تمرین چهارم درس تحلیل و طراحی الگوریتم

مریم سادات هاشمی
سهیل رستگار
سید صالح اعتمادی

دانشگاه علم و صنعت ـ نیمسال دوم ۹۸ـ۹۷

لطفا به نکات زیر توجه کنید:

- مهلت ارسال این تمرین شنبه ۲۵ اسفند ماه ساعت ۱۱:۵۹ ب.ظ است.
- این تمرین شامل سوال های برنامه نویسی می باشد، بنابراین توجه کنید که حتماً موارد خواسته شده را رعایت کنید.
- نام شاخه، پوشه و پول ریکوست همگی دقیقا "$A4$" باشد.
- در صورتی که به اطلاعات بیشتری نیاز دارید می توانید با ایدی تلگرام @maryam_sadat_hashemi یا @soheil_rategar در ارتباط باشید.
- اگر در حل تمرین شماره ی ۴ مشکلی داشتید،لطفا به این لینک مراجعه کنید و زمانی را برای رفع اشکال تنظیم کنید.

موفق باشید.

توضیحات کلی تمرین

تمرین این هفته ی شما،۳ سوال دارد که باید به همه ی این سوال ها پاسخ دهید. برای حل این سری از تمرین ها مراحل زیر را انجام دهید:

۱. ابتدا مانند تمرین های قبل، یک پروژه به نام $A4$ بسازید.

۲. کلاس هر سوال را به پروژه ی خود اضافه کنید و در قسمت مربوطه کد خود را بنویسید. هر کلاس شامل دو متد اصلی است:

متد اول: تابع solve است که شما باید الگوریتم خود را برای حل سوال در این متد پیاده سازی کنید.

متد دوم: تابع process است که مانند تمرین های قبلی در TestCommon پیاده سازی شده است. بنابراین با خیال راحت سوال را حل کنید و نگران تابع process نباشید! زیرا تمامی پیاده سازی ها برای شما انجام شده است و نیازی نیست که شما کدی برای آن بزنید.

۳. اگر برای حل سوالی نیاز به تابع های کمکی دارید؛ می توانید در کلاس مربوط به همان سوال تابع تان را اضافه کنید.

اکنون که پیاده سازی شما به پایان رسیده است، نوبت به تست برنامه می رسد. مراحل زیر را انجام دهید.

۱. یک UnitTest برای پروژه ی خود بسازید.

۲. فولدر TestData که در ضمیمه همین فایل قرار دارد را به پروژه ی تست خود اضافه کنید.

۳. فایل GradedTests.cs را به پروژه ی تستی که ساخته اید اضافه کنید. توجه کنید که مانند تمرین های قبل، لازم نیست که برای هر سوال TestMethod بنویسید. تمامی آنچه که برای تست هر سوالتان نیاز دارید از قبل در این فایل برای شما پیاده سازی شده است.

دقت کنید که TestCommon تغییر یافته است. بنابراین شما باید نسخه ی جدید آن را با دستورgit Pull دریافت کنید .

۱ Building Roads to Connect Cities

در این مسئله، هدف ساختن جاده هایی بین بعضی از شهر هاست، به طوری که بین هر دو شهر مسیری وجود داشته باشد و طول کل مسیرها کمینه باشد. به یاد داشته باشید که طول پاره خط بین دو نقطه ی $(x_1، y_1)$ و $(x_2، y_2)$ برابر است با $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

خط اول فایل ورودی شامل n ـ تعداد نقاط ـ است. در هر کدام از n خط بعدی نقطه ی $(x_i، y_i)$ را نشان می دهد. در خروجی باید کمینه طول کل مسیرها را چاپ کنید. جواب نهایی خود را به ۶ رقم اعشار رند کنید.

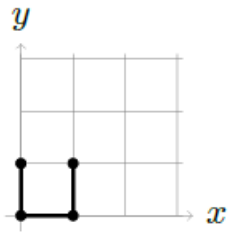
نمونه ۱

ورودی:

```
4
0 0
0 1
1 0
1 1
```

خروجی:

```
3.000000000
```



یک راه بهینه برای اتصال این چهار نقطه را مشاهده می کنید. دقت کنید که راه های دیگری هم برای اتصال این نقاط با پاره خط هایی که در مجموع طول ۳ داشته باشند، وجود دارد.

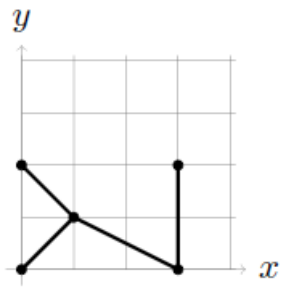
نمونه ۲

ورودی:

```
5
0 0
0 2
1 1
3 0
3 2
```

خروجی:

```
7.064495102
```



یک راه بهینه برای اتصال این پنج نقطه را مشاهده می کنید. در اینجا طول کل برابر است با $2\sqrt{2} + \sqrt{5} + 2$

۲ Clustering

خوشه بندی یک مسئله ی بنیادی در دیتاماینینگ است. هدف تقسیم یک مجموعه از اشیا به زیرمجموعه ها (یا خوشه ها) است به صورتی که اعضای هر زیرمجموعه شبیه یکدیگر باشند، در حالی که اعضای زیرمجموعه های متفاوت از یکدیگر دور باشند. حال با دریافت n نقطه در صفحه، و عدد صحیح k بزرگترین مقدار d را حساب کنید به طوری که نقاط داده شده بتوانند به k زیرمجموعه ی ناتهی تقسیم شوند، به طوری که فاصله ی بین هر دو نقطه از زیرمجموعه های متفاوت حداقل d باشند.

