2η Εργαστηριακή Άσκηση

Ονοματεπώνυμο: Γεώργιος Γιάτσος
ΑΜ: 3202
Μάθημα: Θεωρία Γραφημάτων
Διδάσκων: Ιωσήφ Πολενάκης

Σε αυτή την άσκηση, θα μελετήσουμε τη στρατηγική τοποθέτηση της ομάδας κρούσης και της αποθήκης ανεφοδιασμού μιας αυτοκρατορίας του Μεσαίωνα, σε μία εποχή όπου μάχονται δύο αυτοκρατορίες για το ποια θα επικρατήσει. Η κύρια στρατηγική της αυτοκρατορίας είναι «να συγκεντρώσει μια μεγάλη ομάδα κρούσης σε κάποιο σημείο, ώστε ανά πάσα στιγμή να μπορεί να σπεύσει προκειμένου να ενισχύσει την άμυνα κάποιου άλλου σημείου το οποίο θα δέχεται επίθεση». Αυτό μεταφράζεται σε ένα γράφημα ως το σημείο το οποίο πρέπει να βρίσκεται πιο κοντά από το καθένα σημείο του γραφήματος, ώστε να μεταβαίνουν από το σημείο αυτό προς κάθε άλλο όσο το δυνατόν πιο γρήγορα. Ενώ, για την αποθήκη ανεφοδιασμού αποφασίζεται να τοποθετηθεί σε κάποιο καίριο σημείο ώστε να ανεφοδιάζονται μέσω αυτού (να μεταβαίνουν δηλαδή σε αυτό) οι μαχόμενοι των υπόλοιπων καίριων σημείων. Συνεπώς, εμείς σαν εξειδικευμένοι αναλυτές πληροφορίας της αντίπαλης αυτοκρατορίας καλούμαστε να εντοπίσουμε την τοποθεσία αυτών των σημείων σύμφωνα με τις παραπάνω πληροφορίες.

Έτσι, η τοποθεσία της ομάδα κρούσης θα βρίσκεται στο σημείο με την ελάχιστη κεντρικότητα :

- ➔ $e(v) = \max\{d(v,u) \mid u \in V(G)\}$, eccentricity

δηλαδή στο κέντρο του γραφήματος, καθώς με αυτόν τον τρόπο η ομάδα κρούσης θα έχει τη δυνατότητα να μεταβεί πιο γρήγορα προς κάθε άλλο σημείο (u), αφού το σημείο αυτό (v) έχει την μικρότερη μέγιστη απόσταση μεταξύ δύο σημείων (v,u) ενός γραφήματος.

Επιπλέον, η τοποθεσία της αποθήκης ανεφοδιασμού θα βρίσκεται στο σημείο με τη μικρότερη απόσταση κορυφής (vertex distance):

- ➔ $dist(v) = \sum_{u \in V(G)} \mathrm{dist}(v,u)$, vertex distance

δηλαδή στο μέσο (median) του γραφήματος, καθώς με αυτόν τον τρόπο κάθε μαχόμενη ομάδα οποιουδήποτε σημείου (u) θα έχει τη δυνατότητα να μεταβεί πιο γρήγορα προς αυτό το σημείο (v) για ανεφοδιασμό, αφού το σημείο αυτό (v) έχει το μικρότερο άθροισμα αποστάσεων προς όλα τα υπόλοιπα σημεία (u) του γραφήματος.

Για τον υπολογισμό των σημείων αυτών αρχικά, ορίζω το γράφημα χρησιμοποιώντας τη βιβλιοθήκη NetworkX, απεικονίζω τους κόμβους του γραφήματος χρησιμοποιώντας τη βιβλιοθήκη Matplotlib και στη συνέχεια εκτελώ ανάλυση του γραφήματος για να βρω τον διάμεσο και τον κεντρικό κόμβο του γραφήματος.

