ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Πανεπιστήμιο Πειραιώς

Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών «Πληροφορικής»

ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ ΑΛΓΟΡΙΘΜΟΙ

Ονοματεπώνυμο : Αντώνιος Τζιβάκης

Αριθμός Μητρώου : ΜΠΠΛ2244

Ονοματεπώνυμο : Μαρία Αμοργιανού

Αριθμός Μητρώου : ΜΠΠΛ2205

Αναλυτικές οδηγίες εκτέλεσης του κώδικα:

Ο κώδικας που γράψαμε σε python χρησιμοποιεί τη βιβλιοθήκη NetworkX για τη δημιουργία τυχαίων γραφημάτων και τον υπολογισμό δέντρων Steiner.

Πριν την εκτέλεση του κώδικα θα πρέπει να έχετε εγκαταστήσει τις απαραίτητες βιβλιοθήκες. Αυτό μπορεί να γίνει με την παρακάτω εντολή:

-pip install networkx matplotlib

Κατά τη διάρκεια της εκτέλεσης, θα δείτε τα τυχαία γραφήματα με τους κόμβους και τις ακμές τους, καθώς και τα αντίστοιχα δέντρα Steiner. Το βάρος του δέντρου Steiner θα εμφανίζεται επίσης.

Επεξήγηση του κώδικα:

```
def main():
    for num_nodes in range(6, 13):
        steiner_percentage = 0.7
        max_edge_weight = 50

        G = generate_random_graph(num_nodes, steiner_percentage,
max_edge_weight)
        nx.draw(G, with_labels=True, font_weight='bold',
node_size=500, node_color='lightcoral', font_size=12,
                edge_color='black', width=2)
        plt.show()
        print("Graph G:", G.edges(data=True))

        steiner_tree = find_steiner_tree(G)

        nx.draw(steiner_tree, with_labels=True, font_weight='bold',
node_size=500, node_color='lightcoral', font_size=12,
                edge_color='black', width=2)
        plt.show()

        steiner_tree_weight = sum([steiner_tree[u][v]['weight'] for
u, v in steiner_tree.edges()])

        print(f"Graph with {num_nodes} nodes:")
        print(f"Steiner Tree Weight: {steiner_tree_weight}")
        print("\n")


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

Αρχικά στη ***main*** που έχουμε κατασκευάσει θέτουμε επαναληπτική λούπα για να φτιάξουμε γραφήματα από 6 έως 12 κόμβους. Θέτουμε τη μεταβλητή ***steiner_percentage*** = 0.7 για να πετύχουμε το ζητούμενο ποσοστό των κόμβων να θέτονται ως ‘steiner’ κόμβοι και ορίζουμε μέγιστο βάρος των ακμών το 50.

Στη συνέχεια στη μεταβλητή G εκχωρούμε τη μέθοδο ***generate_random_graph.***

Αυτή η μέθοδος έχει ως εισόδους τον αριθμό των κόμβων, το ποσοστό steiner και το μέγιστο βάρος ακμών κι επιστρέφει ένα τυχαίο γράφημα G.

Αφού εμφανίσουμε το γράφημα G με τις μεθόδους ***nx.draw()*** για το σχεδιασμό και ***plt.show()*** για την εμφάνιση στη μεταβλητή ***steiner_tree*** εκχωρούμε τη μέθοδο ***find_steiner_tree*** και δίνουμε ως είσοδο το γράφημα G.

Αυτή η μέθοδος θα μας επιστρέψει το δέντρο Steiner TH, όπου στη συνέχεια το εμφανίζουμε μαζί με πληροφορίες για το βάρος του που προκύπτει από το άθροισμα

των βαρών όλων των ακμών του, όπως και από γράφημα πόσων κόμβων προήλθε. Ακολουθεί η επεξήγηση των επιμέρους μεθόδων που οδηγούν στα παραπάνω αποτελέσματα.

```
def generate_random_graph(num_nodes, steiner_percentage,
max_edge_weight):
    G = nx.Graph()
    G.add_nodes_from(range(1, num_nodes + 1))

    # Create Steiner nodes
    num_steiner_nodes = int(steiner_percentage * num_nodes)
    steiner_nodes = random.sample(list(G.nodes()), num_steiner_nodes)
    print("Steiner nodes selected:", steiner_nodes)

    # Attach Steiner nodes to the graph
    for node in G.nodes():
        if node in steiner_nodes:
            G.nodes[node]['steiner'] = True
        else:
            G.nodes[node]['steiner'] = False

    # Create random edges
    num_edges = random.randint(num_nodes - 1, num_nodes * (num_nodes
- 1) // 2)

    for u in G.nodes():
        while G.degree(u) < 1:
            v = random.choice(list(G.nodes()))
            if v != u and G.degree(v) < 2:
                weight = random.randint(1, max_edge_weight)
                G.add_edge(u, v, weight=weight)

    while G.number_of_edges() < num_nodes - 1:
        is_connected = nx.is_connected(G)
        if not is_connected:
            u = random.choice(list(G.nodes()))
            v = random.choice(list(G.nodes()))
            if u != v and not nx.has_path(G, u, v):
                weight = random.randint(1, max_edge_weight)
                G.add_edge(u, v, weight=weight)

