1η Υποχρεωτική Εργασία Στο Μάθημα της Αριθμητικής Ανάλυσης: Άσκηση 4

Όνοματεπώνυμο: Μπαρακλιλής Ιωάννης
ΑΕΜ: 3685

18 Δεκεμβρίου 2020

Ζητούμενο 1: Απόδειξη ότι ο πίνακας G είναι στοχαστικός

Πρέπει να αποδείξω ότι ο πίνακας $\mathbf{G}$ είναι στοχαστικός. Δηλαδη αρκεί να δείξω ότι κάθε στήλη του πίνακα αυτού δίνει άθροισμα 1, δηλαδή να δείξω ότι για κάθε j: $\sum_{i=1}^{n} \mathbf{G}_{(i,j)} = 1$.

Έστω $j \in \{1, 2, ..., n\}$,οπου n είναι το πλήθος των στηλών του πίνακα $\mathbf{A}$:

$$\begin{aligned}\sum_{i=1}^{n} \mathbf{G}_{(i,j)} &= \sum_{i=1}^{n} \frac{q}{n} + \frac{\mathbf{A}_{(j,i)}(1-q)}{n_j} = q\sum_{i=1}^{n} \frac{1}{n} + (1-q)\sum_{i=1}^{n} \frac{\mathbf{A}_{(j,i)}}{n_j} \\ &= q \cdot 1 + (1-q)\frac{\sum_{i=1}^{n} \mathbf{A}_{(j,i)}}{n_j} = q + (1-q)\frac{n_j}{n_j} = q + 1 - q = 1.\end{aligned}$$

Αποδείχθηκε το ζητούμενο.

Ζητούμενο 2: Εύρεση του πίνακα G και του ιδιοδιανύσματος της μέγιστης ιδιοτιμής του

Ζητείται να κατασκευαστεί ο πίνακας $\mathbf{G}$ και να επαληθευτεί με την χρήση της μεθόδου της δυνάμεως ότι το ιδιοδιάνυσμα της μέγιστης ιδιοτιμής είναι αυτό που δίνεται στην εκφώνηση.

Αυτά υλοποιούνται προγραμματιστικά στην γλώσσα python (3.7) στο αρχείο b_G_p_calculation.py το οποίο φαίνεται παρακάτω:

```
import math


def construct_G(A, q, n):
    google_matrix = [[0.0 for i in range(n)] for j in range(n)]

    for i in range(n):
        for j in range(n):
            row_sum = 0
            for k in range(len(A[j])):
                row_sum += A[j][k]

            google_matrix[i][j] = ((q / n) + ((A[j][i] * (1 - q
                )) / row_sum))

    return google_matrix


def matrix_columnMatrix_multiplication(lhs_matrix,
    rhs_columnMatrix):
    result = [0.0 for i in range(len(rhs_columnMatrix))]

    for i in range(len(lhs_matrix)):
        for j in range(len(lhs_matrix[i])):
            result[i] += lhs_matrix[i][j] * rhs_columnMatrix[j]

    return result


def integer_columnMatrix_multiplication(integer, columnMatrix)
    :
    result = []

    for element in columnMatrix:
        result.append(integer * element)

    return result


def power_method(matrix, digits_of_precision):
    p = [1/len(matrix) for i in range(len(matrix))]
    eigenvalue_estimate = 0
```

```
    while True:

        p_new = matrix_columnMatrix_multiplication(matrix, p)

        new_eigenvalue_estimate = 0
        for i in p_new:
            if i != 0:
                new_eigenvalue_estimate = i
                break

        if new_eigenvalue_estimate == 0:
            return p_new

        p_new = integer_columnMatrix_multiplication(1 /
            new_eigenvalue_estimate, p_new)

        if abs(new_eigenvalue_estimate - eigenvalue_estimate) <
             0.5 * (10 ** (-1.0 * digits_of_precision)):
            return p_new

        else:
            p = p_new
            eigenvalue_estimate = new_eigenvalue_estimate


def main():
    A = [
            [0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
            [0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
            [0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0],
            [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
            [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
            [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
            [0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
            [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]
        ]
```

```
    q = 0.15
    n = len(A)

    G = construct_G(A, q, n)

    print("Google␣matrix:")
    for i in G:
        for j in i:
            print("{:.3f}".format(j), end="\t")
        print()

    print("\nSums␣of␣Google␣matrix's␣columns:␣")
    temp = 0
    for i in range(len(G)):
        temp = 0
        for j in range(len(G)):
            temp += G[j][i]
        print(temp, end="\t")
    print("\n")

    p_result = power_method(G, 7)

    element_sum = 0
    for i in p_result:
        element_sum += i

    p_result = integer_columnMatrix_multiplication(1 /
        element_sum, p_result)

    print("Max␣eigenvalue␣eigenvector␣p:␣")
    temp = 0
    for count, i in enumerate(p_result):
        print("p[{:02d}]:".format(count + 1), round(i, 7))
        temp += i

    print("\nWith␣sum␣=", temp)

if __name__ == '__main__':
    main()
```

Στον παραπάνω κώδικα:
Αρχικά, ορίζεται η συνάρτηση construct_G, που δέχεται ως παραμέτρους έναν πίνακα A, έναν πραγματικό αριθμό q που αντιστοιχεί στην πιθανότητα μεταπήδησης και έναν ακόμη αριθμό που αντιστοιχεί στο μέγεθος του πίνακα A,

και επιστρέφει τον πίνακα G που αντιστοιχεί στις παραμέτρους για τα στοιχεία του οποίου ισχύει ότι $G_{(i,j)} = \frac{q}{n} + \frac{A_{(j,i)}(1-q)}{n_j}$ (όπου n_j είναι το άθροισμα της γραμμής j).

Στην συνέχεια, ορίζεται η βοηθητική συνάρτηση matrix_columnMatrix_multiplication, που δέχεται ως παραμέτρους έναν πίνακα και έναν πίνακα στήλη (με ίδιο αριθμό γραμμών με τον πίνακα της άλλης παραμέτρου), και επιστρέφει το γινόμενο των δύο πινάκων και η βοηθητική συνάρτηση integer_columnMatrix_multiplication, που δέχεται ως παραμέτρους έναν ακέραιο και έναν πίνακα στήλη, και επιστρέφει τον πίνακα που είναι αποτέλεσμα πολλαπλασιασμού του ακεραίου με τον πίνακα.

