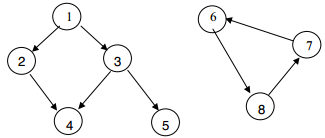
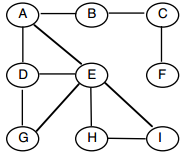
به نام خدا

|  |  |  |
| --- | --- | --- |
| دانشگاه ايلام | ترم اول سال تحصيلي 98-97 | سري پنجم تمرينات ساختمان داده ها |

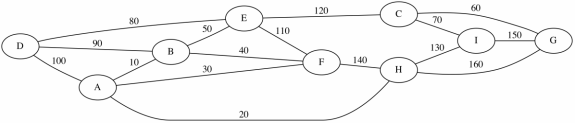
1. گراف: نمایش آرایه‌ی مجاورت و لیست مجاورت گراف زیر را مشخص کنید.



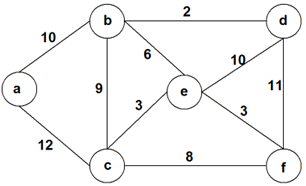
1. پیمایش: پیمایشهای اول عمق و اول سطح گراف زیر را نشان دهید. از نود A شروع کنید.

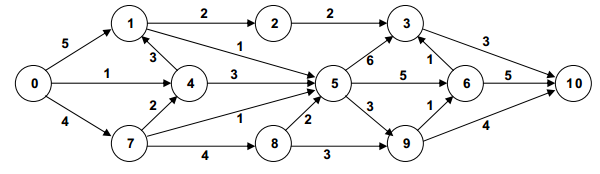


1. MST: برای گراف زیر درخت MST را با استفاده از لگوریتمهای کروسکال، باروفکا و پریم بدست بیاورید.

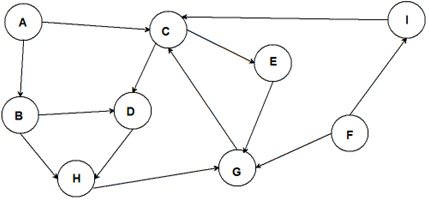


1. با استفاده از الگوریتم دکسترا کوتاهترین مسیرها را از نود ۰ (یا نود a) به بقیه‌ی نودها پیدا کنید. درخت حاصل را رسم کنید. مراحل لازم برای انجام این کار را در یک جدول نشان دهید. اگر در مرحله‌ای چندین انتخاب داشتید از ترتیب عددی یا الفبایی استفاده کنید.





1. با استفاده از الگوریتمهای پریم، کروسکال و باروفکا درخت پوشای مینیم را برای گرافهای فوق بدست آورید. اگر در مرحله‌ای چندین انتخاب داشتید از ترتیب عددی استفاده کنید.
2. برای گراف زیر، ماتریس مجاورت و لیست مجاورت را تعیین کنید. سپس، پیمایشهای اول سطح و اول عمق را تعیین کنید. از نود A شروع کنید. اگر در مرحله‌ای چندین انتخاب داشتید از ترتیب الفبایی استفاده کنید.



1. یک کلاس پایتون به اسم Graph برای نمایش گرافهای بدون جهت پیاده کنید. متدهای زیر را پیاده کنید. جزء سازنده کلاس به صورت زیر است:

**def \_\_init\_\_(self, n)**

این متد یک گراف با گرههای 0 تا n-1 می‌سازد که هیچ یالی ندارد.

کلاس همچنین دارای متدهای زیر است:

**def addEdge(self, i, j)**

این متد یک یال بین نودهای i و j ایجاد می کند به شرطی که 0≤i<n و 0≤j<n و i≠j و قبلاً هیچ یالی بین i و j وجود نداشته باشد.

**def isConnected()**

این متد چک می کند که آیا گراف یکپارچه است یا نه؟

**def clear()**

این متد تمام یالهای گراف را پاک می‌کند.

