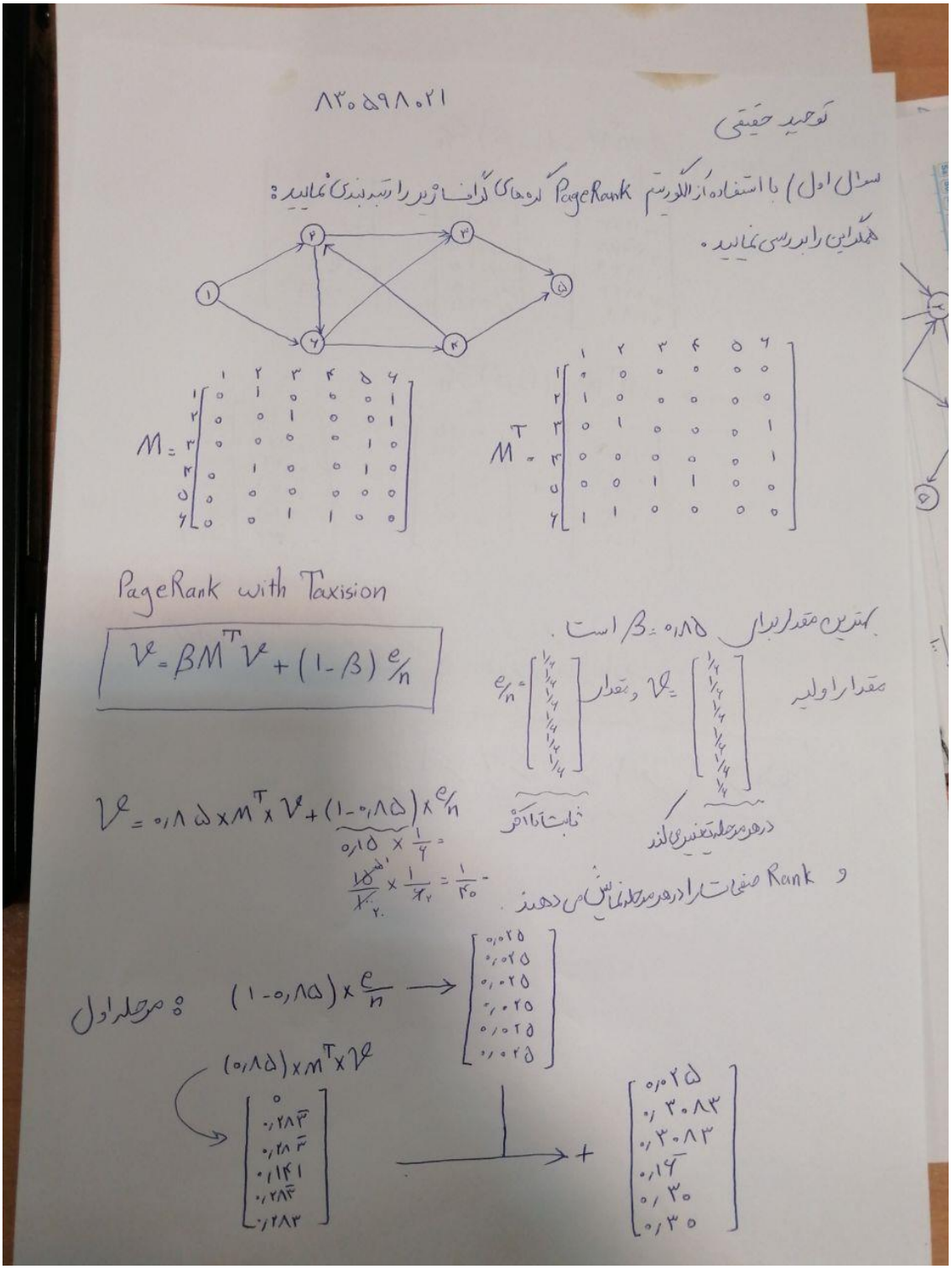
In the name of God

درس : نظریه ریاضی

استاد : دکتر ابراهیمی

دانشجو : توحید حقیقی سیس (۸۳۰۵۹۸۰۲۱)

تمرین اول :