Ακολούθως ορίζω την συνάρτηση power_method, που δέχεται ως παραμέτρους έναν πίνακα και τα απαιτούμενα ψηφία ακρίβειας, και επιστέφει το ιδιοδιάνυσμα που αντιστοιχεί στην μέγιστη ιδιοτιμή του πίνακα της παραμέτρου.

Αναλυτικά:

Ορίζω (αυθαίρετα) ως (και αποθηκεύω στον πίνακα p) αρχική εκτίμηση ιδιοδιανύσματος το ιδιοδιάνυσμα με κάθε του στοιχείο το $\frac{1}{n}$ όπου n είναι το πλήθος των στοιχείων του διανύσματος (το άθροισμα των στοιχείων της αρχικής εκτίμησης αυτής του ιδιοδιανύσματος θα είναι 1) και ορίζω την μεταβλητή eigenvalue_estimate που (θα) αποθηκεύει την εκτίμηση της ιδιοτιμής (την αρχικοποιώ στην τυχαία τιμή 0 που θα αλλάξει στην συνέχεια).

Μετά, ξεκινάει ένας ατέρμονος βρόχος που θα τερματίσει όταν "πετύχω" ακρίβεια εκτίμησης ιδιοτιμής ίση με αυτή που δόθηκε ως παράμετρος όπου:

αποθηκεύω στην (νέα) μεταβλητή p_new την νέα εκτίμηση του ιδιοδιανύσματος που λαμβάνω πολλαπλασιάζοντας τον πίνακα του ορίσματος με την προηγούμενη εκτίμηση του ιδιοδιανύσματος. Στην συνέχεια, ορίζω την νέα εκτίμηση της ιδιοτιμής (μεταβλητή new_eigenvalue_estimate ως το πρώτο μη μηδενικό στοιχείο της εκτίμησης του ιδιοδιανύσματος (αν δεν έχει μη μηδενικά στοιχεία, δηλαδή είναι το μηδενικό διάνυσμα, τότε η νέα εκτίμηση της ιδιοτιμής θα είναι 0 και σε αυτή την περίπτωση σημαίνει ότι βρέθηκε το ιδιοδιάνυσμα μέγιστης ιδιοτιμής το οποίο είναι η νέα εκτίμηση του ιδιοδιανύσματος που ταυτίζεται με το μηδενικό διάνυσμα την οποία επιστρέφω και τερματίζεται η εκτέλεση της συνάρτησης). Ακολούθως, κανονικοποιώ την νέα εκτίμηση του ιδιοδιανύσματος πολλαπλασιάζοντας κάθε στοιχείο του με το αντίστροφο της ιδιοτιμής. Τελικά, ελέγχω άν η διαφορά της νέας και παλιάς εκτίμησης της ιδιοτιμής είναι μικρότερη απο την ζητούμενη ακρίβεια και αν είναι επιστρέφω την νέα εκτίμηση του ιδιοδιανύσματος μέγιστης ιδιοτιμής. Σε διαφορετική περίπτωση, ενημερώνονται οι μεταβλητές των παλιών εκτιμήσεων ιδιοδιανύσματος και ιδιοτιμής αντίστοιχα.

Τέλος ορίζεται η συνάρτηση main, χωρίς παραμέτρους, η οποία θα εκτελε-

στεί όταν θα εκτελέσουμε το παραπάνω αρχείο και εκτελεί τους υπολογισμούς που ζητούνται στην εκφώνηση του ζητούμενου. Αναλυτικά,
Αρχικά, ορίζεται ο πίνακας (γειτνίασης του γραφήματος) A που δίνεται στην εκφώνηση της άσκησης και οι μεταβλητές q, n που αντιστοιχούν στην πιθανότητα μεταπήδησης και μέγεθος του πίνακα A αντίστοιχα. Η μεταβλητή q παίρνει την τιμη 0.15 που δίνεται απο την εκφώνηση.
Στην συνέχεια, υπολογίζεται και αποθηκεύεται στην μεταβλητή G ο πίνακας Google που αντιστοιχεί στον πίνακα A που δημιουργείται με την βοήθεια της συνάρτησης construct_G.
Ακολουθεί εκτύπωση στην οθόνη των στοιχείων (μέχρι 3 δεκαδικά ψηφία) του πίνακα G και το άθροισμα κάθε στήλης αυτού, προκειμένου να επαληθευτεί το γεγονός ότι ο G είναι στοχαστικός.
Μετά, υπολογίζεται και αποθηκεύεται στην μεταβλητή p_result ο πίνακας με το ιδιοδιάνυσμα που αντιστοιχεί την μέγιστη ιδιοτιμή του G (που είναι η 1 εφόσον G στοχαστικός) που στην συνέχεια κανονικοποιείται ώστε το άθροισμα των στοιχείων του να δίνει 1.
Τελικά, τυπώνονται τα στοιχεία (στρογγυλοποιημένα σε 7 δεκαδικά ψηφία) του ιδιοδιανύσματος μέγιστης ιδιοτιμής του G (το οποίο είναι το διάνυσμα με τις τάξεις κάθε σελίδας) και το άθροισμα των στοιχείων του ώστε να επαληθευτεί ότι είναι 1.