    # Connect nodes until reaching the desired number of edges
    while G.number_of_edges() < num_edges:
        u = random.choice(list(G.nodes()))
        v = random.choice(list(G.nodes()))
        if u != v and not G.has_edge(u, v):
            weight = random.randint(1, max_edge_weight)
            G.add_edge(u, v, weight=weight)

    return G
```

Η παραπάνω μέθοδος χρησιμοποιείται για την κατασκευή μη κατευθυνόμενου γραφήματος G. Αρχικά προσθέτουμε τους κόμβους στο νέο γράφημα που δημιουργούμε και ορίζουμε πόσοι από αυτούς θα είναι κόμβοι Steiner με τη

μεταβλητή ***num_steiner_nodes***. Χρησιμοποιούμε επιπλέον αυτή τη μεταβλητή για να θέσουμε τυχαία ποιοι θα είναι οι κόμβοι Steiner στο γράφημα.

Στη συνέχεια ακολουθούν επαναλήψεις για να τεθούν οι ακμές τυχαία επιλέγοντας ανάμεσα στο μικρότερο αριθμό ακμών που μπορούμε να έχουμε για είναι το γράφημα συνεχές (Αριθμός κόμβων -1) και το μέγιστο αριθμό ακμών που μπορούμε να έχουμε (Αρ. κόμβων * (Αρ. κόμβων -1)//2). Στην κάθε ακμή που προστίθεται θέτουμε κι ένα τυχαίο βάρος από 1 μέχρι το μέγιστο 50.

```
def find_steiner_tree(G):
    G1 = build_complete_graph(G)
    print("Complete Graph from original G (G1):",
G1.edges(data=True))

    T1 = nx.minimum_spanning_tree(G1)
    print("Minimum Spanning Tree (T1):", T1.edges(data=True))

    Gs = build_subgraph(G, T1)
    print("Constructed Subgraph (Gs):", Gs.edges(data=True))

    Ts = nx.minimum_spanning_tree(Gs)
    print("Minimum Spanning Tree (Ts):", Ts.edges(data=True))

    TH = build_steiner_tree(G, Ts)
    print("Steiner Tree (TH):", TH.edges(data=True))
    return TH
```

Η μέθοδος ***find_steiner_tree*** που δέχεται ως είσοδο το γράφημα G εκτελεί τις παρακάτω υπομεθόδους για να επιστρέψει ως αποτέλεσμα το δέντρο Steiner:

- ***build_complete_graph(G)*** -Δημιουργεί έναν πλήρη γράφο G1 που περιέχει μόνο τους Steiner κόμβους.
- ***nx.minimum_spanning_tree(G1)*** -Βρίσκει το ελάχιστο γεννητικό δέντρο T1 του G1.
- ***build_subgraph(G,T1)*** -Κατασκευάζει ένα υπογράφημα Gs για το T1 προσθέτοντας τα βάρη των ελαχίστων μονοπατιών στο αρχικό γράφημα G.
- ***nx.minimum_spanning_tree(Gs)*** -Βρίσκει το ελάχιστο γεννητικό δέντρο Ts του Gs.

- ***build_steiner_tree(G, Ts)*** -Από το ελάχιστο γεννητικό δέντρο Ts κατασκευάζει το δέντρο Steiner TH.

```
def build_complete_graph(G):
    G1 = nx.Graph()

    steiner_nodes = [node for node in G.nodes() if
G.nodes[node].get('steiner', False)]
    G1.add_nodes_from(steiner_nodes)

    for u in steiner_nodes:
        for v in steiner_nodes:
            if u != v:
                weight = nx.shortest_path_length(G, u, v,
weight='weight')
                G1.add_edge(u, v, weight=weight)

    return G1
```

Για τη δημιουργία του πλήρους γραφήματος G1 που προέρχεται από το G δημιουργούμε ένα νέο γράφημα και του προσθέτουμε μόνο τους κόμβους που έχουμε ορίσει ως Steiner. Στη συνέχεια συνδέουμε όλους του κόμβους μεταξύ τους και ορίζουμε ως βάρος στην εκάστοτε ακμή το ελάχιστο βάρος που προέκυπτε στο γράφημα G για να κινηθούμε από τον ένα κόμβο Steiner στον άλλο.

```
def build_subgraph(G, T1):
    Gs = nx.Graph()

    for u, v in T1.edges():
        path = nx.shortest_path(G, u, v, weight='weight')

        for node in path:
            Gs.add_node(node)
```

```
        for i in range(len(path) - 1):
            Gs.add_edge(path[i], path[i + 1], weight=G[path[i]]
[path[i + 1]]['weight'])

    return Gs
```

Για το υπογράφημα Gs δημιουργούμε ένα καινούριο γράφο. Ελέγχουμε κάθε ακμή του ελάχιστου δέντρου T1 και υπολογίζουμε το ελάχιστο μονοπάτι από τον κόμβο u στον κόμβο v στο αρχικό γράφημα G, λαμβάνοντας υπόψιν και τα βάρη τους και προσθέτουμε και τους κόμβους και τις ακμές στο Gs.