۸۳۰۵۹۸۰۲۱
توحید حقیقی
سوال اول) با استفاده از الگوریتم PageRank گره‌های گراف زیر را رتبه‌بندی نمایید:
همچنین را بررسی نمایید.
PageRank with Taxision
$v' = \beta M^T v + (1-\beta)\frac{e}{n}$
بهترین مقدار برای $\beta = 0.85$ است.
مقدار اولیه v و مقدار $\frac{e}{n}$
$v' = 0.85 \times M^T \times v + \underbrace{(1-0.85)}_{0.15} \times \frac{e}{n}$
$\frac{15}{100} \times \frac{1}{25} = \frac{1}{40}$
ثابت تاکس
در هر مرحله تغییر می‌کند
و Rank صفحات را در هر مرحله مشخص می‌دهد.
مرحله اول:
$(1-0.85) \times \frac{e}{n}$
$(0.85) \times M^T \times v$
+

مرحله دوم ⟵ $\beta M^{T} V + (1-\beta)\, e/n$

$$\begin{bmatrix} 0 \\ 0,1143 \\ 0,3478 \\ 0,1839 \\ 0,2833 \\ 0,1988 \end{bmatrix} + \begin{bmatrix} 0,025 \\ 0,025 \\ 0,025 \\ 0,025 \\ 0,025 \\ 0,025 \end{bmatrix} = \begin{bmatrix} 0,025 \\ 0,139 \\ 0,392 \\ 0,208 \\ 0,308 \\ 0,2238 \end{bmatrix}$$

مرحله سوم ⟵ $\beta M^{T} V + (1-\beta)\, e/n$

$$\begin{bmatrix} 0 \\ 0,1543 \\ 0,237 \\ 0,164 \\ 0,393 \\ 0,107 \end{bmatrix} + \begin{bmatrix} 0,025 \\ 0,025 \\ 0,025 \\ 0,025 \\ 0,025 \\ 0,025 \end{bmatrix} = \begin{bmatrix} 0,025 \\ 0,178 \\ 0,262 \\ 0,171 \\ 0,418 \\ 0,1332 \end{bmatrix} \begin{matrix} \to 1 \\ \to 2 \\ \to 3 \\ \to 4 \\ \to 5 \\ \to 6 \end{matrix}$$

تا این مرحله نمایش داده می‌شود که ۵ بالاترین رنک را دارد.

$$5 > 3 > 2 > 4 > 6 > 1$$

تمرین دوم :

برای پیاده سازی این قسمت یک کلاس نوشتم به نام PageRank و توابع مورد نیاز در آن قرار دارند .

تابع شماره ۱ :

گرفتن ماتریس از ورودی و از کاربر

```
    #ورودی را کابر میگیرد
    def input_array(self):
        #سطر را وارد کن
        row = int(input("Enter the Number of rows:"))
        #ستون را وارد کن
        column = int(input("Enter the Number of columns:"))
        # لیست خالی ایجاد کن
        data = []
        # سط و ستون ها را از ورودی بگیر
        for c in range(column):
            row_list = []
            for r in range(row):
                row_list.append(float(input("Enter value for i={}&j{} = ".format(
c, r))))
            data.append(row_list)
        data_array = np.array(data)
        return data_array
```

و فراخوانی ان در main.py انجام میشود

```
from PageRank import PageRank
import numpy as np

# بهترین مقدار برایalpha=0.85
pagerank=PageRank(0.001,0.85)

adjacency_matrix=pagerank.input_array()
```

حالت اجرا به صورت زیر است :

```
Enter the Number of rows:3
Enter the Number of columns:3
Enter value for i=0&j0 = 1
Enter value for i=0&j1 = 4
Enter value for i=0&j2 = 5
Enter value for i=1&j0 = 1
Enter value for i=1&j1 = 2
Enter value for i=1&j2 = 4
Enter value for i=2&j0 = 5
Enter value for i=2&j1 = 1
Enter value for i=2&j2 = 0
```

تابع شماره ۲ :

در این تابع ماتریس ورودی را ترنس پورت میکنیم و وارون میکنیم

```
    # ترانهاده ماتریس را تولید میکند
    def Transpose_Matrix(self,matrix):
        return matrix.T
```

تابع شماره ۳ :