Ο γράφος ορίζεται χρησιμοποιώντας τον πίνακα γειτνίασης με τη μορφή πίνακα NumPy. Το γράφημα δημιουργείται με τη χρήση της μεθόδου from_numpy_array() της βιβλιοθήκης NetworkX. Στη συνέχεια, οι κόμβοι επανασημειώνονται χρησιμοποιώντας τη μέθοδο relabel_nodes() της βιβλιοθήκης NetworkX για να δώσω τα ονόματα της άσκησης στους κόμβους. Τα νέα ονόματα των κόμβων ορίζονται στο λεξικό αντιστοίχισης.

```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np

# Create the graph from the adjacency matrix
G = nx.from_numpy_array(np.array([[0,0,0,1,0,0,0,0],
                                  [0,0,0,0,0,0,0,1],
                                  [0,0,0,1,0,0,0,0],
                                  [1,0,1,0,1,1,1,0],
                                  [0,0,0,1,0,0,0,0],
                                  [0,0,0,1,0,0,0,0],
                                  [0,0,0,1,0,0,0,1],
                                  [0,1,0,0,0,0,1,0]]))

# Define the mapping of old node names to new node names
mapping = {0: "S1", 1: "S2", 2: "S3", 3: "S4", 4: "S5", 5: "S6", 6: "S7", 7: "S8"}

# Relabel the nodes with the new names
G = nx.relabel_nodes(G, mapping)
```

Η γραφική παράσταση απεικονίζεται με τη χρήση της συνάρτησης spring_layout() της βιβλιοθήκης NetworkX, η οποία υπολογίζει τις θέσεις των κόμβων και στη συνέχεια με τις συναρτήσεις draw_networkx_nodes(), draw_networkx_edges() και draw_networkx_labels() της βιβλιοθήκης NetworkX, καθώς και με τις συναρτήσεις axis() και show() της βιβλιοθήκης Matplotlib, υλοποιεί το γράφημα.

```
# Draw the graph
pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, node_size=500)
nx.draw_networkx_edges(G, pos)
nx.draw_networkx_labels(G, pos, font_size=10, font_family="sans-serif")
plt.axis("off")
plt.show()
```

Η συνάρτηση print_all_distances() υπολογίζει τις αποστάσεις κάθε κόμβου από όλους τους άλλους κόμβους του γραφήματος χρησιμοποιώντας τη μέθοδο shortest_path_length() της βιβλιοθήκης NetworkX. Στη συνέχεια εκτυπώνει τις

αποστάσεις με τη μορφή "source_node: distance_1, distance_2, ..., distance_N". Η συνάρτηση επιστρέφει επίσης ένα λεξικό που περιέχει το άθροισμα των αποστάσεων για κάθε κόμβο.

```
def print_all_distances(graph):
    """
    Calculates the distances from every node to all other nodes in the graph,
    and prints them in the format "source_node: distance_1, distance_2, ..., distance_N".
    Also returns a dictionary containing the sum of the distances for each node.
    """
    sums = {}
    for source in graph.nodes():
        distances = nx.shortest_path_length(graph, source=source)
        dist_list = [str(distances[target]) for target in distances if target != source]
        dist_str = ", ".join(dist_list)
        print(f"Distances from {source}: {dist_str}")
        sums[source] = sum(distances[target] for target in distances if target != source)
    return sums

sums = print_all_distances(G)
print(f"Sums of distances: {sums}")
```

Η συνάρτηση eccentricity() υπολογίζει την εκκεντρότητα κάθε κόμβου στο γράφημα χρησιμοποιώντας τη μέθοδο shortest_path_length() της βιβλιοθήκης NetworkX. Η εκκεντρότητα ενός κόμβου ορίζεται ως η μέγιστη απόσταση του κόμβου από οποιονδήποτε άλλο κόμβο του γράφου.

```
def eccentricity(graph):
    """
    Calculates the eccentricity of each node in the graph, defined as the maximum distance
    from the node to any other node in the graph.
    """
    eccentricities = {}
    for node in graph.nodes():
        distances = nx.shortest_path_length(graph, source=node)
        max_distance = max(distances.values())
        eccentricities[node] = max_distance
    return eccentricities

ecc = eccentricity(G)
print(f"Eccentricities: {ecc}")
```