```
def build_steiner_tree(G, Ts):
    TH = Ts.copy()

    while True:
        leaves = [node for node in TH.nodes() if TH.degree(node) == 1
and not G.nodes[node].get('steiner', False)]

        if not leaves:
            break

        for leaf in leaves:
            edges_to_remove = list(TH.edges(leaf))
            TH.remove_edges_from(edges_to_remove)
            TH.remove_node(leaf)

    return TH
```

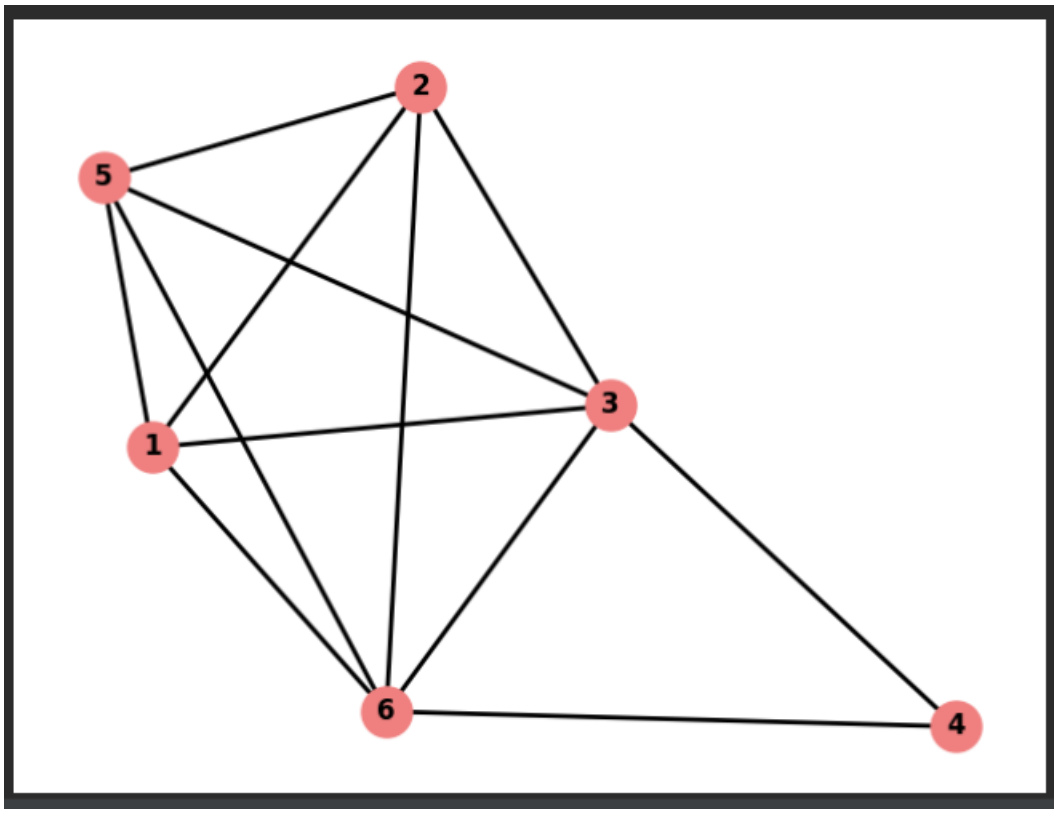
Με τη παραπάνω μέθοδο δημιουργούμε μια αντιγραφή του ελάχιστου γεννητικού δέντρου Ts και την αποθηκεύουμε στη μεταβλητή TH.

Στη συνέχεια η συνάρτηση βρίσκεται σε μια συνεχή λούπα που διαρκεί όσο το γράφημα περιέχει κόμβους που έχουν μία ακμή και δεν είναι κόμβοι Steiner.

Βρίσκουμε τις ακμές αυτών των κόμβων και αφαιρούμε αυτές και τον κόμβο, ώστε να προκύψει το ελάχιστο δέντρο που συνδέει τους κόμβους Steiner.

Επαλήθευση αποτελεσματικότητας του κώδικα:

Για να αποδείξουμε την αποτελεσματικότητα του αλγορίθμου θα τον τρέξουμε και θα υπολογίσουμε τη βέλτιστη λύση του κάθε γραφήματος στο χέρι και θα συγκρίνουμε αυτή τη λύση με τη λύση που εμφανίζει ο αλγόριθμος.



Δημιουργούμε το πρώτο τυχαίο γράφημα 6 κόμβων και ακμών με τυχαία βάρη και έχουμε ως πληροφορίες από τον κώδικα:

Steiner nodes selected: [2, 5, 1, 6]

Graph G: [(1, 6, {'weight': 28}), (1, 5, {'weight': 4}), (1, 2, {'weight': 42}), (1, 3, {'weight': 34}), (2, 5, {'weight': 50}), (2, 6, {'weight': 2}), (2, 3, {'weight': 47}), (3, 5, {'weight': 35}), (3, 4, {'weight': 47}), (3, 6, {'weight': 26}), (4, 6, {'weight': 23}), (5, 6, {'weight': 19})]

Complete Graph from original G (G1): [(1, 2, {'weight': 25}), (1, 5, {'weight': 4}), (1, 6, {'weight': 23}), (2, 5, {'weight': 21}), (2, 6, {'weight': 2}), (5, 6, {'weight': 19})]

Minimum Spanning Tree (T1): [(1, 5, {'weight': 4}), (2, 6, {'weight': 2}), (5, 6, {'weight': 19})]

Constructed Subgraph (Gs): [(1, 5, {'weight': 4}), (5, 6, {'weight': 19}), (2, 6, {'weight': 2})]

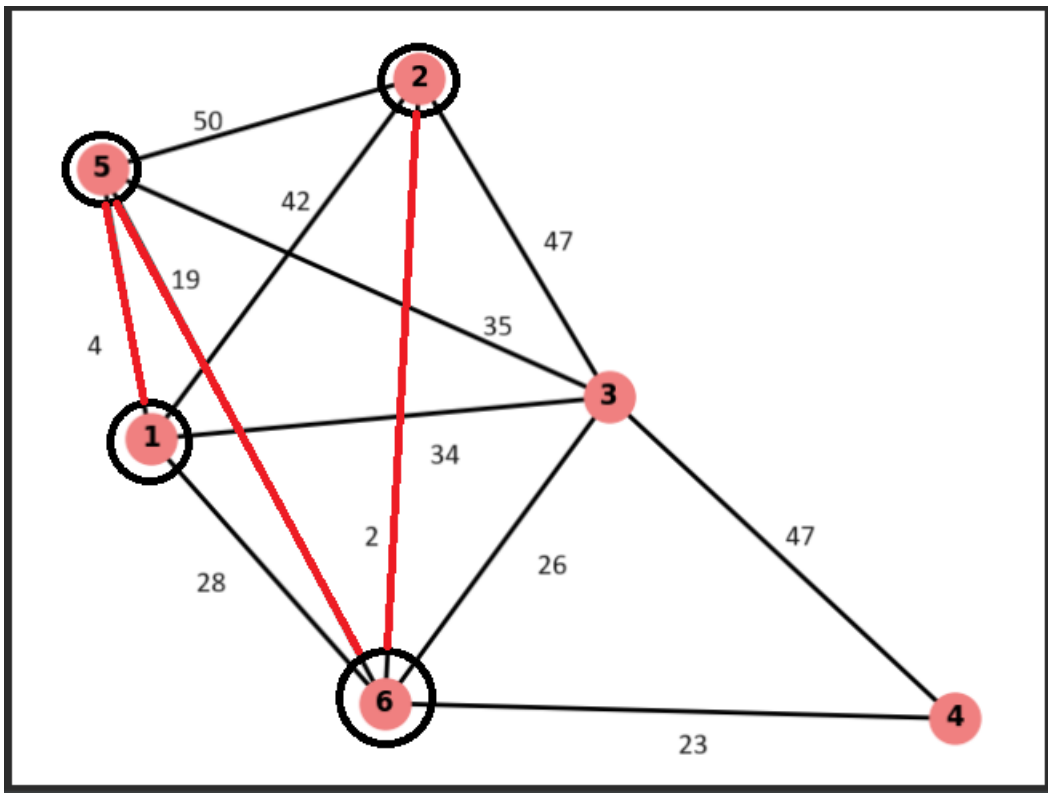
Minimum Spanning Tree (Ts): [(1, 5, {'weight': 4}), (5, 6, {'weight': 19}), (2, 6, {'weight': 2})]

Steiner Tree (TH): [(1, 5, {'weight': 4}), (5, 6, {'weight': 19}), (2, 6, {'weight': 2})]

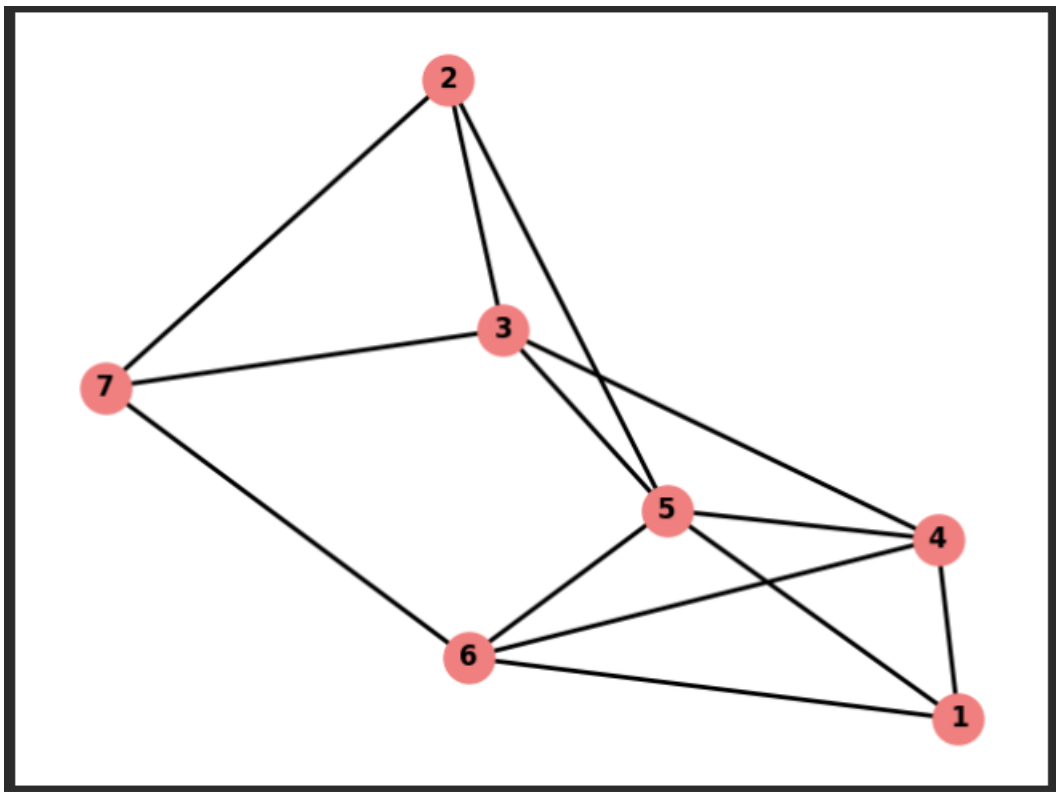
Graph with 6 nodes:

Steiner Tree Weight: 25

Από αυτό η βέλτιστη λύση που προκύπτει είναι:



Η μέγιστη λύση έχει βάρος 2+19+4 =25. Παρατηρούμε ότι είναι ακριβώς και η λύση που δίνει ο αλγόριθμος.



Για το τυχαίο γράφημα 7 κόμβων έχουμε τις παρακάτω πληροφορίες