Άν εκτελέσουμε το αρχείο θα έχουμε ως αποτελέσματα:

```
Google matrix:
0.010   0.010   0.010   0.010   0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010
0.435   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.293   0.010   0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.293   0.010   0.010   0.010
0.010   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.293   0.010   0.010   0.010
0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.435   0.010   0.010
0.010   0.010   0.010   0.010   0.435   0.435   0.435   0.010   0.293   0.010   0.010   0.010   0.010   0.223   0.010
0.010   0.010   0.010   0.010   0.010   0.435   0.435   0.435   0.010   0.010   0.010   0.293   0.010   0.223   0.010
0.010   0.010   0.010   0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.435
0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.860   0.010   0.010   0.010   0.223   0.010
0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.435   0.010   0.435
0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.860   0.010   0.010   0.223   0.010

Sums of Google matrix's columns:
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

Max eigenvalue eigenvector p:
p[01]: 0.0268246
p[02]: 0.0298611
p[03]: 0.0298611
p[04]: 0.0268246
p[05]: 0.0395872
p[06]: 0.0395872
p[07]: 0.0395872
p[08]: 0.0395872
p[09]: 0.0745644
p[10]: 0.10632
p[11]: 0.10632
p[12]: 0.0745644
p[13]: 0.1250916
p[14]: 0.1163279
p[15]: 0.1250916

With sum = 0.9999999999999998
```

Στο παραπάνω μπορούμε να δούμε ότι ο πίνακας G (με στοιχεία που εμφανίζονται μέχρι 3 δεκαδικά ψηφία) για το συγκεκριμένο παράδειγμα είναι:

$$\mathbf{G} = \begin{bmatrix}
0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.435 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.293 & 0.010 & 0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 \\
0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.435 & 0.435 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.223 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.435 & 0.435 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.223 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.860 & 0.010 & 0.010 & 0.010 & 0.223 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.435 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.860 & 0.010 & 0.010 & 0.223 & 0.010
\end{bmatrix}$$

όπου κάθε στήλη του έχει άθροισμα 1 (επαληθεύοντας το ζητούμενο 1 ότι ο πίνακας G είναι στοχαστικός)

Επίσης, βλέπουμε ότι το ιδιοδιάνυσμα που αντιστοιχεί στην μέγιστη ιδιοτιμή είναι:

$$\mathbf{p} = \begin{bmatrix}
0.0268246 \\
0.0298611 \\
0.0298611 \\
0.0268246 \\
0.0395872 \\
0.0395872 \\
0.0395872 \\
0.0395872 \\
0.0745644 \\
0.10632 \\
0.10632 \\
0.0745644 \\
0.1250916 \\
0.1163279 \\
0.1250916
\end{bmatrix}$$

που έχει σφάλμα σε σχέση με αυτό της εκφώνησης (ως προς την μέγιστη νόρμα) $\simeq 0.26392 \cdot 10^{-}7$.

Ζητούμενο 3: Βελτίωση σημαντικότητας επιλεγμένης σελίδας με ορισμένες αλλαγές στις συνδέσεις

Ζητείται να προστεθούν 4 συνδέσεις και να αφαιρεθεί 1 από αυτές που ήδη υπάρχουν στο γράφημα και να κατασκευαστεί ο νέος πίνακα G έτσι ώστε να βελτιωθεί ο βαθμός σημαντικότητας μιας σελίδας που θα επιλέξω.

Επιλέγω να βελτιώσω την τάξη της σελίδας 1.
Απο την εκφώνηση της άσκησης γνωρίζουμε ότι η σημαντικότητα μία σελίδας δεν κρίνεται αυστηρά και μόνο απο τον αριθμό των σελιδών που "δείχνουν" σε αυτή αλλά λαμβάνεται υπόψιν και το γεγονός ότι όταν μία σελίδα "δείχνει" σε μία άλλη "μεταφέρει" κάποια απο την σημαντικότητα της. Επομένως, με την προσθήκη και αφαίρεση συνδέσεων, στόχος μου είναι να μεγιστοποιήσω την "μεταφορά" σημαντικότητας πρός και να ελαχιστοποιήσω την μεταφορά σημαντικότητας από την σελίδα 1.
Τέλος, αξίζει να σημειωθεί ότι η σημαντικότητα "πηγάζει" απο τον αριθμό εισερχόμενων συνδέσεων ακόμη και αν δεν βασίζεται αποκλειστικά σε αυτόν. Δηλαδή, η σελίδα 13 έχει μεγάλη σημαντικότητα λόγω της άμεσης και έμεσης σύνδεσης της με τις σελίδες 10 και 11 που έχουν μεγάλο αριθμό εισερχόμενων συνδέσεων. Αν για παράδειγμα η σελίδα 10 σταματήσει να "δείχνει" στην 13, αυτή χάνει σημαντικότητα. Ομοίως, αν η 10 αρχίσει να δείχνει σε κάποια άλλη σελίδα εκτός της 13, το "ποσό" σημαντικότητας που θα μεταφερθεί στην 13 μειώνεται γιατί το "ποσό" σημαντικότητας που μεταφέρεται απο την σελίδα 10 εξαρτάται απο τον αριθμό των σελιδών στις οποίες δείχνει η σελίδα 10.

Επομένως, λαμβάνοντας υπόψιν τις παραπάνω παρατηρήσεις:
Αρχικά, προσθέτω συνδέσεις απο την 10 στην 1 και απο την 11 στην 1, έτσι "μεταφέρεται" (κάποια απο την) σημαντικότητα αυτών προς την 1.
Στην συνέχεια, προσθέτω συνδέσεις απο την 15 στην 1, ώστε να "μεταφέρεται" κάποια απο την σημαντικότητα που "πηγαίνει" απο την 11 στην 15 να "πηγαίνει" στην 1, και απο την 2 στην 1, ώστε η σημαντικότητα που "φεύγει" απο την 1 προς στην 2 να "επιστρέφει στην 1". Τέλος, αφαιρώ την σύνδεση απο την 10 στην 13, ώστε όλη η "μεταφορά" σημαντικότητας να γίνεται αποκλειστικά προς την 1.