**def \_\_STR\_\_()**

این متد لیست مجاورت گراف را بر می گرداند. مثلاً، اگر کد زیر را اجرا کنیم:

**g = new Graph(4)**

**g.addEdge(0,1)**

**g.addEdge(0,2)**

**g.addEdge(0,3)**

**g.addEdge(1,3)**

**g.addEdge(2,3)**

**print(g)**

نتیجه ی زیر چاپ خواهد شد:

**From To**

**0: 1,2,3**

**1: 0,3**

**2: 0,3**

**3: 0,1,2**

**def randomize(p)**

این متد تمام یالهای گراف را حذف می‌کند. سپس، برای هر زوج (i,j) که 0 ≤ i < j ≤ n-1  باشد، یک عد تصادفی r بین ۰ و ۱ تولید می کنیم. اگر r ≤ p باشد، یک یال بین یالهای i و j اضافه می‌شود.

**def runTests(size, p0, p1, steps, k)**

این متد الگوریتم زیر را اجرا می‌کند:

Let *d = (p1-p0)/steps* and g = new Graph(size).  
Then: for *p=p0,p0+d,p0+2d,…,p0+(steps-1)d,p1*,   
    count = 0;   
    repeat *k* many times   
        g.randomize();   
        if (g.isConnected()) count++;   
    print *p* and *(count / k)*

لذا، خروجی فراخوانی runTests(20,0.0,0.5,10,60); چیزی شبیه زیر خواهد بود:

**p = 0.000 fraction connected = 0.000**

**p = 0.050 fraction connected = 0.000**

**p = 0.100 fraction connected = 0.050**

**p = 0.150 fraction connected = 0.350**

**p = 0.200 fraction connected = 0.683**

**p = 0.250 fraction connected = 0.917**

**p = 0.300 fraction connected = 0.950**

**p = 0.350 fraction connected = 0.983**

**p = 0.400 fraction connected = 1.000**

**p = 0.450 fraction connected = 1.000**

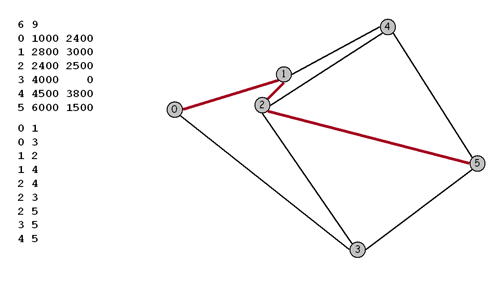
**p = 0.500 fraction connected = 1.000**

**def DFS(a)**

این متد پیمایش اول عمق را از نود a تولید می‌کند.

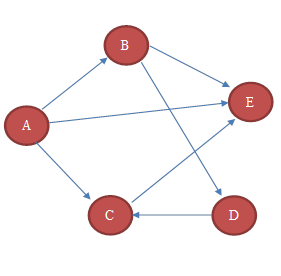
الگوریتمهای دکسترا، بلمان-فورد، پریم و کروسکال را پیاده کنید.

1. در این تمرین الگوریتم کوتاهترین مسیر دکسترا پیاده می گردد. نقشه شهرها در یک فایل داده می‌شود. این فایل شامل دو عدد است که تعداد نودها و یالها را مشخص می‌کند. سپس موقعیت هر گره داده شده است. در نهایت، یالها مشخص شده‌اند. به مثال زیر توجه کنید:



برنامه‌ی شما باید ابتدا اسم فایل را از کاربر بگیرد. سپس، محتویات فایل را در یک گراف بریزد. سپس، شماره‌ی نود منبع و مقصد را از کاربر بگیرید و کوتاهترین مسیر بین آنها را پیدا کنید. مثلاً، خطوط قرمز کوتاهترین مسیر بین نودهای ۰ و ۵ را نشان می‌دهند.

1. تعداد تمام مسیرهای ممکن بین دو راس دلخواه را بشمارید و مسیرها را چاپ کنید! توجه داشته باشید که مسیرها شامل دور نباشند .برای مثال :



Output :

Total paths between A and E are 4

Explanation:

The 4 paths between A and E are:

A -> B -> E

A -> C -> E

A -> E

A -> B -> D -> C -> E

1. فشرده سازی: در **لینک روبرو** [**https://checkoway.net/teaching/cs241/2019-fall/homeworks/hw6.html**](https://checkoway.net/teaching/cs241/2019-fall/homeworks/hw6.html) **توضیحات دقیقی در خصوص انجام فشرده سازی هافمن داده شده است. این روش را طبق این لینک پیاده و آزمایش کنید.**