Steiner nodes selected: [5, 4, 2, 6]

Graph G: [(1, 5, {'weight': 27}), (1, 6, {'weight': 14}), (1, 4, {'weight': 45}), (2, 7, {'weight': 4}), (2, 5, {'weight': 7}), (2, 3, {'weight': 20}), (3, 7, {'weight': 40}), (3, 4, {'weight': 20}), (3, 5, {'weight': 43}), (4, 6, {'weight': 18}), (4, 5, {'weight': 44}), (5, 6, {'weight': 50}), (6, 7, {'weight': 16})]

Complete Graph from original G (G1): [(2, 4, {'weight': 38}), (2, 5, {'weight': 7}), (2, 6, {'weight': 20}), (4, 5, {'weight': 44}), (4, 6, {'weight': 18}), (5, 6, {'weight': 27})]

Minimum Spanning Tree (T1): [(2, 5, {'weight': 7}), (2, 6, {'weight': 20}), (4, 6, {'weight': 18})]

Constructed Subgraph (Gs): [(2, 5, {'weight': 7}), (2, 7, {'weight': 4}), (7, 6, {'weight': 16}), (6, 4, {'weight': 18})]

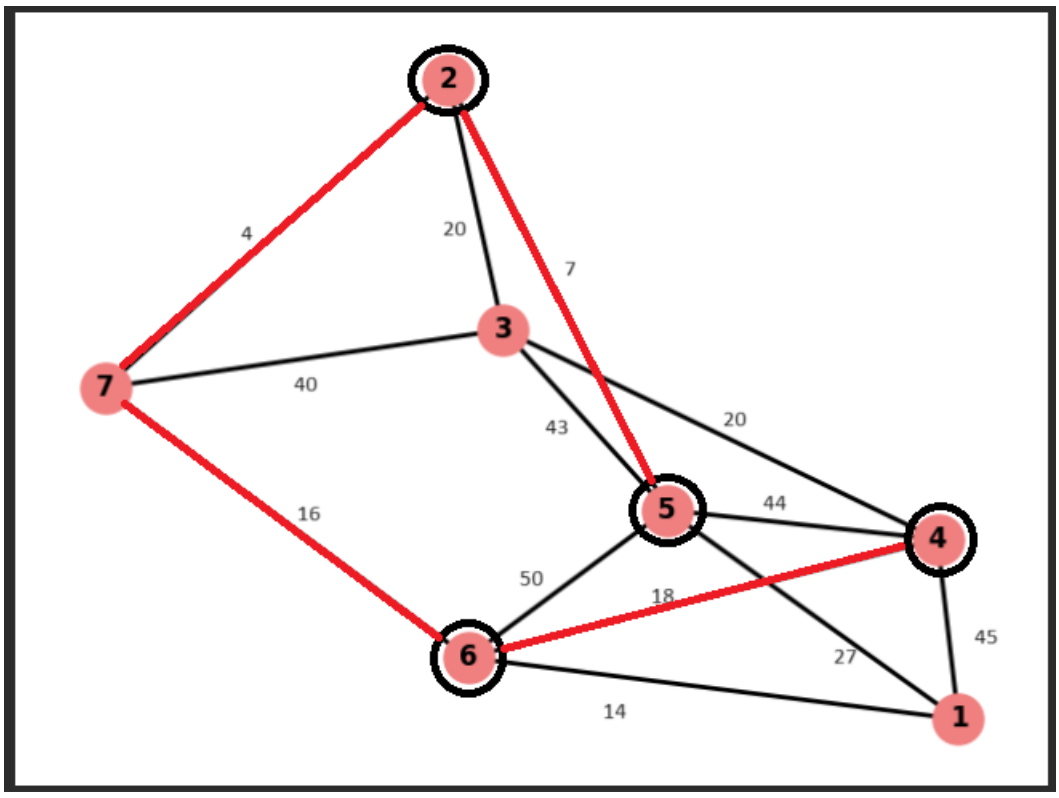
Minimum Spanning Tree (Ts): [(2, 7, {'weight': 4}), (2, 5, {'weight': 7}), (7, 6, {'weight': 16}), (6, 4, {'weight': 18})]

Steiner Tree (TH): [(2, 7, {'weight': 4}), (2, 5, {'weight': 7}), (7, 6, {'weight': 16}), (6, 4, {'weight': 18})]

Graph with 7 nodes:

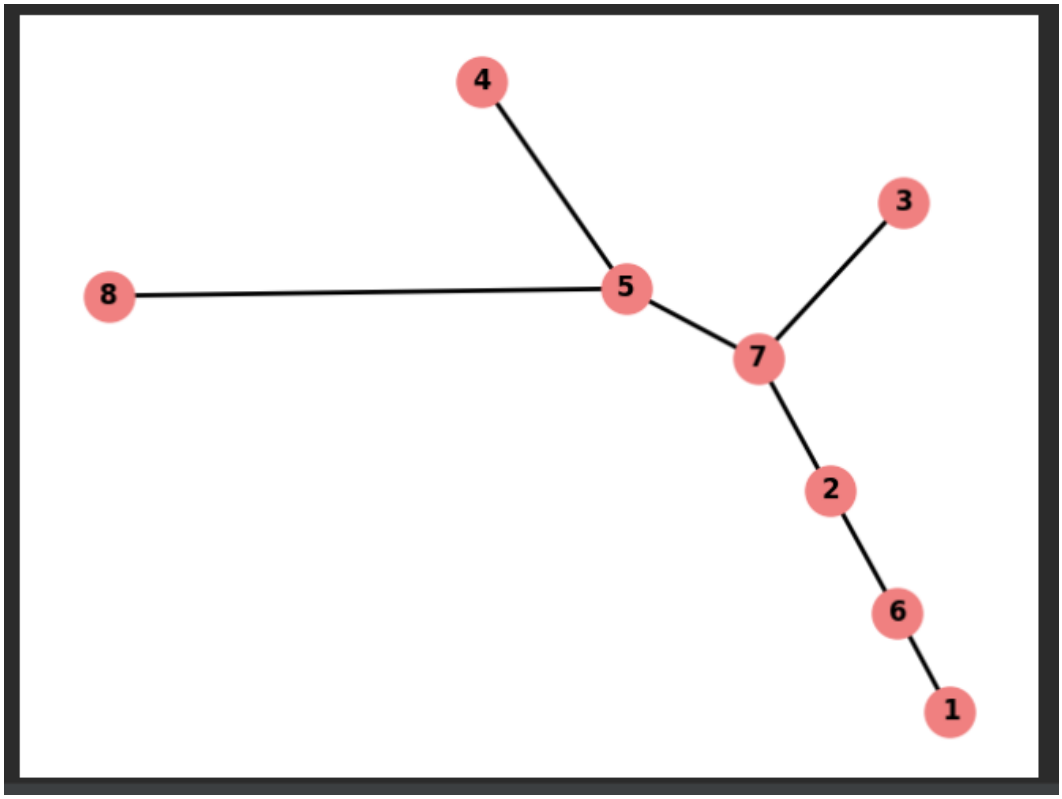
Steiner Tree Weight: 45

Η βέλτιστη λύση που βρίσκουμε είναι:



Με βάρος 7+4+16+18 = 45

