ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΜΣ "ΠΡΟΗΓΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΚΗΣ -
ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ"

Σημειώσεις του μαθήματος

ΕΙΔΙΚΑ ΘΕΜΑΤΑ ΘΕΩΡΙΑΣ ΚΑΙ ΕΦΑΡΜΟΓΩΝ ΓΡΑΦΗΜΑΤΩΝ

Κ. Μανές - Ι. Τασούλας

Σημειώσεις διαλέξεων 3

ΠΕΙΡΑΙΑΣ 2023

Οι παρούσες σημειώσεις βασίζονται σε προηγούμενες σημειώσεις του μαθήματος που έχουν συγγράψει ο Καθηγητής κ. Αριστείδης Σαπουνάκης και ο Καθηγητής κ. Παναγιώτης-Γεώργιος Τσικούρας.

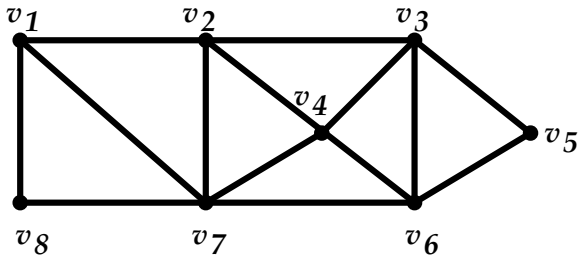
5. ΑΠΟΣΤΑΣΕΙΣ

Απόσταση (distance) $d(u,v)$ μεταξύ δύο κόμβων u,v μιας συνιστώσας του G ονομάζεται το ελάχιστο μήκος μεταξύ όλων των διαδρομών που τους συνδέουν.

Μερικοί συγγραφείς γενικεύουν τον παραπάνω ορισμό, ορίζοντας ως απόσταση δύο κόμβων που ανήκουν σε διαφορετικές συνιστώσες ενός μη συνεκτικού γραφήματος το ∞.

Γεωδαιτικό ή **συντομότερο** λέγεται κάθε $u-v$ μονοπάτι ενός γραφήματος G, με μήκος ίσο με $d(u,v)$.

Παράδειγμα:



$d(v_1,v_4)=2$.

(v_1,v_2,v_4): γεωδαιτικό, (v_1,v_7,v_4): γεωδαιτικό.

(v_1,v_8,v_7,v_4): όχι γεωδαιτικό.

Η εύρεση της απόστασης ανάμεσα σε δύο κόμβους u,v ενός γραφήματος μπορεί να γίνει χρησιμοποιώντας την αναζήτηση σε πλάτος, ή τους αλγορίθμους του Dijkstra, ή των Bellman - Ford. (Οι δύο τελευταίοι αλγόριθμοι μπορούν να δώσουν απαντήσεις και σε γραφήματα που έχουν βάρη πάνω στους δεσμούς τους.)

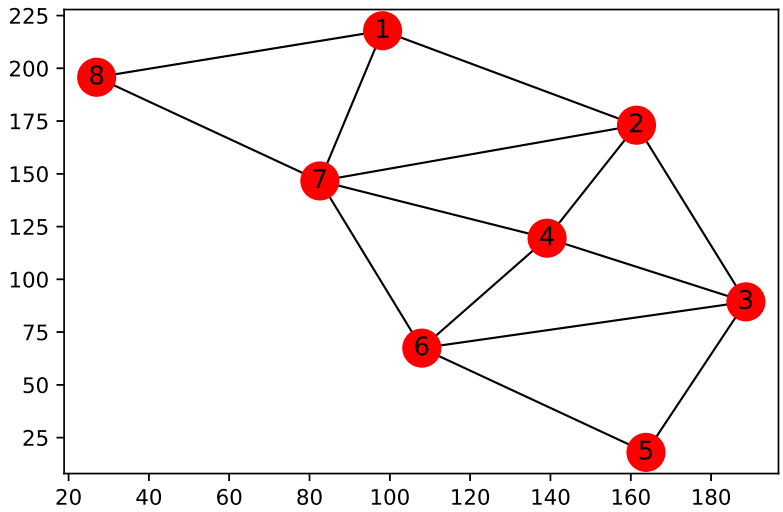
Η βιβλιοθήκη `networkx` έχει τις μεθόδους `shortest_path(G,v,u)` και `shortest_path_length(G,v,u)` που υπολογίζουν ένα γεωδαιτικό μονοπάτι μεταξύ των κορυφών v και u και την απόσταση των v και u αντίστοιχα.

Επιπρόσθετα, υπάρχουν οι μέθοδοι `all_pairs_shortest_path(G)` και `all_pairs_shortest_path_length(G)` που υπολογίζουν ένα γεωδαιτικό μονοπάτι ανάμεσα σε όλα τα ζεύγη κορυφών και τα αντίστοιχα μήκη τους.