Επομένως καταλήγουμε με έναν πίνακα γειτνίασης της μορφής:

$$\mathbf{A} = \begin{bmatrix}
0&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
1&0&1&0&1&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&1&0&0&0&1&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&1&1&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1&1&0&1&0&1\\
1&0&0&0&0&0&0&0&0&0&0&1&0&1&0
\end{bmatrix}$$

Προγραμματιστικά, η κατασκευή νέου πίνακα G και ο υπολογισμός νέας τάξης σελιδών γίνεται στην γλώσσα python στο αρχείο c_Graph_Modifications.py το οποίο φαίνεται παρακάτω:

```
def construct_G(A, q, n):
    google_matrix = [[0.0 for i in range(n)] for j in range(n)]

    for i in range(n):
        for j in range(n):
            row_sum = 0
            for k in range(len(A[j])):
                row_sum += A[j][k]

            google_matrix[i][j] = ((q / n) + ((A[j][i] * (1 - q
                )) / row_sum))

    return google_matrix


def matrix_columnMatrix_multiplication(lhs_matrix,
    rhs_columnMatrix):
    result = [0.0 for i in range(len(rhs_columnMatrix))]

    for i in range(len(lhs_matrix)):
        for j in range(len(lhs_matrix[i])):
            result[i] += lhs_matrix[i][j] * rhs_columnMatrix[j]
```

```
    return result


def integer_columnMatrix_multiplication(integer, columnMatrix)
    :
    result = []

    for element in columnMatrix:
        result.append(integer * element)

    return result


def power_method(matrix, digits_of_precision):
    p = [1/len(matrix) for i in range(len(matrix))]
    eigenvalue_estimate = 0

    while True:

        p_new = matrix_columnMatrix_multiplication(matrix, p)

        new_eigenvalue_estimate = 0
        for i in p_new:
            if i != 0:
                new_eigenvalue_estimate = i
                break

        if new_eigenvalue_estimate == 0:
            return p_new

        p_new = integer_columnMatrix_multiplication(1 /
            new_eigenvalue_estimate, p_new)

        if abs(new_eigenvalue_estimate - eigenvalue_estimate) <
             0.5 * (10 ** (-1.0 * digits_of_precision)):
            return p_new

        else:
            p = p_new
            eigenvalue_estimate = new_eigenvalue_estimate


if __name__ == '__main__':
```

```
A = [
    [0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]
]

q = 0.15
n = len(A)

G = construct_G(A, q, n)

print("New␣Google␣matrix:")
for i in G:
    for j in i:
        print("{:.3f}".format(j), end="\t")
    print()

print("\nSums␣of␣new␣Google␣matrix's␣columns:␣")
element_sum = 0
for i in range(len(G)):
    element_sum = 0
    for j in range(len(G)):
        element_sum += G[j][i]
    print(element_sum, end="\t")
print("\n")


p = power_method(G, 7)

element_sum = 0
for i in p:
```

```
        element_sum += i
    p = integer_columnMatrix_multiplication(1 / element_sum, p)

    print("\nNew␣rank␣matrix␣p:␣")
    element_sum = 0
    for count, i in enumerate(p):
        print("p[{:02d}]:".format(count + 1), round(i, 7))
        element_sum += i

    print("\nWith␣sum␣=", element_sum)
```

Στον παραπάνω κώδικα:
Αρχικά, ορίζονται εκ νέου οι συναρτήσεις construct_G, matrix_columnMatrix_multiplication, integer_columnMatrix_multiplication και power_method του αρχείου b_G_p_calculation.py που βρίσκεται στον ίδιο φάκελο Exercise4 μαζί με το τρέχον. Αμέσως μετά βρίσκεται ο εκτελέσιμος κώδικας που θα εκτελεστεί αν εκτελέσουμε το αρχείο: Ορίζεται ο νέος πίνακας (γειτνίασης του γραφήματος) A (σύμφωνα με τα παραπάνω). Στην συνέχεια, κατασκευάζεται ο πίνακας G και τυπώνονται τα στοιχεία του και το άθροισμα των στηλών του (για επαλήθευση ότι είναι στοχαστικός). Στην συνέχεια, υπολογίζεται το ιδιοδιάνυσμα της μέγιστης ιδιοτιμής (που είναι η 1 εφόσον ο G είναι στοχαστικός) και αφού κανονικοποιηθεί αποτελεί τον πίνακα με τις τάξεις των σελίδων. Τέλος εμφανίζονται στην οθόνη οι τάξεις των σελίδων (στρογγυλοποιημένες σε 7 δεκαδικά ψηφία) και το άθροισμα όλων των τάξεων (για επαλήθευση ότι αθροίζουν στο 1 ως πιθανότητες).

Άν εκτελέσουμε το αρχείο έχουμε:

```
New Google matrix:
0.010   0.223   0.010   0.010   0.435   0.010   0.010   0.010   0.010   0.860   0.435   0.010   0.010   0.010   0.293
0.435   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.223   0.010   0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.223   0.010   0.010   0.010   0.010   0.010   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010
0.010   0.223   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.293   0.010   0.010   0.010
0.010   0.010   0.293   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.293   0.010   0.010   0.010
0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.435   0.010   0.010
0.010   0.010   0.010   0.010   0.435   0.435   0.435   0.010   0.293   0.010   0.010   0.010   0.010   0.223   0.010
0.010   0.010   0.010   0.010   0.010   0.435   0.435   0.435   0.010   0.010   0.010   0.293   0.010   0.223   0.010
0.010   0.010   0.010   0.435   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.293
0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.223   0.010
0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.435   0.010   0.293
0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.010   0.435   0.010   0.010   0.223   0.010

Sums of new Google matrix's columns:
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0


New rank matrix p:
p[01]: 0.2043197
p[02]: 0.1090782
p[03]: 0.0432083
p[04]: 0.023598
p[05]: 0.0626258
p[06]: 0.0516891
p[07]: 0.0429321
p[08]: 0.0319954
p[09]: 0.1039295
p[10]: 0.1129677
p[11]: 0.080256
p[12]: 0.0344224
p[13]: 0.016691
p[14]: 0.0314869
p[15]: 0.0507998

With sum = 0.9999999999999999
```

Στο παραπάνω μπορούμε να δούμε ότι ο νέος πίνακας G είναι:

$$\mathbf{G} = \begin{bmatrix}
0.010 & 0.223 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.860 & 0.435 & 0.010 & 0.010 & 0.010 & 0.293 \\
0.435 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.223 & 0.010 & 0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.223 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.223 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 \\
0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 \\
0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.435 & 0.435 & 0.010 & 0.293 & 0.010 & 0.010 & 0.010 & 0.010 & 0.223 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.435 & 0.435 & 0.010 & 0.010 & 0.010 & 0.293 & 0.010 & 0.223 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.293 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.223 & 0.010 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.293 \\
0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.010 & 0.435 & 0.010 & 0.010 & 0.223 & 0.010
\end{bmatrix}$$