Παρατηρούμε ότι ο λόγος της λύσης του αλγορίθμου προς τη βέλτιστη εξακολουθεί να είναι 1.



Για το γράφημα 8 κόμβων έχουμε αντίστοιχα:

Steiner nodes selected: [1, 7, 4, 8, 6]

Graph G: [(1, 6, {'weight': 41}), (2, 7, {'weight': 36}), (2, 6, {'weight': 40}), (3, 7, {'weight': 9}), (4, 5, {'weight': 5}), (5, 8, {'weight': 1}), (5, 7, {'weight': 37})]

Complete Graph from original G (G1): [(1, 4, {'weight': 159}), (1, 6, {'weight': 41}), (1, 7, {'weight': 117}), (1, 8, {'weight': 155}), (4, 6, {'weight': 118}), (4, 7, {'weight': 42}), (4, 8, {'weight': 6}), (6, 7, {'weight': 76}), (6, 8, {'weight': 114}), (7, 8, {'weight': 38})]

Minimum Spanning Tree (T1): [(1, 6, {'weight': 41}), (4, 8, {'weight': 6}), (6, 7, {'weight': 76}), (7, 8, {'weight': 38})]

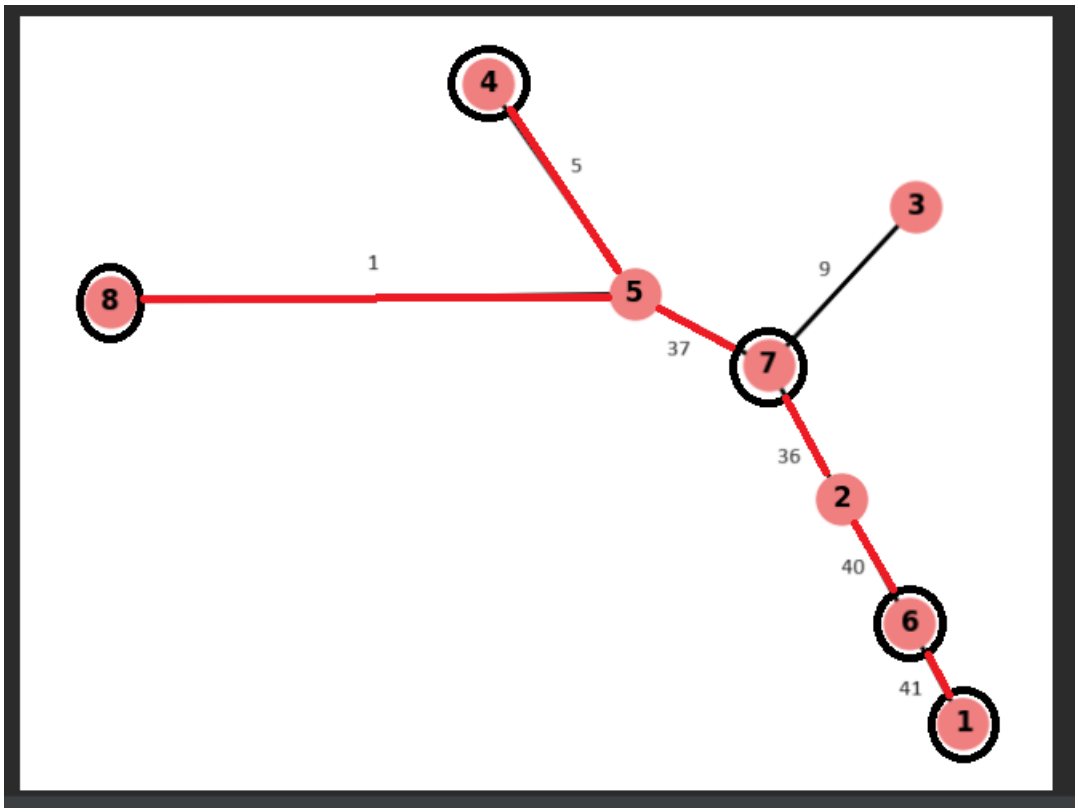
Constructed Subgraph (Gs): [(1, 6, {'weight': 41}), (6, 2, {'weight': 40}), (4, 5, {'weight': 5}), (5, 8, {'weight': 1}), (5, 7, {'weight': 37}), (2, 7, {'weight': 36})]

Minimum Spanning Tree (Ts): [(1, 6, {'weight': 41}), (6, 2, {'weight': 40}), (4, 5, {'weight': 5}), (5, 8, {'weight': 1}), (5, 7, {'weight': 37}), (2, 7, {'weight': 36})]

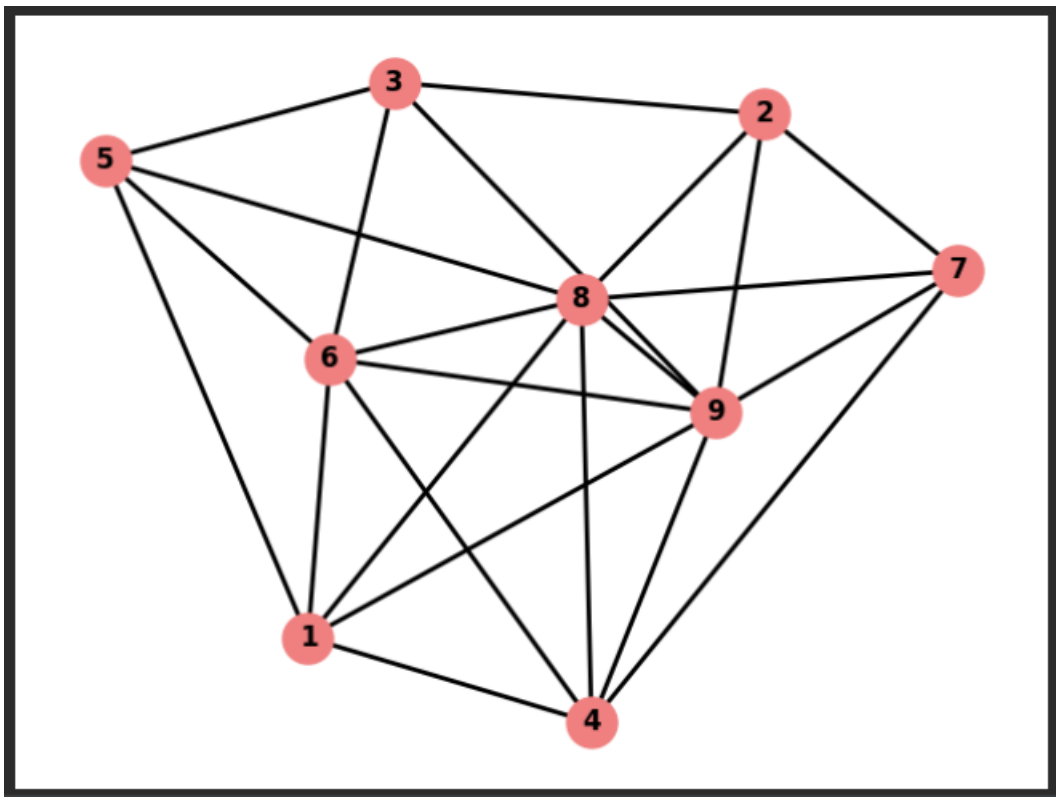