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

G = nx.Graph()
V = [1,2,3,4,5,6,7,8]
E = [[1,2],[1,7],[1,8],[2,3],[2,4],[2,7],[3,4],[3,5],[3,6],
     [4,6],[4,7],[5,6],[6,7],[7,8]]

G.add_nodes_from(V)
G.add_edges_from(E)
pos = nx.nx_agraph.graphviz_layout(G)
nx.draw_networkx(G,pos)

v, u = 1, 4
print("The distance between nodes",v,"and",u,"is:",nx.shortest_path_length(G,v,u))

shortestpaths = dict(nx.all_pairs_shortest_path(G))
for v in G:
```

```
    print("A shortest path between",v)
    for u in G:
        print("and",u,"is:",shortestpaths[v][u])

alldistances = dict(nx.all_pairs_shortest_path_length(G))
n = G.order()
D = np.zeros((n,n)) #create a n x n matrix with zero entries
for v in G:
    for u in G:
        #D has numbering 0...7 while nodes are numbered 1...8
        D[u-1][v-1] = alldistances[v][u]
print("The distances between all nodes pairs of G are:")
print(D)

plt.show()
```

Output:




```
The distance between nodes
   1 and 4 is: 2
A shortest path between 1
and 1 is: [1]
and 2 is: [1, 2]
and 3 is: [1, 2, 3]
and 4 is: [1, 2, 4]
and 5 is: [1, 2, 3, 5]
and 6 is: [1, 7, 6]
and 7 is: [1, 7]
and 8 is: [1, 8]
A shortest path between 2
and 1 is: [2, 1]
and 2 is: [2]
and 3 is: [2, 3]
and 4 is: [2, 4]
and 5 is: [2, 3, 5]
and 6 is: [2, 3, 6]
and 7 is: [2, 7]
and 8 is: [2, 1, 8]
A shortest path between 3
and 1 is: [3, 2, 1]
and 2 is: [3, 2]
and 3 is: [3]
and 4 is: [3, 4]
and 5 is: [3, 5]
and 6 is: [3, 6]
and 7 is: [3, 2, 7]
and 8 is: [3, 2, 1, 8]
A shortest path between 4
and 1 is: [4, 2, 1]
and 2 is: [4, 2]
and 3 is: [4, 3]
and 4 is: [4]
and 5 is: [4, 3, 5]
and 6 is: [4, 6]
and 7 is: [4, 7]
and 8 is: [4, 7, 8]
A shortest path between 5
and 1 is: [5, 3, 2, 1]
and 2 is: [5, 3, 2]
and 3 is: [5, 3]
and 4 is: [5, 3, 4]
and 5 is: [5]
and 6 is: [5, 6]
and 7 is: [5, 6, 7]
and 8 is: [5, 6, 7, 8]
A shortest path between 6
and 1 is: [6, 7, 1]
and 2 is: [6, 3, 2]
and 3 is: [6, 3]
and 4 is: [6, 4]
and 5 is: [6, 5]
and 6 is: [6]
and 7 is: [6, 7]
and 8 is: [6, 7, 8]
A shortest path between 7
and 1 is: [7, 1]
and 2 is: [7, 2]
and 3 is: [7, 2, 3]
and 4 is: [7, 4]
and 5 is: [7, 6, 5]
and 6 is: [7, 6]
```

```
and 7 is: [7]
and 8 is: [7, 8]
A shortest path between 8
and 1 is: [8, 1]
```

```
and 2 is: [8, 1, 2]
and 3 is: [8, 1, 2, 3]
and 4 is: [8, 7, 4]
and 5 is: [8, 7, 6, 5]
```