όπου κάθε στήλη του έχει άθροισμα 1 (επαληθεύοντας το ζητούμενο 1 ότι ο πίνακας G είναι στοχαστικός)

Επίσης, βλέπουμε ότι το διάνυσμα με τις τάξεις των σελίδων είναι:

$$\mathbf{p} = \begin{bmatrix}
0.2043197 \\
0.1090782 \\
0.0432083 \\
0.023598 \\
0.0626258 \\
0.0516891 \\
0.0429321 \\
0.0319954 \\
0.1039295 \\
0.1129677 \\
0.080256 \\
0.0344224 \\
0.016691 \\
0.0314869 \\
0.0507998
\end{bmatrix}$$

Επομένως, βλέπουμε ότι η τάξη της σελίδας 1 (1ο στοιχείο του διανύσματος με τις τάξεις) είναι 0.2043197 σε σχέση με 0.0268246 που ήταν πρίν τις μεταβολές (απο ζητούμενο 2).
Βλέπουμε μία αύξηση $0.2043197 - 0.0268246 = 0.1774951$.

Ζητούμενο 4: Αλλαγή και μελέτη της πιθανότητα μεταπήδησης q στον νέο γράφο

Ζητείται να γίνει αλλαγή της πιθανότητας μεταπήδησης q σε (α) q = 0.02 και (β) q = 0.6 στον νέο γράφο και να γίνει περιγραφή των αλλαγών στην τάξη σελίδας.

Αυτά υλοποιούνται προγραμματιστικά στην γλώσσα python στο αρχείο d_p_Modifications.py το οποίο φαίνεται παρακάτω:

```
def construct_G(A, q, n):
    google_matrix = [[0.0 for i in range(n)] for j in range(n)]

    for i in range(n):
        for j in range(n):
            row_sum = 0
            for k in range(len(A[j])):
                row_sum += A[j][k]

            google_matrix[i][j] = ((q / n) + ((A[j][i] * (1 - q
                )) / row_sum))

    return google_matrix


def matrix_columnMatrix_multiplication(lhs_matrix,
    rhs_columnMatrix):
    result = [0.0 for i in range(len(rhs_columnMatrix))]

    for i in range(len(lhs_matrix)):
        for j in range(len(lhs_matrix[i])):
            result[i] += lhs_matrix[i][j] * rhs_columnMatrix[j]

    return result


def integer_columnMatrix_multiplication(integer, columnMatrix)
    :
    result = []

    for element in columnMatrix:
        result.append(integer * element)
```

```
    return result


def power_method(matrix, digits_of_precision):
    p = [1/len(matrix) for i in range(len(matrix))]
    eigenvalue_estimate = 0

    while True:

        p_new = matrix_columnMatrix_multiplication(matrix, p)

        new_eigenvalue_estimate = 0
        for i in p_new:
            if i != 0:
                new_eigenvalue_estimate = i
                break

        if new_eigenvalue_estimate == 0:
            return p_new

        p_new = integer_columnMatrix_multiplication(1 /
            new_eigenvalue_estimate, p_new)

        if abs(new_eigenvalue_estimate - eigenvalue_estimate) <
             0.5 * (10 ** (-1.0 * digits_of_precision)):
            return p_new

        else:
            p = p_new
            eigenvalue_estimate = new_eigenvalue_estimate


if __name__ == '__main__':

    A = [
        [0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0],
```

```
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]
]

q = 0.02
n = len(A)

G = construct_G(A, q, n)

p = power_method(G, 7)

element_sum = 0
for i in p:
    element_sum += i
p = integer_columnMatrix_multiplication(1 / element_sum, p)

print("Rank␣matrix␣p␣for␣q␣=␣0.02:␣")
element_sum = 0
for count, i in enumerate(p):
    print("p[{:02d}]:".format(count + 1), round(i, 7))
    element_sum += i

print("\nWith␣sum␣=", element_sum)


q = 0.6
n = len(A)

G = construct_G(A, q, n)

p = power_method(G, 7)

element_sum = 0
for i in p:
    element_sum += i

p = integer_columnMatrix_multiplication(1 / element_sum, p)

print("\nRank␣matrix␣p␣for␣q␣=␣0.6:␣")
element_sum = 0
```

```
    for count, i in enumerate(p):
        print("p[{:02d}]:".format(count + 1), round(i, 7))
        element_sum += i

    print("\nWith␣sum␣=", element_sum)
```

Στον παραπάνω κώδικα:
Αρχικά, ορίζονται εκ νέου οι συναρτήσεις construct_G, matrix_columnMatrix_multiplication, integer_columnMatrix_multiplication και power_method του αρχείου b_G_p_calculation.py που βρίσκεται στον ίδιο φάκελο Exercise4 μαζί με το τρέχον. Αμέσως μετά βρίσκεται ο εκτελέσιμος κώδικας που θα εκτελεστεί αν εκτελέσουμε το αρχείο:
Ορίζεται ο πίνακας (γειτνίασης του γραφήματος) A που είναι πανομοιότυπος με εκείνον του προηγούμενου ζητούμενου (ζητούμενο 3).
Μετά, για κάθε ένα απο τα q (q = 0.02 και q = 0.6): ορίζεται η μεταβλητή q που αποθηκεύει την πιθανότητας μεταπήδησης και η μεταβλητή n που αποθηκεύει το μέγεθος του πίνακα A. Στην συνέχεια, κατασκευάζεται ο πίνακας G. Στην συνέχεια, υπολογίζεται το ιδιοδιάνυσμα της μέγιστης ιδιοτιμής και αφού κανονικοποιηθεί αποτελεί τον πίνακα με τις τάξεις των σελίδων. Τέλος εμφανίζονται στην οθόνη οι τάξεις των σελίδων (στρογγυλοποιημένες σε 7 δεκαδικά ψηφία) και το άθροισμα όλων των τάξεων (για επαλήθευση ότι αθροίζουν στο 1 ως πιθανότητες).