در این قسمت ماتریس مجاورت را به مارکوف تبدیل مینیم تا جمع سطر ها ۱ شود

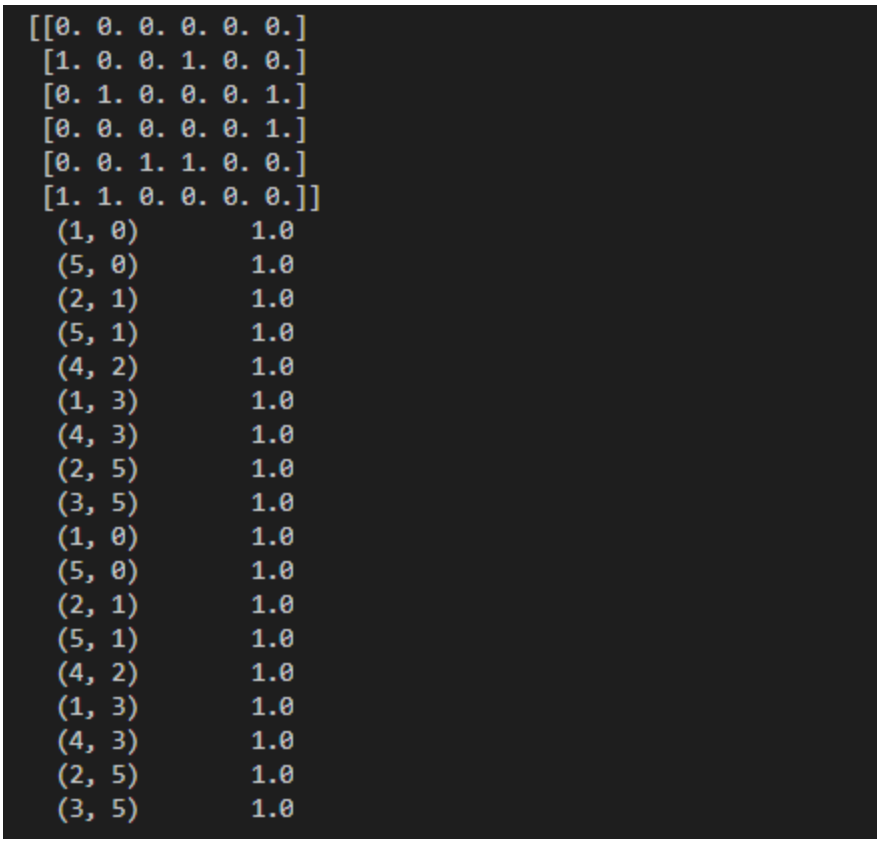
```
    # شود ۱تبدیل ماتریس مجاورت به ماتریس مارکوف که جمع ستون ها
    def ConvertMatrix_To_Markov(self,matrix):
        M = np.zeros(matrix.shape)
        for c in range(matrix.shape[1]):
            s = matrix[:, c].sum()
            if(s!=0):
                M[:, c] = matrix[:, c]/s
        return M
```

تابع شماره ۴ :

برای این که اگر نود ها زیاد باشد ممکن است ماتریس مجاورت ان خیلی بزرگ شود

فقط یال ها را نگه میدارد

```
    #فقط نود های غیر صفر را نگه میدارد
    def Spars_Matrix(self,matrix):
        return sparse.csc_matrix(matrix)
```



خروجی این قسمت به صورت بالا میباشد

تابع شماره ۵ :

در این قسمت تابع اصلی برنامه که کار pagerank را میکند توضشیح میدهیم

تابع پیج رنک راه های زیادی دارد در حالتی که هیچ سطری همه صفر نباشد از فرمول

V=MV میتوانیم استفاده کنیم ولی این فرمول یک مشکل اساسی دارد

وقتی یک نود باشد که به هیچ نود دیگری راه نداشته باشد اگر این فرمول را استفاده کنیم

در اخر رنک همه را صفر میکند

برای رفع این مشکل از فرمول زیر استفاده میکنیم که به ان PageRank With Taxation میگویند و فرمول ان به صورت زیر است