```
and 6 is: [8, 7, 6]
and 7 is: [8, 7]
and 8 is: [8]
```

```
The distances between all nodes pairs of G are:
[[0. 1. 2. 2. 3. 2. 1. 1.]
 [1. 0. 1. 1. 2. 2. 1. 2.]
 [2. 1. 0. 1. 1. 1. 2. 3.]
 [2. 1. 1. 0. 2. 1. 1. 2.]
 [3. 2. 1. 2. 0. 1. 2. 3.]
 [2. 2. 1. 1. 1. 0. 1. 2.]
 [1. 1. 2. 1. 2. 1. 0. 1.]
 [1. 2. 3. 2. 3. 2. 1. 0.]]
```

Μερικές φορές θέλουμε να αποκτήσουμε μια συνολική εικόνα για τις πιθανές αποστάσεις που έχουν τα ζεύγη κόμβων του γραφήματος.

Εκκεντρότητα (eccentricity) $e(v)$ ενός κόμβου v ενός συνεκτικού γραφήματος G είναι η $\max_{u \in V(G)} d(u, v)$, (δηλαδή η μεγαλύτερη απόσταση που έχει ο v από όλες τις άλλες κορυφές u).

Διάμετρος (diameter) $d(G)$ ενός συνεκτικού γραφήματος G λέγεται το μήκος του μεγαλύτερου γεωδαιτικού του, (δηλαδή η μεγαλύτερη απόσταση ανάμεσα σε όλα τα δυνατά ζεύγη κόμβων).

Παρατήρηση: Προφανώς $d(G) = \max_{v \in V(G)} e(v)$.

Ακτίνα (radius) $r(G)$ ενός συνεκτικού γραφήματος G είναι η ελάχιστη εκκεντρότητα, ανάμεσα σε όλους τους κόμβους του G, δηλαδή $r(G) = \min_{v \in V(G)} e(v)$.

Ο v λέγεται **κεντρικός κόμβος** (central node) του συνεκτικού γραφήματος G, αν $e(v) = r(G)$. **Κέντρο** (center) του συνεκτικού γραφήματος ονομάζεται το σύνολο των κεντρικών του κόμβων.

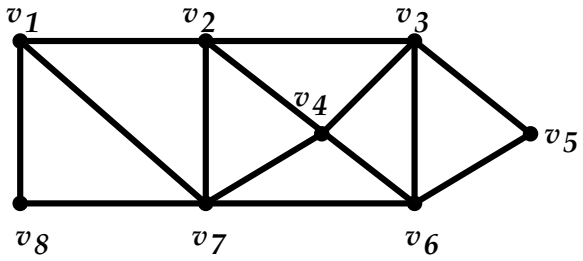
Ο v λέγεται **περιφερειακός κόμβος** (peripheral node) του συνεκτικού γραφήματος G, αν $e(v) = d(G)$. **Περιφερειακό σύνολο** (periphery set) του συνεκτικού γραφήματος ονομάζεται το σύνολο των περιφερειακών του κόμβων.

Για κάθε κόμβο v συμβολίζουμε με $s(v)$ το συνολικό άθροισμα όλων των αποστάσεών του από κάθε άλλο κόμβο u, δηλαδή

$$s(v) = \sum_{u \in G} d(u, v)$$

Βαρύκεντρο (barycenter ή median) ενός γραφήματος ονομάζεται το σύνολο των κόμβων v που ελαχιστοποιούν το συνολικό άθροισμα $s(v)$ όλων των αποστάσεων τους από κάθε άλλο κόμβο u.

Παράδειγμα:



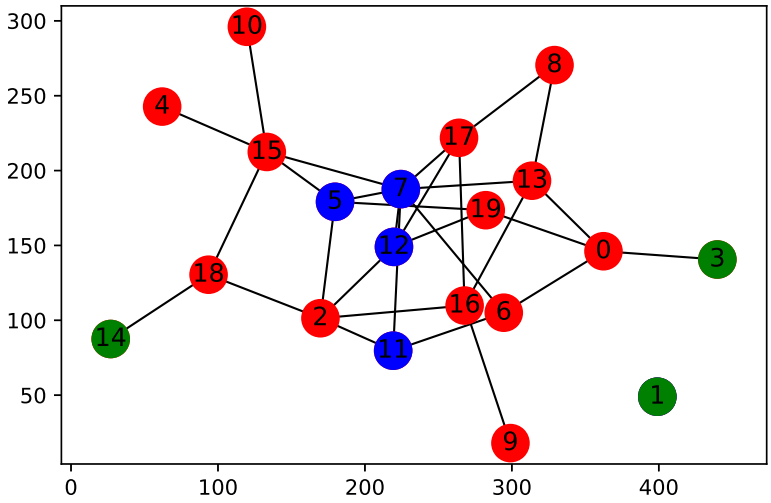
$d(G) = 3$.
$e(v_1) = e(v_3) = e(v_5) = e(v_8) = 3$.
$e(v_2) = e(v_4) = e(v_6) = e(v_7) = 2$.
$r(G) = 2$.
Κέντρο του $G = \{v_2, v_4, v_6, v_7\}$.
Περιφερειακό σύνολο του $G = \{v_1, v_3, v_5, v_8\}$.
$s(v_5) = s(v_8) = 14,\ s(v_1) = 12,\ s(v_3) = 11,\ s(v_2) = s(v_4) = s(v_6) = 10,\ s(v_7) = 9$.
Βαρύκεντρο του $G = \{v_7\}$.

Η βιβλιοθήκη `networkx` διαθέτει τις μεθόδους `radius(G)`, `diameter(G)`, `center(G)` και `periphery(G)` για τον υπολογισμό των αντίστοιχων εννοιών. Υπολογιστικά όμως συμφέρει να υπολογίσουμε μια φορά τις εκκεντρότητες των κορυφών του G με την μέθοδο `eccentricity(G)` και έπειτα, με βάση αυτές, τα υπόλοιπα στατιστικά. Επίσης, διαθέτει την μέθοδο `barycenter(G)` για τον υπολογισμό του βαρύκεντρου του G.

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

#create a random graph with n nodes and m edges
n,m = 20,30
G = nx.gnm_random_graph(n,m)
pos = nx.nx_agraph.graphviz_layout(G)
nx.draw_networkx(G,pos)

print("G has",nx.number_connected_components(G),"connected component(s)")

#for every connected component compute the eccentricities of its nodes
#and then its radius, diameter and center
Gcc = nx.connected_components(G)
for cc in Gcc:
    G1 = G.subgraph(cc)       #G1 is a connected component
    #find the eccentricities every node in G1
    eccdict = nx.eccentricity(G1)
    #print the eccentricities of every node in G1
    for v in G1:
        print("The eccentricity of node",v,"is",eccdict[v])

    all_values = eccdict.values()
    G1rad = min(all_values) #compute the radius of G1
    G1diam = max(all_values) #compute the diameter of G1
    #find center of G1
    G1center = []
```

```
    G1periphery = []
    for node in eccdict:
        eccnode = eccdict[node]
        if eccnode == G1rad:
            G1center.append(node)
        if eccnode == G1diam:
            G1periphery.append(node)
    print("The connected component",cc,"has radius",G1rad,"diameter",G1diam,"center",
   G1center,"and periphery",G1periphery)

    #color blue the central and peripheral nodes of G1
    G2 = G.subgraph(G1center)  #G2 is the induced subgraph of Gcenter
    nx.draw_networkx_nodes(G2,pos,node_color='blue',width=3.0)
    G3 = G.subgraph(G1periphery)  #G2 is the induced subgraph of Gcenter
    nx.draw_networkx_nodes(G3,pos,node_color='green',width=3.0)

plt.show()
```