Άν εκτελέσουμε το αρχείο αυτό το αρχείο θα δούμε στην οθόνη:

```
Rank matrix p for q = 0.02:
p[01]: 0.2383283
p[02]: 0.1308345
p[03]: 0.0389395
p[04]: 0.0113301
p[05]: 0.0728329
p[06]: 0.0534987
p[07]: 0.0397358
p[08]: 0.0204016
p[09]: 0.1207504
p[10]: 0.1261982
p[11]: 0.0674096
p[12]: 0.0194326
p[13]: 0.00538
p[14]: 0.016517
p[15]: 0.0384107

With sum = 1.0000000000000002

Rank matrix p for q = 0.6:
p[01]: 0.1201608
p[02]: 0.0716838
p[03]: 0.0573872
p[04]: 0.0510942
p[05]: 0.0569262
p[06]: 0.0574095
p[07]: 0.0549876
p[08]: 0.0554708
p[09]: 0.0731837
p[10]: 0.0893801
p[11]: 0.0871504
p[12]: 0.0586439
p[13]: 0.0457577
p[14]: 0.0575766
p[15]: 0.0631877

With sum = 0.9999999999999999
```

Για $q = 0.02$ έχουμε διάνυσμα με τις τάξεις των σελίδων:

$$\mathbf{p} = \begin{bmatrix} 0.2383283 \\ 0.1308345 \\ 0.0389395 \\ 0.0113301 \\ 0.0728329 \\ 0.0534987 \\ 0.0397358 \\ 0.0204016 \\ 0.1207504 \\ 0.1261982 \\ 0.0674096 \\ 0.0194326 \\ 0.00538 \\ 0.016517 \\ 0.0384107 \end{bmatrix}$$

Συμπεραίνουμε ότι αύξηση έχουν οι σελίδες 1, 2, 5, 6, 9, 10 και μείωση οι υπόλοιπες.
Παρατηρούμε ότι οι σελίδες με μεγάλη (πριν την αλλαγή) τάξη, όπως 1, 2, 9, 10 αυξάνουν την τάξη τους. Παράλληλα, μερικές σελίδες που συνδέονται άμεσα με τις παραπάνω που είναι οι 5 και 6, βλέπουν επίσης βελτίωση στην τάξη τους. Τέλος, άλλες σελίδες που δεν συνδέονται άμεσα με αυτές με μεγάλη (πριν την αλλαγή) τάξη έχουν μείωση της τάξης τους.

Επίσης, για $q = 0.6$ έχουμε διάνυσμα με τις τάξεις των σελίδων:

$$\mathbf{p} = \begin{bmatrix} 0.1201608 \\ 0.0716838 \\ 0.0573872 \\ 0.0510942 \\ 0.0569262 \\ 0.0574095 \\ 0.0549876 \\ 0.0554708 \\ 0.0731837 \\ 0.0893801 \\ 0.0871504 \\ 0.0586439 \\ 0.0457577 \\ 0.0575766 \\ 0.0631877 \end{bmatrix}$$

Συμπεραίνουμε ότι μείωση έχουν οι σελίδες 1, 2, 5, 9, 10 και αύξηση οι υπόλοιπες.

Παρατηρούμε ότι οι σελίδες με μεγάλη (πριν την αλλαγή) τάξη, όπως 1, 2, 9, 10 μειώνουν την τάξη τους. Παράλληλα, μία σελίδα που συνδέεται άμεσα με τις παραπάνω που είναι η 5, βλέπει επίσης μείωση στην τάξη της. Τέλος, άλλες σελίδες που δεν συνδέονται άμεσα με αυτές με μεγάλη (πριν την αλλαγή) τάξη έχουν αύξηση της τάξης τους.

Τέλος, σχετικά με την πιθανότητα μεταπήδησης, αυτή ορίζει την πιθανότητα να βρεθεί ο χρήστης σε οποιαδήποτε σελίδα του γραφήματος ανεξαρτήτως της διασύνδεσης αυτών. Επομένως, δίνει μία "βασική" πιθανότητα να βρεθεί ο χρήστης σε κάποια σελίδα η οποία μετά μεταβάλλεται ανάλογα με τις διασυνδέσεις και έτσι "ρυθμίζει" την σημασία των διασυνδέσεων στην τάξη της σελίδας.
Το παραπάνω επαληθεύεται, αν δούμε τα αποτελέσματα της εκτέλεσης του κώδικα αυτού του ζητουμένου: Όταν μειώνουμε το q απο 0.15 σε 0.02 βλέπουμε ότι οι σελίδες στις οποίες συνδέονται πολλές σελίδες άμεσα ή λιγότερο άμεσα βλέπουν αύξηση της τάξης τους ενώ, σελίδες στις οποίες δεν συνδέονται πολλές άλλες βλέπουν μείωση. Αντίστοιχα, όταν αυξάνουμε το q απο 0.15 σε 0.6 παρατηρούμε το αντίθετο φαινόμενο.

Ζητούμενο 5: Απόπειρα βελτίωσης της σελίδας 11

Ζητείται να μελετηθεί η απόπειρα της σελίδας 11 του αρχικού γραφήματος να βελτιώσει την σχέση της με τον ανταγωνιστή της όταν το αποτέλεσμα των προσπαθειών της μοντελοποιείται με το να έχουμε τον αρχικό πίνακα (γειτνίασης του γραφήματος) A για τον οποίο όμως ισχύει ότι $\mathbf{A}_{(8,11)} = \mathbf{A}_{(12,11)} = 3$.