V=BVM+(۱-B)e/n

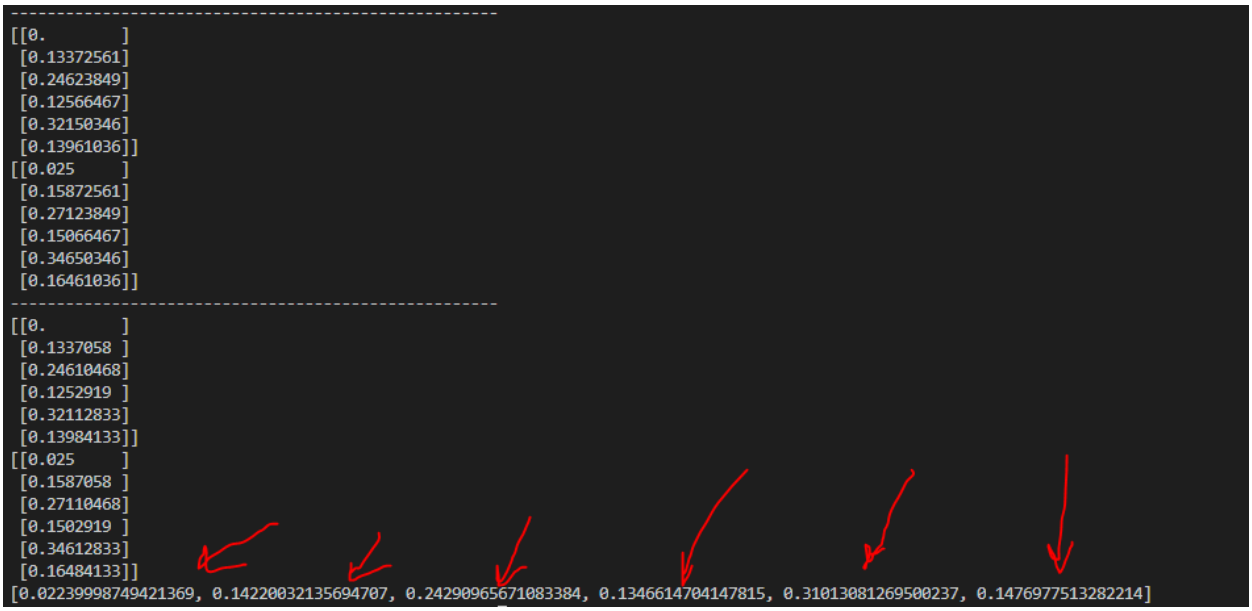
که در این فرمول بهترین مقدار B=۰,۸۵ است و از انجا که e ماتریس تمام ۱ است و n تعداد نود ها است و قسمت دوم فرمول برای جلو گیری از صفر شدن اضافه شده است و هدف این است که اگر به نود هایی رسید که به هیچ نود دیگری راه ندارد به صورت رندوم به نود های دیگر پرش کند .

```
    #تابع پیج رنک
    def PageRank_Function(self,M_Matrix,V_Matrix,EN_Matrix):
        # مقادیر اولیه
        finish_num = 1
        beta = self.beta
        alpha = self.alpha
        while(finish_num>alpha):
            print("-----------------------------------------------------")
            print(beta * (M_Matrix * V_Matrix))
            V_Rank = beta * (M_Matrix * V_Matrix) + (1-beta) * EN_Matrix
            print(V_Rank)

            #scall mikonam ta har marhalle jam 1 shavad
            rescale = 1 / V_Rank.sum()
            V_Rank = V_Rank * rescale
            #ماتریس جدید را از قبلی کم میکند تا شرط پایان را ایجاد کند
            v_d = abs(V_Rank - V_Matrix)
            finish_num = v_d.sum()
            # ابدیت رنک صفحات
            V_Matrix = V_Rank
        #از حالت ماتریسی در میاورد و به ارایه تبدیل میکند
        df_all=V_Rank
        listall_df=list(map(float, df_all))
        #2 تا صفحه برتر را میدهد
        return listall_df
```

و در اخر خروجی به صورت زیر است :

اون قسمت هایی که فلش قرمز دارد رنک هارا نشان میدهد



[[0.
[0.13372561]
[0.24623849]
[0.12566467]
[0.32150346]
[0.13961036]]
[[0.025
[0.15872561]
[0.27123849]
[0.15066467]
[0.34650346]
[0.16461036]]

[[0.
[0.1337058]
[0.24610468]
[0.1252919]
[0.32112833]
[0.13984133]]
[[0.025
[0.1587058]
[0.27110468]
[0.1502919]
[0.34612833]
[0.16484133]]
[0.02239998749421369, 0.1422003213569 4707, 0.24290965671083384, 0.1346614704147815, 0.31013081269500237, 0.1476977513282214]