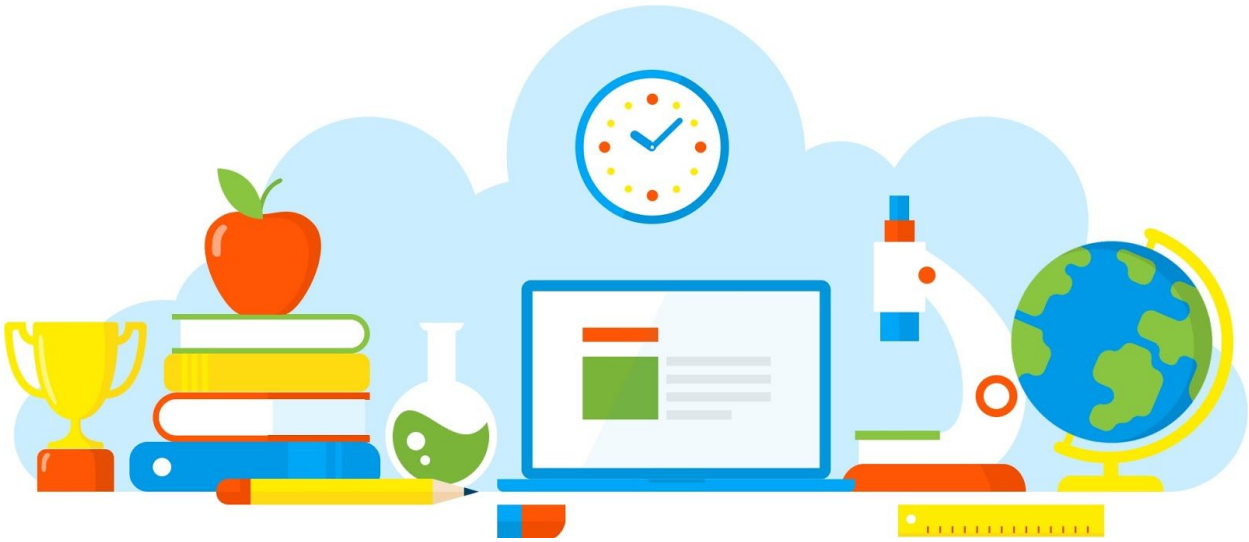

تکلیف ۴ نظریه ریاضی و سیستم‌ها

استاد: دکتر ابراهیمی

مهدی صادقی
۸۳۰۵۹۸۰۳۵

سوال ۱:

در ابتدا ماتریس انتقال گراف (T) را از فرمول زیر بدست می‌آوریم:

$$T_{ij} \begin{cases} \frac{1}{k_j} & i,\ j\ is\ an\ edge \\ 0 & other \end{cases}$$

که در اینجا منظور از k_j درجه‌ی گره‌ی j است. پس داریم:

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 1 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 & 0 & 0 \end{bmatrix}$$

از آنجایی که گره‌ی پنجم گره‌ی خروجی ندارد و اصطلاحا deadblock است، در صورتی که الگوریتم pagerank را روی این ماتریس اجرا کنیم همه‌ی قدم زن‌ها در نهایت به این گره رفته و محو میشوند. برای حل این مشکل ماتریس را به گونه‌ای تغییر میدهیم که قدم زن پس از ورود به گره‌ی پنجم، با احتمال مساوی به یکی از گره‌های دیگر برود. به عبارت دیگر تمام ستون پنجم ماتریس انتقال را برابر ۱/۶ قرار میدهیم.

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{6} & 0 \\ \frac{1}{2} & 0 & 0 & 0.5 & \frac{1}{6} & 0 \\ 0 & \frac{1}{2} & 0 & 0 & \frac{1}{6} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{2} & \frac{1}{6} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{6} & 0 \end{bmatrix}$$

حال ماتریس pagerank را به صورت زیر تعریف میکنیم:

$$P_0 = \begin{bmatrix} \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \\ \frac{1}{6} \end{bmatrix}$$

$$P_{k+1} = \alpha T P_k + (1-\alpha)\frac{1}{n}e$$

که در اینجا e یک ماتریس n*1 است که تمام درایه های آن برابر ۱ باشد. همچنین α یک ضریب است که در اکثر موارد برابر ۰/۸۵ قرار میگیرد.