Επομένως έχουμε τον τροποποιημένο πίνακα γειτνίασης A:

$$\mathbf{A} = \begin{bmatrix}
0&1&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&1&0&1&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&1&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&1&0&0&0\\
1&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&0\\
0&0&0&1&0&0&0&0&0&0&3&0&0&0&0\\
0&0&0&0&0&1&1&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&1&1&0&0&3&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&1&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&1
\end{bmatrix}$$

Το ζητούμενο υλοποιείται προγραμματιστικά στην γλώσσα python στο αρχείο e_11_optimization.py το οποίο φαίνεται παρακάτω:

```
def construct_G(A, q, n):
    google_matrix = [[0.0 for i in range(n)] for j in range(n)]

    for i in range(n):
        for j in range(n):
            row_sum = 0
            for k in range(len(A[j])):
                row_sum += A[j][k]

            google_matrix[i][j] = ((q / n) + ((A[j][i] * (1 - q
                )) / row_sum))

    return google_matrix


def matrix_columnMatrix_multiplication(lhs_matrix,
    rhs_columnMatrix):
    result = [0.0 for i in range(len(rhs_columnMatrix))]

    for i in range(len(lhs_matrix)):
        for j in range(len(lhs_matrix[i])):
            result[i] += lhs_matrix[i][j] * rhs_columnMatrix[j]

    return result


def integer_columnMatrix_multiplication(integer, columnMatrix)
    :
    result = []

    for element in columnMatrix:
        result.append(integer * element)

    return result


def power_method(matrix, digits_of_precision):
    p = [1/len(matrix) for i in range(len(matrix))]
    eigenvalue_estimate = 0
```

```
    while True:

        p_new = matrix_columnMatrix_multiplication(matrix, p)

        new_eigenvalue_estimate = 0
        for i in p_new:
            if i != 0:
                new_eigenvalue_estimate = i
                break

        if new_eigenvalue_estimate == 0:
            return p_new

        p_new = integer_columnMatrix_multiplication(1 /
            new_eigenvalue_estimate, p_new)

        if abs(new_eigenvalue_estimate - eigenvalue_estimate) <
             0.5 * (10 ** (-1.0 * digits_of_precision)):
            return p_new

        else:
            p = p_new
            eigenvalue_estimate = new_eigenvalue_estimate


if __name__ == '__main__':

    A_modified = [
        [0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
        [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 3, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]
    ]
```

```
q = 0.15
n = len(A_modified)

G = construct_G(A_modified, q, n)

p = power_method(G, 7)

element_sum = 0
for i in p:
    element_sum += i
p = integer_columnMatrix_multiplication(1 / element_sum, p)

print("New␣rank␣matrix␣p:␣")
element_sum = 0
for count, i in enumerate(p):
    print("p[{:02d}]:".format(count + 1), round(i, 7))
    element_sum += i

print("\nWith␣sum␣=", element_sum)
```

Στον παραπάνω κώδικα:
Αρχικά, ορίζονται εκ νέου οι συναρτήσεις construct_G, matrix_columnMatrix_multiplication, integer_columnMatrix_multiplication και power_method του αρχείου b_G_p_calculation.py που βρίσκεται στον ίδιο φάκελο Exercise4 μαζί με το τρέχον. Αμέσως μετά βρίσκεται ο εκτελέσιμος κώδικας που θα εκτελεστεί αν εκτελέσουμε το αρχείο:
Ακολούθως, ορίζεται ο πίνακας (γειτνίασης του γραφήματος) A που είναι πανομοιότυπος με εκείνον του ζητούμενου 2 με εξαίρεση τα στοιχεία $\mathbf{A}_{(8,11)}$ και $\mathbf{A}_{(12,11)}$ για τα οποία ισχύει: $\mathbf{A}_{(8,11)} = \mathbf{A}_{(12,11)} = 3$.
Μετά, ορίζεται η μεταβλητή q που αποθηκεύει την πιθανότητας μεταπήδησης (που έχει οριστεί απο την εκφώνηση σε 0.15) και η μεταβλητή n που αποθηκεύει το μέγεθος του πίνακα A. Στην συνέχεια, κατασκευάζεται ο πίνακας G. Στην συνέχεια, υπολογίζεται το ιδιοδιάνυσμα της μέγιστης ιδιοτιμής (1) και αφού κανονικοποιηθεί αποτελεί τον πίνακα με τις τάξεις των σελίδων. Τέλος εμφανίζονται στην οθόνη οι τάξεις των σελίδων (στρογγυλοποιημένες σε 7 δεκαδικά ψηφία) και το άθροισμα όλων των τάξεων (για επαλήθευση ότι αθροίζουν στο 1 ως πιθανότητες).

Άν εκτελέσουμε το αρχείο αυτό το αρχείο θα δούμε στην οθόνη:

```
New rank matrix p:
p[01]: 0.0265505
p[02]: 0.0283717
p[03]: 0.0250156
p[04]: 0.0164163
p[05]: 0.0389424
p[06]: 0.0379915
p[07]: 0.0311454
p[08]: 0.0301945
p[09]: 0.0737779
p[10]: 0.1028933
p[11]: 0.1240084
p[12]: 0.0770987
p[13]: 0.1235151
p[14]: 0.1226157
p[15]: 0.141463

With sum = 0.9999999999999998
```

Επομένως έχουμε διάνυσμα με τάξεις σελίδων:

$$\mathbf{p} = \begin{bmatrix} 0.0265505 \\ 0.0283717 \\ 0.0250156 \\ 0.0164163 \\ 0.0389424 \\ 0.0379915 \\ 0.0311454 \\ 0.0301945 \\ 0.0737779 \\ 0.1028933 \\ 0.1240084 \\ 0.0770987 \\ 0.1235151 \\ 0.1226157 \\ 0.141463 \end{bmatrix}$$

Απο το παραπάνω, παρατηρούμε ότι η τάξη της σελίδας 11 είναι 0.1240084 και της 10 είναι 0.1028933. Πρίν την αλλαγή η σελίδα 11 είχε τάξη 0.10632 και η 10 τάξη 0.10362. Επομένως, μπορούμε να συμπεράνουμε ότι η σελίδα 11 πέτυχε τον στόχο της και αύξησε την τάξη της κατά $0.1240084 - 0.10632 = 0.0176884$ ενω ο ανταγωνιστής της, η σελίδα 10, παρατήρησε μείωση: $0.1028933 - 0.10632 = -0.0034267$.