Steiner Tree (TH): [(1, 6, {'weight': 41}), (6, 2, {'weight': 40}), (4, 5, {'weight': 5}), (5, 8, {'weight': 1}), (5, 7, {'weight': 37}), (2, 7, {'weight': 36})]

Graph with 8 nodes:

Steiner Tree Weight: 160



Η βέλτιστη λύση έχει βάρος 1+5+37+36+40+41 =160 και για άλλη μια φορά ο ταυτίζεται με το αποτέλεσμα του αλγορίθμου.



Για το γράφημα 9 κόμβων:

Steiner nodes selected: [2, 6, 5, 7, 4, 3]

Graph G: [(1, 5, {'weight': 7}), (1, 4, {'weight': 30}), (1, 8, {'weight': 14}), (1, 9, {'weight': 14}), (1, 6, {'weight': 23}), (2, 7, {'weight': 48}), (2, 8, {'weight': 38}), (2, 9, {'weight': 38}), (2, 3, {'weight': 43}), (3, 9, {'weight': 14}), (3, 5, {'weight': 41}), (3, 6, {'weight': 50}), (4, 8, {'weight': 10}), (4, 9, {'weight': 29}), (4, 6, {'weight': 14}), (4, 7, {'weight': 1}), (5, 6, {'weight': 22}), (5, 8, {'weight': 21}), (6, 9, {'weight': 25}), (6, 8, {'weight': 41}), (7, 9, {'weight': 40}), (7, 8, {'weight': 43}), (8, 9, {'weight': 39})]

Complete Graph from original G (G1): [(2, 3, {'weight': 43}), (2, 4, {'weight': 48}), (2, 5, {'weight': 59}), (2, 6, {'weight': 62}), (2, 7, {'weight': 48}), (3, 4, {'weight': 43}), (3, 5, {'weight': 35}), (3, 6, {'weight': 39}), (3, 7, {'weight': 44}), (4, 5, {'weight': 31}), (4, 6, {'weight': 14}), (4, 7, {'weight': 1}), (5, 6, {'weight': 22}), (5, 7, {'weight': 32}), (6, 7, {'weight': 15})]