با جایگذاری در فرمول بالا داریم (برای ۳ تکرار):

$$P_0 = \begin{bmatrix} 0.167 \\ 0.167 \\ 0.167 \\ 0.167 \\ 0.167 \\ 0.167 \end{bmatrix} \quad P_1 = \begin{bmatrix} 0.0448 \\ 0.1808 \\ 0.1808 \\ 0.1128 \\ 0.2488 \\ 0.1808 \end{bmatrix} \quad P_2 = \begin{bmatrix} 0.0566 \\ 0.1236 \\ 0.2102 \\ 0.1334 \\ 0.2582 \\ 0.1524 \end{bmatrix} \quad P_3 = \begin{bmatrix} 0.0576 \\ 0.1383 \\ 0.1749 \\ 0.1333 \\ 0.2930 \\ 0.1341 \end{bmatrix}$$

سوال ۲:

در ابتدا گراف را از کاربر دریافت میکنیم. برای این کار اول تعداد گره‌های گراف را گرفته و پس از آن از کاربر میخواهیم که یال‌ها یکی پس از دیگری وارد کند. پس از آن ماتریس مجاورت ساخته شده و سپس هر درایه غیر صفر آن را بر درجه‌ی خروجی تقسیم میکنیم تا ماتریس انتقال گراف بدست آید:

```
def input_m():
    n = int(input('Enter Matrix Size: '))

    l_matrix_rows = []
    l_matrix_columns = []

    degree_list = [0] * n

    print('!!! enter -1 to end')

    i = 0
    while True:
        s = input('edge {0}: '.format(i))

        if s == 'e':
            break

        node1, node2 = s.split(',')
        node1 = int(node1)
        node2 = int(node2)

        if node1 >= n or node1 < 0 or node1 == node2:
            print('!!! Wrong Node Number')
            continue

        l_matrix_rows.append(node1)
        l_matrix_columns.append(node2)

        degree_list[node1] += 1

        i += 1
```

```
    matrix_data = [1.0] * len(l_matrix_rows)

    for i, node in enumerate(l_matrix_rows):
        matrix_data[i] = 1 / degree_list[node]

    m_matrix = sparse.csr_matrix((matrix_data, (l_matrix_rows,
l_matrix_columns)), shape=(n, n))

    return m_matrix.todense().T
```

پس از آن که ماتریس انتقال گراف را بدست آوردیم، آن را به تابع pagerank میدهیم. در این تابع به صورت Iterative فرمول pagerank که در سوال یک بدست آمده بود را استفاده میکنیم تا یکی از دو شرط توقف زیر برقرار شود:

۱. تعداد ۱۰۰ تکرار اتفاق افتاده باشد.

۲. فاصله‌ی تمام درایه‌های pagerank در دور قبلی با این دور، کمتر از یک حد (در اینجا ۰/۰۱) باشد.

```
def pagerank(m_matrix):
    """
    computes pagerank algorithm on input transition matrix
    :param m_matrix: transition matrix
    :return:
    """
    n = m_matrix.shape[0]

    ranks = np.ones((n, 1)) / n  # initial ranks is same as e/n
    last_ranks = ranks.copy()

    bm_matrix = BETA * m_matrix  # b*M
    bteleport_matrix = (1 - BETA) * (np.ones((n, 1)) / n)  # (1-b) *
(e/n)
    for i in range(MAX_ITERATIONS):
        ranks = (bm_matrix.dot(ranks)) + bteleport_matrix  # v' = bMv
+ (1-b)e/n
        ranks = ranks / ranks.sum()
```

```
        err = np.abs(last_ranks - ranks)
        if err.all() <= STOP_THRESHOLD:
            break

        last_ranks = ranks.copy()

    return ranks  # a n * 1 vector showing pagerank of each node (ith
element shows ith node pagerank)
```