Η διάμεσος του γραφήματος υπολογίζεται ως ο κόμβος με το ελάχιστο άθροισμα αποστάσεων χρησιμοποιώντας τη συνάρτηση min() στο λεξικό sums. Το κέντρο του γραφήματος υπολογίζεται ως ο κόμβος με την ελάχιστη εκκεντρότητα χρησιμοποιώντας τη συνάρτηση min() στο λεξικό eccentricities.

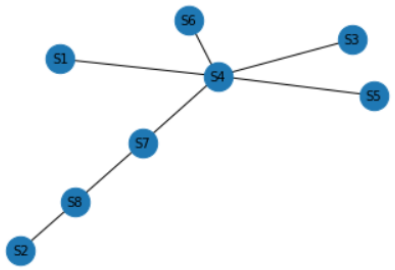
```
# Calculate the median of the graph (minimum sum of distances)
median = min(sums, key=sums.get)

# Calculate the center of the graph (minimum eccentricity)
center = min(ecc, key=ecc.get)
```

Τέλος, ο κεντρικός κόμβος και η διάμεσος εκτυπώνονται ως τα σημεία στα οποία ο στρατός θα επιτεθεί αρχικά για να χτυπήσει την ομάδα κρούσης της αυτοκρατορίας Β και έπειτα για να καταστρέψει την αποθήκη ανεφοδιασμού, αντίστοιχα.

```
# Print the results
print(f"Median of the graph is node: {median}")
print(f"Center of the graph is node: {center}")
print(f"The point where our army will initially attack in order to strike first the strike group of the B empire is: {center}",
      "\n"f"and the point at which our army will attack in order to destroy the supply depot is: {median}")
```

Το τελικό αποτέλεσμα που θα πάρουμε από τον αλγόριθμο είναι το παρακάτω. Σε αυτό διακρίνουμε το γράφημα με τα σημεία-κόμβους του, στη συνέχεια τις αποστάσεις των κόμβων προς τους υπόλοιπους κόμβους (δε μας δίνονται βάρη ακμών οπότε θεωρούμε πως το βάρος κάθε ακμής είναι ίσο με 1), έπειτα δύο λεξικά sums που περιέχει τα αθροίσματα των αποστάσεων για κάθε κόμβο του γραφήματος και eccentricities που περιέχει την κεντρικότητα για κάθε κόμβο. Τέλος, εκτυπώνεται ο μέσος και το κέντρο του γραφήματος, καθώς και η τελική ετυμηγορία μας για τις τοποθεσίες στις οποίες πρέπει να επιτεθεί ο στρατός της αυτοκρατορίας μας.




```
Distances from S1: 1, 2, 2, 2, 2, 3, 4
Distances from S2: 1, 2, 3, 4, 4, 4, 4
Distances from S3: 1, 2, 2, 2, 2, 3, 4
Distances from S4: 1, 1, 1, 1, 1, 2, 3
Distances from S5: 1, 2, 2, 2, 2, 3, 4
Distances from S6: 1, 2, 2, 2, 2, 3, 4
Distances from S7: 1, 1, 2, 2, 2, 2, 2
Distances from S8: 1, 1, 2, 3, 3, 3, 3
Sums of distances: {'S1': 16, 'S2': 22, 'S3': 16, 'S4': 10, 'S5': 16, 'S6': 16, 'S7': 12, 'S8': 16}
Eccentricities: {'S1': 4, 'S2': 4, 'S3': 4, 'S4': 3, 'S5': 4, 'S6': 4, 'S7': 2, 'S8': 3}
Median of the graph is node: S4
Center of the graph is node: S7
The point where our army will initially attack in order to strike first the strike group of the B empire is: S7
and the point at which our army will attack in order to destroy the supply depot is: S4
```