Ζητούμενο 6: Αποτέλεσμα διαγραφής της σελίδας 10

Ζητείται να μελετηθεί επίδραση της διαγραφής της σελίδας 10 από το γράφημα. Σε αυτή την περίπτωση αφαιρείται η 10η γραμμή και 10η στήλη του αρχικού πίνακα γειτνίασης. Επομένως, ο τροποποιημένος πίνακας γειτνίασης θα είναι ο:

$$\mathbf{A} = \begin{bmatrix} 0&1&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&1&0&1&0&1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&1&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1 \\ 0&0&0&0&0&0&0&0&0&0&1&0&1&0 \end{bmatrix}$$

Το ζητούμενο υλοποιείται προγραμματιστικά στην γλώσσα python στο αρχείο f_10_deleted.py το οποίο φαίνεται παρακάτω:

```
def construct_G(A, q, n):
    google_matrix = [[0.0 for i in range(n)] for j in range(n)]

    for i in range(n):
        for j in range(n):
            row_sum = 0
            for k in range(len(A[j])):
                row_sum += A[j][k]

            google_matrix[i][j] = ((q / n) + ((A[j][i] * (1 - q
                )) / row_sum))

    return google_matrix


def matrix_columnMatrix_multiplication(lhs_matrix,
    rhs_columnMatrix):
    result = [0.0 for i in range(len(rhs_columnMatrix))]
```

```
    for i in range(len(lhs_matrix)):
        for j in range(len(lhs_matrix[i])):
            result[i] += lhs_matrix[i][j] * rhs_columnMatrix[j]

    return result


def integer_columnMatrix_multiplication(integer, columnMatrix)
    :
    result = []

    for element in columnMatrix:
        result.append(integer * element)

    return result


def power_method(matrix, digits_of_precision):
    p = [1/len(matrix) for i in range(len(matrix))]
    eigenvalue_estimate = 0

    while True:

        p_new = matrix_columnMatrix_multiplication(matrix, p)

        new_eigenvalue_estimate = 0
        for i in p_new:
            if i != 0:
                new_eigenvalue_estimate = i
                break

        if new_eigenvalue_estimate == 0:
            return p_new

        p_new = integer_columnMatrix_multiplication(1 /
            new_eigenvalue_estimate, p_new)

        if abs(new_eigenvalue_estimate - eigenvalue_estimate) <
             0.5 * (10 ** (-1.0 * digits_of_precision)):
            return p_new

        else:
            p = p_new
            eigenvalue_estimate = new_eigenvalue_estimate
```

```
if __name__ == '__main__':

    A_modified = [
        [0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
        [0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]
    ]


    q = 0.15
    n = len(A_modified)

    G = construct_G(A_modified, q, n)

    p = power_method(G, 7)

    element_sum = 0
    for i in p:
        element_sum += i
    p = integer_columnMatrix_multiplication(1 / element_sum, p)

    print("\nNew␣rank␣matrix␣p:␣")
    element_sum = 0
    for i in range(9):
        print("p[{:02d}]:".format(i + 1), round(p[i], 7))
        element_sum += p[i]
    for i in range(9, len(p)):
        print("p[{:02d}]:".format(i + 2), round(p[i], 7))
        element_sum += p[i]

    print("\nWith␣sum␣=", element_sum)
```

Στον παραπάνω κώδικα:
Αρχικά, ορίζονται εκ νέου οι συναρτήσεις construct_G, matrix_columnMatrix_multiplication, integer_columnMatrix_multiplication και power_method του αρχείου b_G_p_calculation.py που βρίσκεται στον ίδιο φάκελο Exercise4 μαζί με το τρέχον. Αμέσως μετά βρίσκεται ο εκτελέσιμος κώδικας που θα εκτελεστεί αν εκτελέσουμε το αρχείο: Ορίζεται ο πίνακας (γειτνίασης του γραφήματος) A που είναι πανομοιότυπος με εκείνον του ζητούμενου 2 με εξαίρεση το ότι λείπει η γραμμή και στήλη 10. Μετά, ορίζεται η μεταβλητή q που αποθηκεύει την πιθανότητας μεταπήδησης (που έχει οριστεί απο την εκφώνηση σε 0.15) και η μεταβλητή n που αποθηκεύει το μέγεθος του πίνακα A. Στην συνέχεια, κατασκευάζεται ο πίνακας G. Στην συνέχεια, υπολογίζεται το ιδιοδιάνυσμα της μέγιστης ιδιοτιμής (1) και αφού κανονικοποιηθεί αποτελεί τον πίνακα με τις τάξεις των σελίδων. Τέλος εμφανίζονται στην οθόνη οι τάξεις των σελίδων (στρογγυλοποιημένες σε 7 δεκαδικά ψηφία) και το άθροισμα όλων των τάξεων (για επαλήθευση ότι αθροίζουν στο 1 ως πιθανότητες).

Άν εκτελέσουμε το αρχείο αυτό το αρχείο θα δούμε στην οθόνη:

```
New rank matrix p:
p[01]: 0.047095
p[02]: 0.0409114
p[03]: 0.0359356
p[04]: 0.03207
p[05]: 0.0428008
p[06]: 0.041391
p[07]: 0.0516587
p[08]: 0.0502489
p[09]: 0.0482235
p[11]: 0.1709627
p[12]: 0.1035981
p[13]: 0.0411619
p[14]: 0.1074622
p[15]: 0.1864802

With sum = 1.0
```

(σημείωση: οι δείκτες i του p[i] που εμφανίζονται στο παραπάνω στιγμι-

ότυπο και στο αποτέλεσμα εκτέλεσης του αρχείου αντιστοιχούν σε αρίθμηση σελιδών (στοιχεία του γραφήματος) και όχι θέση στο διάνυσμα με τάξεις)

Επομένως έχουμε διάνυσμα με τάξεις σελίδων:

$$\mathbf{p} = \begin{bmatrix} 0.047095 \\ 0.0409114 \\ 0.0359356 \\ 0.03207 \\ 0.0428008 \\ 0.041391 \\ 0.0516587 \\ 0.0502489 \\ 0.0482235 \\ 0.1709627 \\ 0.1035981 \\ 0.0411619 \\ 0.1074622 \\ 0.1864802 \end{bmatrix}$$

Απο τα παραπάνω αποτελέσματα παρατηρούμε ότι για τάξεις των σελιδών (σε σχέση με αυτές που πήραμε ως αποτέλεσμα στο ζητούμενο 2) ισχύει ότι οι τάξεις των σελιδών 9, 13 και 14 μειώνονται ενώ αυξάνονται οι τάξεις όλων των υπολοίπων σελιδών.