات گراف:

برای شبیه‌سازی این مسئله، از یک گراف استفاده می‌کنیم که خیابان‌های شهر، هرکدام یک یال گراف می‌باشند و کافه‌ها در راس‌های گراف قرار دارند و هر کدام با یک شماره متمایز مشخص می‌شوند؛ همچنین وزن هر یال نشان‌دهنده‌ی طول خیابان است.

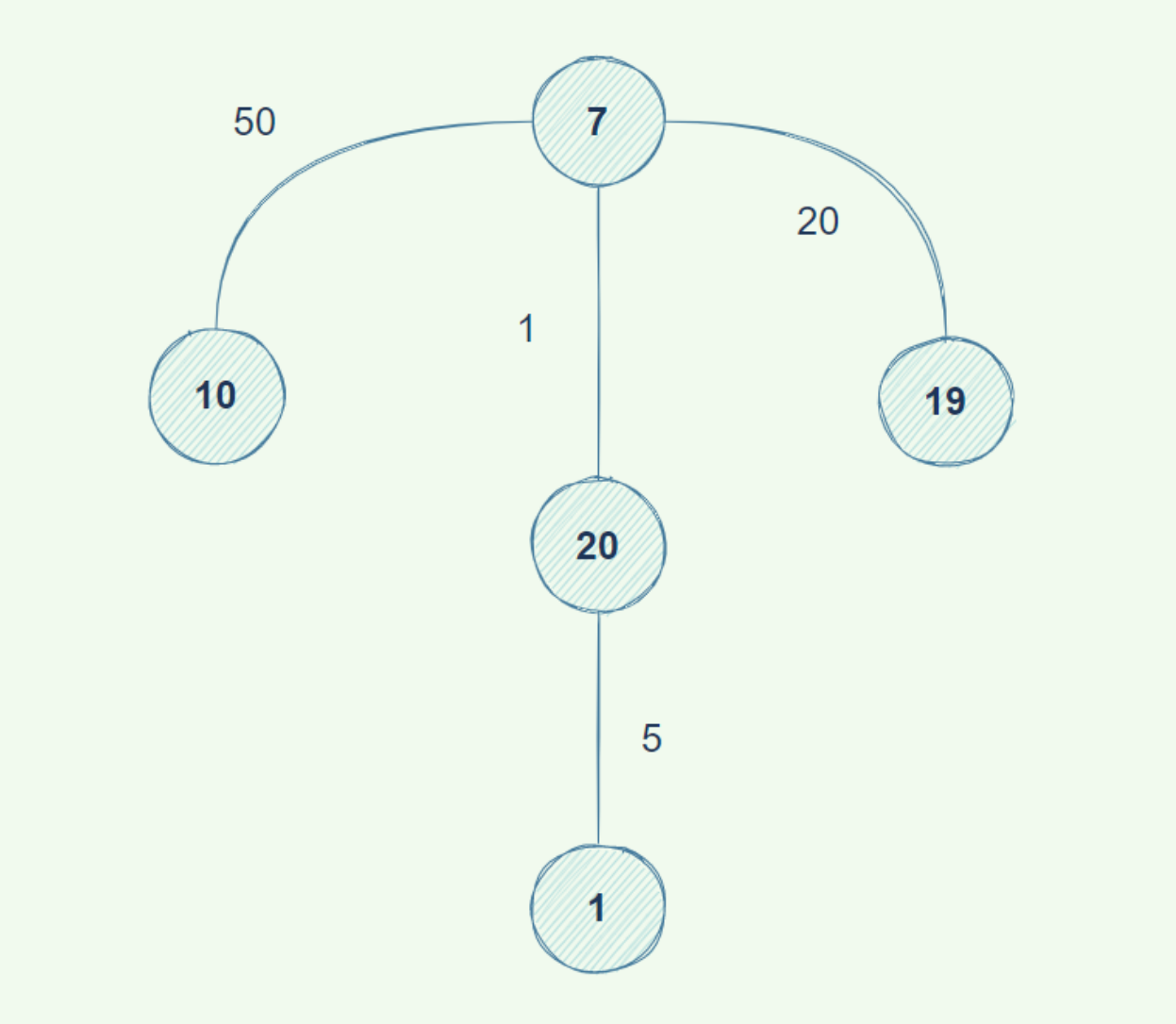
نحوه ورودی دادن اطلاعات را به صورت زیر در نظر بگیرید:

```
> [Number of nodes] [Number of edges]
> [Line of node Numbers]
> [First node number] [Second node number] [cost of the edge between first and second node]
...
...
...
...
```

مثال:

```
5 4
10 7 19 20 1
10 7 50
19 7 20
7 20 1
1 20 5
```

شکل گراف:



برای مطمئن شدن از صحت الگوریتم خود، دستوری به نام test داشته باشید تا زمانی که دستور test را وارد کردیم، الگوریتم پیمایش را اجرا کند و شماره Node ها را در خروجی چاپ کند.