Output:



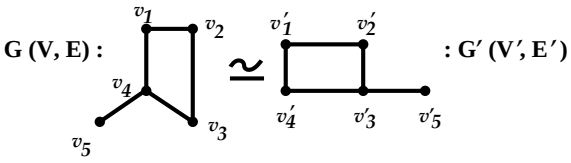

```
G has 2 connected component(s)
The eccentricity of node 0 is 5
The eccentricity of node 2 is 4
The eccentricity of node 3 is 6
The eccentricity of node 4 is 5
The eccentricity of node 5 is 3
The eccentricity of node 6 is 4
The eccentricity of node 7 is 3
The eccentricity of node 8 is 5
The eccentricity of node 9 is 5
The eccentricity of node 10 is 5
The eccentricity of node 11 is 3
The eccentricity of node 12 is 3
The eccentricity of node 13 is 4
The eccentricity of node 14 is 6
```

```
The eccentricity of node 15 is 4
The eccentricity of node 16 is 4
The eccentricity of node 17 is 4
The eccentricity of node 18 is 5
The eccentricity of node 19 is 4
The connected component {0, 2, 3, 4, 5,
   6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
   16, 17, 18, 19} has radius 3 diameter
   6 center [5, 7, 11, 12] and periphery
   [3, 14]
The eccentricity of node 1 is 0
The connected component {1} has radius 0
   diameter 0 center [1] and periphery
   [1]
```

6. ΙΣΟΜΟΡΦΑ ΓΡΑΦΗΜΑΤΑ

Τα γραφήματα δεσμών $G = (V, E)$ και $G' = (V', E')$ ονομάζονται **ισόμορφα** αν και μόνο αν υπάρχει αμφιμονοσήμαντη απεικόνιση $f : V \to V'$, με $\{x, y\} \in E \Leftrightarrow \{f(x), f(y)\} \in E'$. Αν δύο γραφήματα G και G' είναι ισόμορφα, θα γράφουμε $G \simeq G'$.

Παραδείγματα: Τα επόμενα γραφήματα είναι ισόμορφα:



διότι για την $f : V \to V'$ με

$$f(v_1) = v'_4,$$
$$f(v_2) = v'_1,$$
$$f(v_3) = v'_2,$$
$$f(v_4) = v'_3,$$
$$f(v_5) = v'_5,$$

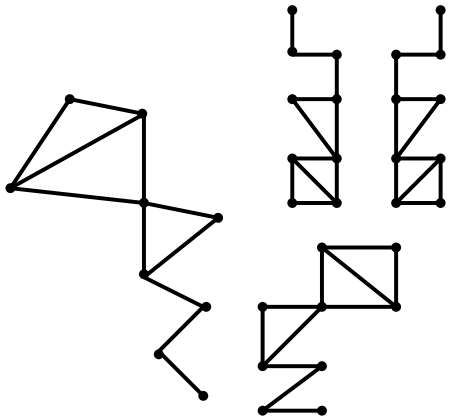
έχουμε πράγματι ότι

$$\{v_i, v_j\} \in E \Leftrightarrow \{f(v_i), f(v_j)\} \in E',$$

(για παράδειγμα:

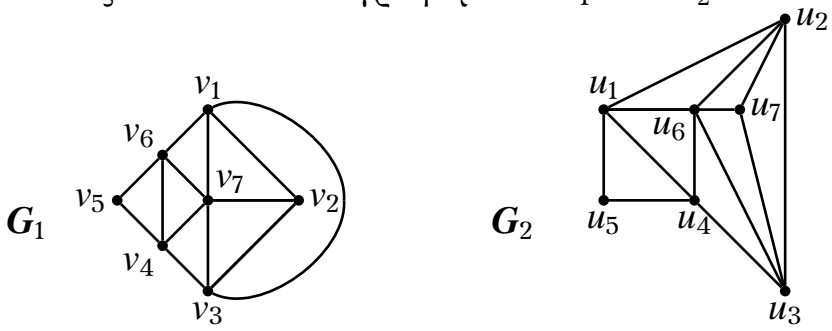
$$\{v_1, v_2\} \in E \text{ και } \{v'_4, v'_1\} \in E',$$
$$\{v_2, v_4\} \notin E \text{ και } \{v'_1, v'_3\} \notin E',$$
$$\{v_4, v_5\} \in E \text{ και } \{v'_3, v'_5\} \in E', \text{ κ.ο.κ.}).$$

Τα επόμενα γραφήματα είναι όλα ισόμορφα:



Αντίθετα τα επόμενα δύο γραφήματα δεν είναι ισόμορφα:

Παράδειγμα Να εξετασθεί αν τα γραφήματα G_1 και G_2 είναι ισόμορφα.



Λύση. Τα G_1, G_2 είναι ισόμορφα. Ένας ισομορφισμός f είναι ο εξής:

$$f(v_1) = u_2$$
$$f(v_2) = u_7$$
$$f(v_3) = u_3$$
$$f(v_4) = u_4$$
$$f(v_5) = u_5$$
$$f(v_6) = u_1$$
$$f(v_7) = u_6$$

□

Μπορούμε να ελέγξουμε αν τα G_1, G_2 είναι ισόμορφα χρησιμοποιώντας την μέθοδο `GraphMatcher` της βιβλιοθήκης networkx.

```
import networkx as nx
G1 = nx.Graph()
G1.add_nodes_from(range(1,8))
G1.add_edges_from([[1,2],[1,3],[1,6],[1,7],[2,3],[2,7],
                   [3,4],[3,7],[4,5],[4,6],[4,7],[5,6],[6,7]])
G2 = nx.Graph()
G2.add_nodes_from(range(1,8))
G2.add_edges_from([[1,2],[1,5],[1,4],[1,6],[2,3],[2,6],
                   [2,7],[3,4],[3,6],[3,7],[4,5],[4,6],[6,7]])
#Test whether the graphs are isomorphic
GM = nx.isomorphism.GraphMatcher(G1,G2)
if GM.is_isomorphic():  #If G1, G2 are isomorphic
    print("The graphs are isomorphic")
    print("An isomorphism between them is the following:")
    for i in G1:
        print("v",i,"->","u",GM.mapping[i])
    print(GM.mapping)
else:
    print("The graphs are not isomorphic")
```

Output:

```
The graphs are isomorphic
An isomorphism between them is the following:
v 1 -> u 3
v 2 -> u 7
v 3 -> u 2
v 4 -> u 1
v 5 -> u 5
v 6 -> u 4
```

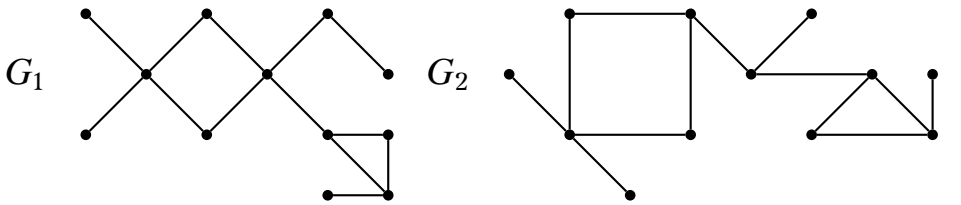
```
v 7 -> u 6
{4: 1, 3: 2, 1: 3, 6: 4, 5: 5, 7: 6, 2: 7}
```

Παρατήρηση Παρατηρήστε ότι ο αλγόριθμος ανακάλυψε έναν διαφορετικό ισομορφισμό από αυτόν που δόθηκε αρχικά.

Πρόταση 22. *Αν δύο γραφήματα G, H είναι ισόμορφα, τότε:*
i) Έχουν την ίδια ακολουθία βαθμών, και μάλιστα ισχύει ότι $d_G(v) = d_H(f(v))$, για κάθε $v \in V(G)$.
ii) Έχουν ισόμορφα υπογραφήματα.

