به نام خدا

گزارش پروژه‌ی نظریه گراف

موضوع پروژه : تطابق ماکسیمم در گراف دوبخشی در زبان پایتون

زهرا دهقانی تفتی (۹۶۲۲۲۰۳۷)

برای پیدا تطابق ماکسیمم در گراف دوبخشی بدون وزن، در زبان پایتون، از کتابخانه‌ی networkx استفاده می‌کنیم. از کتابخانه‌ی networkx ماژول bipartite را فراخوانی می‌کنیم. این ماژول function ها (تابع‌ها) و operation هایی (عملگرهایی) را برای کار با گراف‌های دوبخشی (bipartite) فراهم می‌کند. networkx کلاس سفارشی bipartite graph را ندارد اما کلاس‌های ()Graph یا ()Digraph می‌توانند استفاده بشوند تا گراف‌های دوبخشی را نشان دهند.

برای رسم گراف ورودی و تطابق ماکسیمم از کتابخانه‌ی matplotlib استفاده می‌شود.

حالا می‌خواهیم در مورد پیاده سازی تطابق ماکسیمم و نحوه‌ی کار تابع ()maximum_matching در کتابخانه networkx توضیح دهیم.

در ابتدا تابع plotGraph را تعریف می‌کنیم که برای رسم گراف ورودی و تطابق ماکسیمم استفاده می‌شود و دارای ۳ ورودی می‌باشد. ورودی‌های این تابع به ترتیب زیر است:

- Graph : گراف که می‌خواهیم رسم کنیم.
- ax : مربوط به ویژگی‌های عکس است.
- titile : عنوان شکل خروجی است.

در خط ۱۵ام کد، با استفاده از کتابخانه‌ی networkx که آن را nx نامیدیم و کلاس ()Graph ، گراف دوبخشی g را می‌سازیم. یال‌های گراف ورودی در لیست edges وارد می‌شود.

در خط ۳۶ ام کد، به ازای هر یال موجود در لیست edges، راس های دو سر آن یال، به دو بخش گراف دوبخشی اضافه می‌شود. ویژگی bipartite در تابع ()add_node دارای دو مقدار ۰ و ۱ می‌باشد که این

اعداد نمایانگر بخشی است که هر راس به آن متعلق می‌باشد.(۰ نشان دهنده بخش اول گراف دوبخشی و ۱ نشان دهنده بخش دوم گراف دوبخشی است.)

در خط ٤١ ام کد، یال‌های موجود در لیست egdes به گراف g اضافه می‌شود.

در خط ٤٥ ام کد، با استفاده از ماژول bipartite و تابع ()maximum_matching ، تطابق ماکسیمم در گراف g محاسبه می‌شود. تابع ()maximum_matching یال‌های موجود در تطابق ماکسیمم را به صورت دیکشنری نمایش می‌دهد. مثلا برای کد ما خروجی به صورت زیر است:

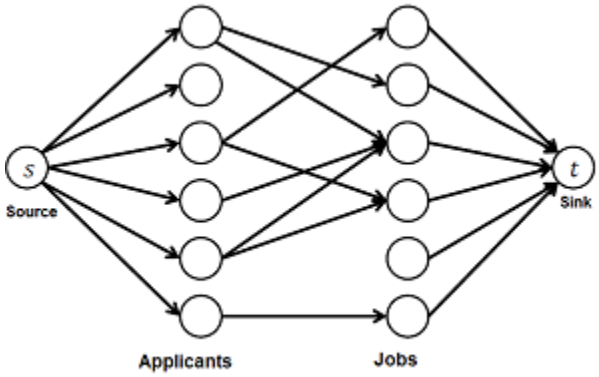
```
C:\Users\Zahra\Graph_Project\venv\Scripts\
python.exe C:/Users/Zahra/Graph_Project/
Max_Matching.py
{(1, 2): (0, 5), (1, 3): (0, 2), (1, 4): (0,
3), (1, 5): (0, 4), (1, 6): (0, 1), (1, 0): (
0, 0), (0, 1): (1, 6), (0, 0): (1, 0), (0, 5
): (1, 2), (0, 4): (1, 5), (0, 3): (1, 4), (0
, 2): (1, 3)}
```

در خط ٥٦ و ٥٩، گراف ورودی و تطابق ماکسیمم با استفاده از تابع ()plotGraph نمایش داده می‌شود.

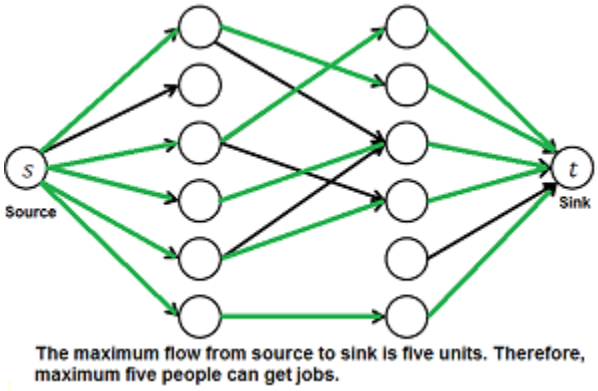
نحوه‌ی کار تابع ()maximum_matching در کتابخانه networkx به صورت زیر است:

مسئله‌ی پیدا کردن تطابق ماکسیمم در گراف دو بخشی می‌تواند به مسئله‌ی بیشترین جریان (max flow) تبدیل شود و به صورت زیر حل شود:

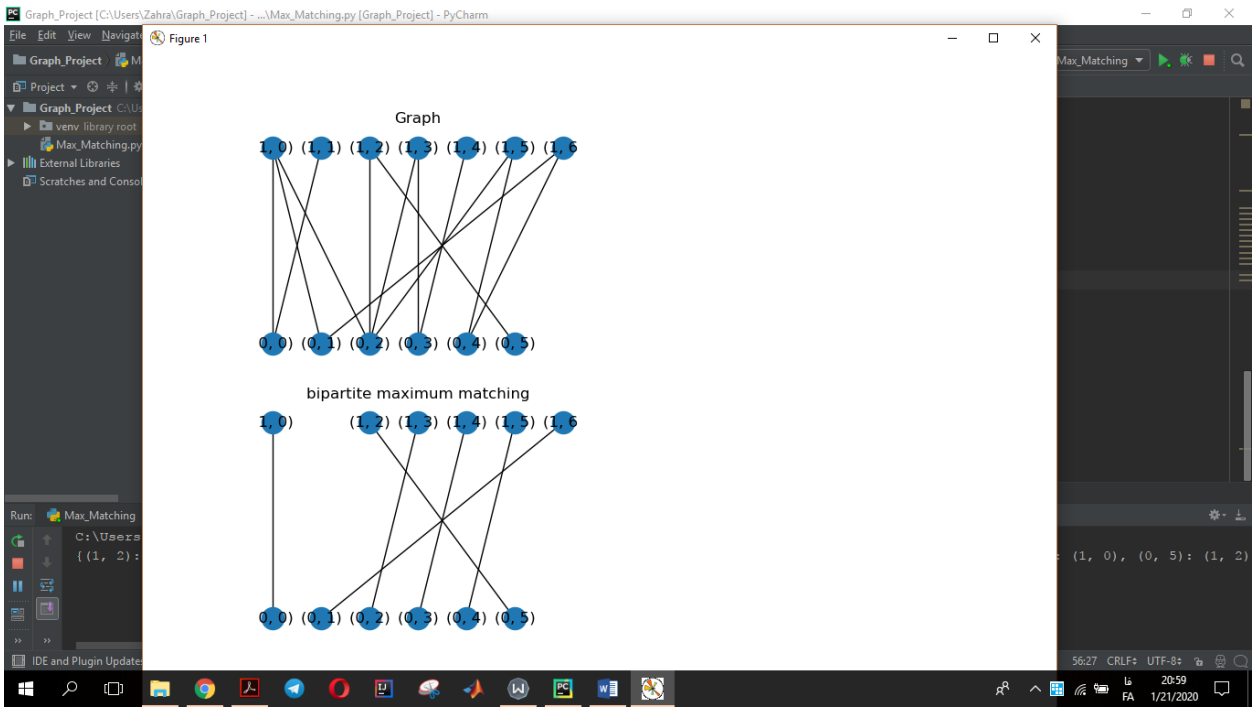
- ساختن یک شبکه جریان : در شبکه‌ی جریان باید یک مبدا و یک مقصد وجود داشته باشد. بنابراین یک راس مبدا اضافه می‌کنیم و یال‌هایی از مبدا به تمام راس‌های بخش پایین گراف دو بخشی اضافه می‌کنیم. همچنین یک راس مقصد اضافه می‌کنیم و از تمام راس‌های بخش بالای گراف دو بخشی یال‌هایی به راس مقصد اضافه می‌کنیم. ظرفیت هر یال را ۱ در نظر می‌گیریم.



- پیدا کردن بیشترین جریان : از روش بیشترین جریان و کمترین برش استفاده می‌کنیم تا جریان بیشینه را در شبکه جریان از راس مبدا به راس مقصد بدست آوریم. جریان بیشینه در واقع همان تطابق ماکسیمم است که ما به دنبال آن بودیم. (در واقع با حذف راس مبدا و راس مقصد و یال‌های متصل به مبدا و مقصد، یال‌های تطابق ماکسیمم بدست می‌آید.)



خروجی کد پروژه Max_Matching.py به صورت زیر می باشد:



در این عکس، شکل اول که عنوان آن "Graph" است، گراف ورودی است که دارای ۷ راس در بخش بالا و ٦ راس در بخش پایین می‌باشد.

شکل دوم که عنوان آن، "bipartite maximum matching" است، تطابق ماکسیمم گراف دوبخشی ورودی را نشان می‌دهد که دارای ٦ یال تطابقی است که حداکثر تعداد یال‌های ممکن برای گراف دوبخشی با ۱۳ راس است و ما می‌دانیم که تعداد یال‌های تطابقی در گراف دوبخشی، کوچکتر یا مساوی تعداد راس‌های هر بخش آن است. در اینجا راس (1,1) غیر تطابقی است و به هیچ یال تطابقی‌ای متصل نیست و بقیه‌ی یال‌ها تطابقی هستند.