مثال:

```
> test
< 10 1 19 20 7
```

توجه: برای شبیه سازی گراف، استفاده از آرایه مجاز نمی‌باشد و باید از Dynamic list برای ذخیره یال‌ها و الگوریتم‌های Hashing برای مشخص کردن Node ها در پیاده‌سازی گراف استفاده کنید.

فاز اول (شروع)

زمانی که سینا و امیرحسین به کمک شما محل قرار را مشخص کردند، ناگهان خبر قرار گذاشتن این دو نفر به گوش کیان می‌رسد. کیان نیز این خبر را در گروه تلگرامی‌شان منتشر می‌کند؛حال بعد از منتشر شدن خبر؛ کیان، پریا، فرشید، بردیا، آریا و رادین نیز می‌خواهند تا به این قرار ملاقات اضافه شوند.

در هر لحظه ممکن است یک نفر از افراد فوق در گروه اطلاع بدهد که می‌خواهد به قرار ملاقات اضافه شود؛ همچنین ممکن است یکی از افراد اطلاع بدهد که از اضافه شدن به قرار ملاقات منصرف شده و نمی‌تواند به بقیه ملحق بشود.

شما باید این اطلاعات را ذخیره کنید و در هر مرحله یک نقطه‌ی جدید برای محل قرار پیدا کنید که منصفانه باشد.

منصفانه بودن محل قرار، برای تعداد بیش از دو نفر، به این صورت محاسبه می‌شود که، محل انتخابی باید دارای کمترین مقدار حاصل میانگین قدرمطلق اختلاف مسیرهای هر دو نفر باشد.

برای مثال:

$$Selected\ Cafe:\ D$$

$$Sina:\ A,\ Amir:\ B,\ Kian:\ C$$

$$Fair\ score\ of\ D\ =\ \frac{|\ (A,D)-(B,D)\ |+|\ (A,D)-(C,D)\ |+|\ (B,D)-(C,D)\ |}{3}$$

در این مرحله شما با استفاده از اطلاعات داده شده و الگوریتمی که در فاز صفر پیاده کردید، باید یک الگوریتم جدید پیاده‌سازی کنید تا محل قرار منصفانه را، ابتدا برای دو و سپس برای بیش از دو نفر، در این گراف پیدا کند.

شما باید از الگوریتم دایکسترا (dijkstra) برای حل این فاز استفاده کنید.

نحوه ورودی دادن را به صورت زیر درنظر بگیرید:

```
> join [Node number] // Person enters the meeting
> left [Node number] // Person leaves the meeting
```

در این فاز شما باید در هر مرحله شماره یک Node که محل قرار منصفانه است را خروجی بدهید. در صورتی که چند محل قرار منصفانه باشند، شما باید تمامی آن‌ها را در خروجی نشان بدهید.

برنامه را به صورتی بنویسید که با وارد کردن دستور exit خاتمه یابد.

```
> exit
```

برای درک بهتر دایکسترا و شکل کارکرد آن، پیشنهاد می‌شود به لینک زیر مراجعه کنید:

https://www.cs.usfca.edu/~galles/visualization/Dijkstra.html

فاز دوم (بهینه کردن)

در این بخش، می‌خواهیم در ساختار گراف یک تغییر ایجاد کنیم و تعداد محاسبات انجام شده برای یافتن پاسخ را کاهش بدهیم تا به حالت بهینه حل این مسئله دست بیابیم.

زمانی که برای یافتن نقطه منصفانه، الگوریتم دایکسترا را انجام می‌دهید، اگر نتایج به دست آمده برای هر Node را در یک لیست ذخیره کنید،می‌توانید در مراحل بعدی، تنها با انجام یک سری محاسبات، نقطه‌ی منصفانه جدید را پیدا کنید و نیازی به اجرای دوباره الگوریتم دایکسترا ندارید؛ این عمل باعث می‌شود تا زمان مصرفی برای یافتن پاسخ به طرز چشمگیری کاهش بیابد و راه‌حل شما بهینه بشود.

در این فاز شما باید با الگوریتم های فاز صفر و یک، تمام Node ها را پیمایش کنید و الگوریتم دایکسترا را انجام بدهید؛ پس از آن نتایج به دست آمده از الگوریتم دایکسترا برای هر Node را Hash کنید و در لیستی به عنوان یک Property جدید در آن Node ذخیر کنید و پس از آن در ادامه فرآیند، از این نتایج استفاده کنید تا بار محاسباتی کمتری را داشته باشید.

نکته: ساختار class برای هر Node باید یک چیزی شبیه به این باشد

```
Node {
        Node number;
        List of edges connected to this node;
        List of Dijkstra results for this node;
        .....
}
```

توجه: در روز ارائه شما باید توضیح دهید که تابع Hash ایی که پیاده‌سازی کردید چگونه کار می‌کند و در صورت وقوع Collision چگونه آن را مدیریت کردید.

برای آشنایی بیشتر با Hash و روش های Hash کردن، پیشنهاد می‌شود به لینک زیر مراجعه کنید:

https://blog.jscrambler.com/hashing-algorithms