Συνήθως η Πρόταση 22 χρησιμοποιείται αρνητικά, δηλαδή αν δύο γραφήματα δεν έχουν την ίδια ακολουθία βαθμών, ή/και δεν έχουν τα ίδια υπογραφήματα, τότε δεν είναι ισόμορφα. Για παράδειγμα,

(1)



$(4,4,3,3,2,2,2,2,1,1,1,1)$ $(4,3,3,3,3,2,2,2,1,1,1,1)$

$G_1 \not\simeq G_2$, διότι έχουν διαφορετικές ακολουθίες βαθμών.

(2)

Λύση. Δεν είναι ισόμορφα: Το G_1 περιέχει κύκλο μήκους 3 ενώ το G_2 όχι. □

Πάλι, μπορούμε να ελέγξουμε αν τα G_1, G_2 είναι ισόμορφα χρησιμοποιώντας την μέθοδο `GraphMatcher` της βιβλιοθήκης networkx.

```
import networkx as nx
G1 = nx.Graph()
G1.add_nodes_from(range(1,8))
G1.add_edges_from([[1,2],[1,7],[2,3],[2,6],[3,4],[3,5],[4,5],[5,6],[6,7]])
G2 = nx.Graph()
G2.add_nodes_from(range(1,8))
G2.add_edges_from([[1,2],[1,4],[1,7],[2,3],[3,4],[3,5],[4,6],[5,6],[6,7]])
#Test whether the graphs are isomorphic
GM = nx.isomorphism.GraphMatcher(G1,G2)
if GM.is_isomorphic():  #If G1, G2 isomorphic? then
    print("The graphs are isomorphic")
    print("An isomorphism between them is the following:")
    for i in G1:
        print("v",i,"->","u",GM.mapping[i])
    print(GM.mapping)
```

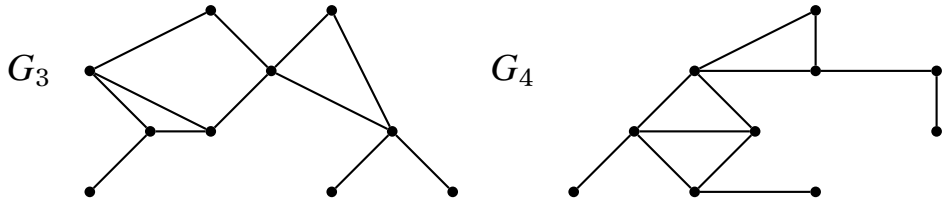
```
else:
    print("The graphs are not isomorphic")
    print("Degree sequence of G1:",sorted((d for n, d in G1.degree()),reverse=
    True))
    print("Degree sequence of G2:",sorted((d for n, d in G2.degree()),reverse=
    True))
```

Output:

```
The graphs are not isomorphic
Degree sequence of G1: [3, 3, 3, 3, 2, 2, 2]
Degree sequence of G2: [3, 3, 3, 3, 2, 2, 2]
```

Παρατήρηση Παρατηρήστε ότι παρόλο που τα γραφήματα είναι μη ισόμορφα έχουν την ίδια ακολουθία βαθμών. Στην περίπτωση όπου τα δύο γραφήματα δεν είναι ισόμορφα, η μέθοδος `GraphMatcher` δεν μας δίνει κάποια εξήγηση γιατί συμβαίνει αυτό.

(3)



$(4,4,3,3,3,2,2,1,1,1)$ $(4,4,3,3,3,2,2,1,1,1)$

$G_3 \neq G_4$, διότι, ενώ έχουν την ίδια ακολουθία βαθμών, έχουν διαφορετικά υπογραφήματα, (για παράδειγμα, το G_3 περιέχει δύο K_3, ενώ το G_4 περιέχει τρία K_3).

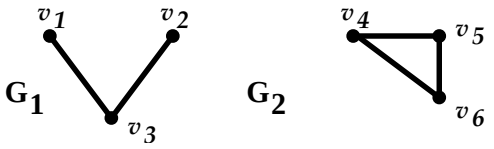
Παρατήρηση. Ο έλεγχος αν δύο μεγάλα γραφήματα είναι ισόμορφα ή όχι είναι υπολογιστικά δύσκολος μέχρι σήμερα (Απρίλιος 2021). Παρόλα αυτά υπάρχει η άποψη το πρόβλημα μπορεί να λυθεί γρηγορότερα αλλά δεν έχει βρεθεί ακόμα η κατάλληλη ιδέα.

7. ΠΡΑΞΕΙΣ

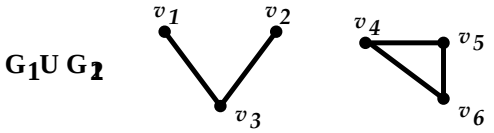
Έστω $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$.