در خط اول n که تعداد نقاط است داده می شود. هرکدام از n خط بعد نقطه ی (y_i,x_i) را نشان می دهد. خط آخر شامل k ـ تعداد خوشه ها ـ است. در خروجی، بزرگترین مقدار ممکن d را چاپ کنید. جواب نهایی خود را به ۶ رقم اعشار رند کنید.

نمونه ۱

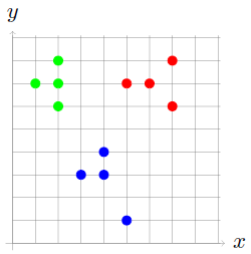
ورودی:

```
12
7 6
4 3
5 1
1 7
2 7
5 7
3 3
7 8
2 8
4 4
6 7
2 6
3
```

خروجی:

```
2.828427124746
```

جواب $\sqrt{8}$ است. دسته بندی مجموعه نقاط داده شده در سه خوشه در شکل زیر نشان داده شده است.



نمونه ۲

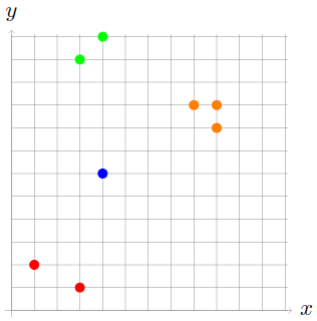
ورودی:

```
8
3 1
1 2
4 6
9 8
9 9
8 9
3 11
4 12
4
```

خروجی:

```
5.000000000
```

جواب ۵ است. دسته بندی مجموعه نقاط داده شده در چهار خوشه در شکل زیر نشان داده شده است.



۳ Compute Distance Faster Using Coordinates

در این مسئله به شما توصیفی از یک شبکه ی واقعی از جاده ها، نه تنها با یال ها و طولشان بلکه همینطور با مختصات راس ها داده می شود. شما باید فاصله ی بین جفت هایی از راس ها را پیدا کنید، اما همزمان باید از اطلاعات اضافه راجع به مختصات راس ها استفاده کنید و سرعت جستجوی خود را بالا ببرید.

در خط اول ورودی دو عدد صحیح n و m ـ تعداد راس ها و یال های موجود در شبکه ـ داده می شود. راس ها از ۱ تا n شماره گذاری می شوند. در هر کدام از n خط بعد مختصات x و y راس متناظر داده شده است. در هرکدام از m خط بعدی شامل سه عدد u و v و l است که نشان می دهد یالی جهت دار با طول l از راس u به راس v وجود دارد. تضمین می شود که طول l بزرگتر مساوی فاصله ی دکارتی بین دو راس u و v است. در خط بعدی عدد صحیح q داده می شود که تعداد سوال هاست. هرکدام از q خط بعد شامل دو عدد صحیح u و v است که شماره ی دو راسی است که باید فاصله ی u به v را حساب کنید.

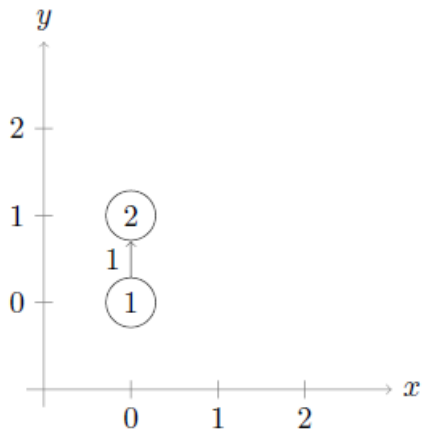
نمونه ۱

ورودی:

```
2 1
0 0
0 1
1 2 1
4
1 1
2 2
1 2
2 1
```

خروجی:

```
0
0
1
-1
```



فاصله ی یک راس تا خودش همواره صفر است. فاصله ی ۱ به ۲ برابر با یک است.
مسیری از ۲ به ۱ وجود ندارد.

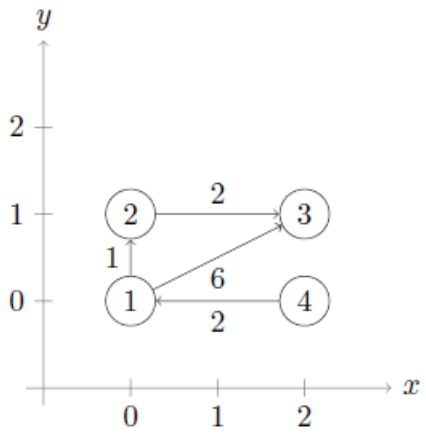
نمونه ۲

ورودی:

```
4 4
0 0
0 1
2 1
2 0
1 2 1
4 1 2
2 3 2
1 3 6
1
1 3
```

خروجی:

```
3
```



یالی مستقیم از راس ۱ به راس ۳ با طول ۶ وجود دارد. اما مسیر کوتاه تر ۱ -> ۲ -> ۳ با طول $1 + 2 = 3$ است.