Minimum Spanning Tree (T1): [(2, 3, {'weight': 43}), (3, 5, {'weight': 35}), (4, 7, {'weight': 1}), (4, 6, {'weight': 14}), (5, 6, {'weight': 22})]

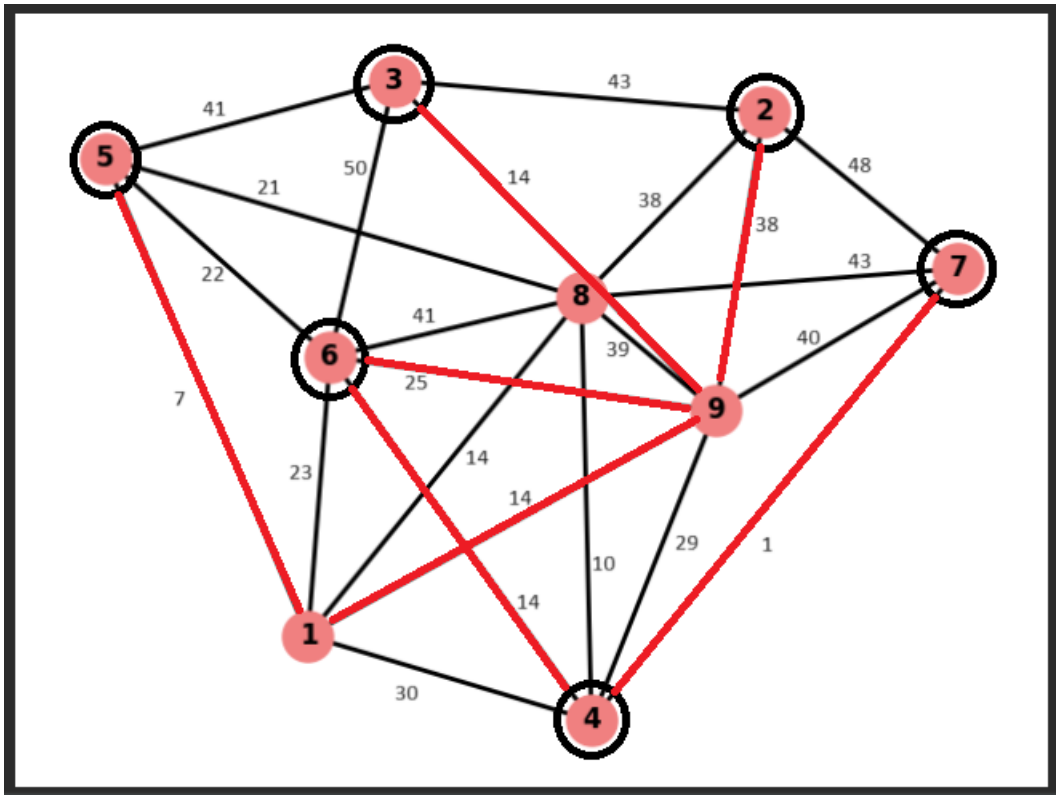
Constructed Subgraph (Gs): [(2, 3, {'weight': 43}), (3, 9, {'weight': 14}), (9, 1, {'weight': 14}), (1, 5, {'weight': 7}), (5, 6, {'weight': 22}), (4, 7, {'weight': 1}), (4, 6, {'weight': 14})]

Minimum Spanning Tree (Ts): [(2, 3, {'weight': 43}), (3, 9, {'weight': 14}), (9, 1, {'weight': 14}), (1, 5, {'weight': 7}), (5, 6, {'weight': 22}), (4, 7, {'weight': 1}), (4, 6, {'weight': 14})]

Steiner Tree (TH): [(2, 3, {'weight': 43}), (3, 9, {'weight': 14}), (9, 1, {'weight': 14}), (1, 5, {'weight': 7}), (5, 6, {'weight': 22}), (4, 7, {'weight': 1}), (4, 6, {'weight': 14})]

Graph with 9 nodes:

Steiner Tree Weight: 115



Από τη βέλτιστη λύση προκύπτει:

Βάρος = 38+14+25+14+1+7 = 113

Εδώ βλέπουμε ότι ο λόγος του αλγορίθμου προς τη βέλτιστη λύση είναι 115/113 =1.02

Παρατηρείται ότι όσο θα αυξάνονται οι αριθμοί των κόμβων το αποτέλεσμα που θα εμφανίζει ο αλγόριθμος δε θα είναι πάντα το βέλτιστο αλλά θα είναι κοντά σε αυτό με μέγιστο λόγο τη διπλάσια τιμή.