Ένωση $G = G_1 \cup G_2$ είναι το γράφημα $G = (V, E)$ με $V = V_1 \cup V_2$ και $E = E_1 \cup E_2$.

Παράδειγμα: Έστω

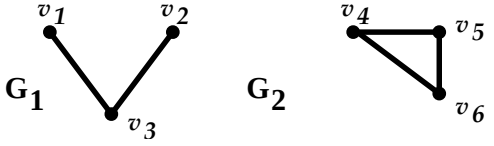


Η ένωση των γραφημάτων G_1 και G_2 είναι το γράφημα $G = G_1 \cup G_2$:

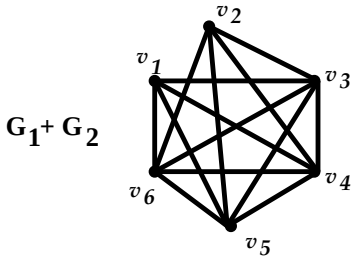


Άθροισμα $G = G_1 + G_2$ είναι το γράφημα $G = (V, E)$ με $V = V_1 \cup V_2$ και $E = E_1 \cup E_2 \cup \{\{v_i, v_j\} : v_i \in V_1, v_j \in V_2\}$ (δηλαδή το $G_1 + G_2$ είναι το $G_1 \cup G_2$ μαζί με όλους τους δεσμούς που ενώνουν τα στοιχεία του V_1 με στοιχεία του V_2).

Παράδειγμα: Έστω



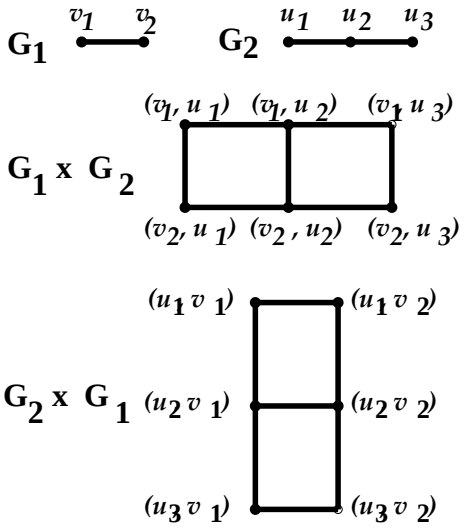
Το άθροισμα των γραφημάτων G_1 και G_2 είναι το γράφημα $G = G_1 + G_2$:



Γινόμενο $G = G_1 \times G_2$ είναι το γράφημα $G = (V, E)$ με $V = V_1 \times V_2$ και αν $\alpha = (v_1, u_1)$, $\beta = (v_2, u_2) \in V$ τότε $\{\alpha, \beta\} \in E$ αν και μόνο αν:

$(v_1 = v_2$ και $\{u_1, u_2\} \in E(G_2))$, ή $(u_1 = u_2$ και $\{v_1, v_2\} \in E(G_1))$.

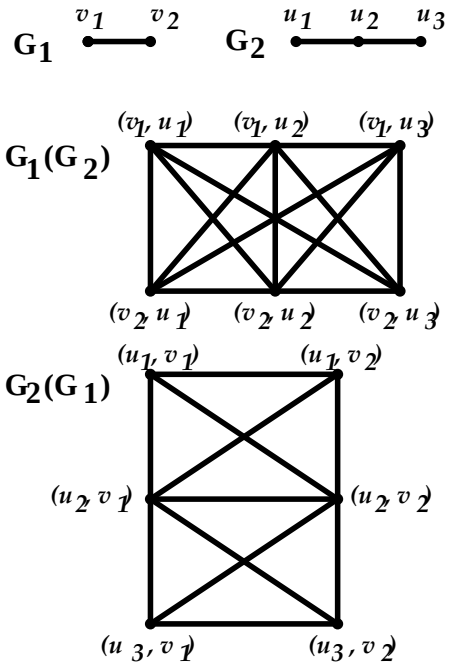
Παράδειγμα:



Σύνθεση $G = G_1(G_2)$ είναι το γράφημα $G = (V, E)$ με $V = V_1 \times V_2$ και αν $\alpha = (v_1, u_1)$, $\beta = (v_2, u_2) \in V$ τότε $\{\alpha, \beta\} \in E$ αν και μόνο αν:

$$(\{v_1, v_2\} \in E(G_1)), \text{ ή } (v_1 = v_2 \text{ και } \{u_1, u_2\} \in E(G_2)).$$

Παράδειγμα:

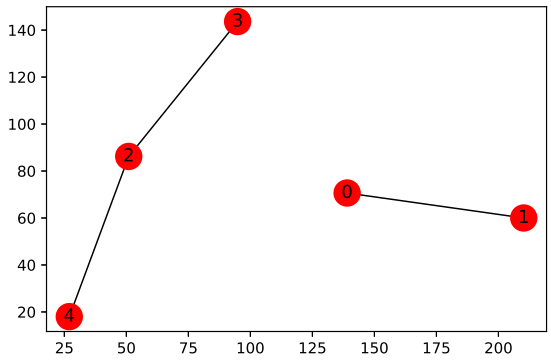


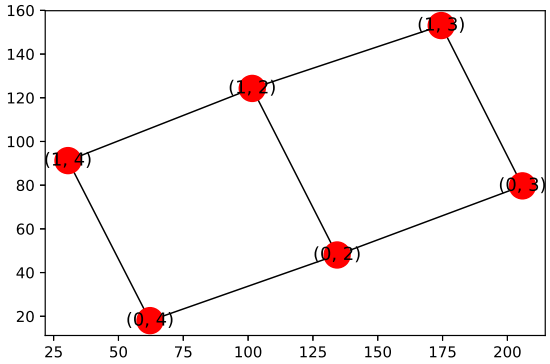
Μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη networkx για να υλοποίησουμε τις παραπάνω πράξεις.

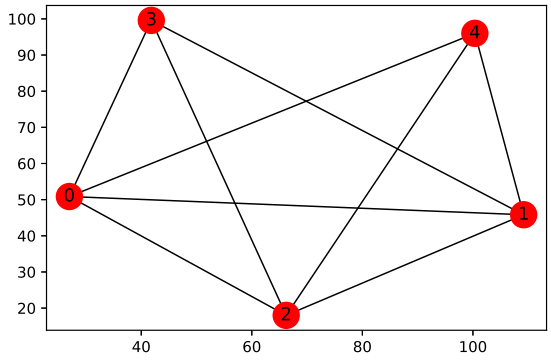
```
import networkx as nx
import matplotlib.pyplot as plt

G1 = nx.Graph()
V1 = [0,1]
E1 = [[0,1]]
G1.add_nodes_from(V1)
G1.add_edges_from(E1)

```

```
G2 = nx.Graph()
V2 = [2,3,4]
E2 = [[2,3],[2,4]]
G2.add_nodes_from(V2)
G2.add_edges_from(E2)

#The disjoint union works even if V1 and V2 are not disjoint
#It renumbers the vertices starting from G1 and then continuing
#with G2
Gunion = nx.disjoint_union(G1, G2)
pos = nx.nx_agraph.graphviz_layout(Gunion)
nx.draw_networkx(Gunion,pos)
plt.show()

Gproduct = nx.cartesian_product(G1,G2)
pos = nx.nx_agraph.graphviz_layout(Gproduct)
nx.draw_networkx(Gproduct,pos)
plt.show()

def graphsum(G1, G2):
    Gsum = nx.disjoint_union(G1,G2)
    for i in range(G1.order()):
        for j in range(G1.order(), G1.order() + G2.order()):
            Gsum.add_edge(i, j)
    return Gsum
Gsum = graphsum(G1, G2)
pos = nx.nx_agraph.graphviz_layout(Gsum)
nx.draw_networkx(Gsum,pos)
plt.show()
```

Output:







8. ΑΝΑΠΟΦΕΥΚΤΕΣ ΙΔΙΟΤΗΤΕΣ (*)

Complete disorder is impossible. – Theodore S. Motzkin (1908 – 1970).

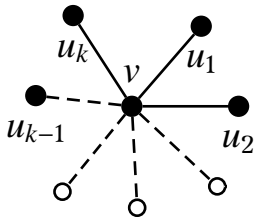
Στα γραφήματα δεσμών και ιδιαίτερα στα μεγάλα γραφήματα εμφανίζονται πάντα ή σχεδόν πάντα ορισμένα χαρακτηριστικά που εκ πρώτης όψεως μοιάζουν με συμπτώσεις. Υπάρχει μια ολόκληρη κατηγορία αποτελεσμάτων που δείχνουν ότι όσο τυχαίο να είναι ένα γράφημα πάντα ή σχεδόν πάντα θα εμφανίζει κάποια "κανονικότητα", πάντα ή σχεδόν πάντα θα περιέχει κάποιες "συμπτώσεις".

Στην ενότητα αυτή δίνονται δύο πολύ απλά παραδείγματα τέτοιων αποτελεσμάτων που ισχύουν πάντα.

Πρόταση 23. *Σε κάθε συνεκτικό γράφημα δεσμών $G = (V, E)$ με $|V| \geq 2$, υπάρχουν τουλάχιστον δύο κορυφές με τον ίδιο βαθμό οι οποίες είναι γειτονικές ή έχουν κοινό γείτονα.*

Λύση. Έστω $\Delta(G) = k$ ο μέγιστος βαθμός των κορυφών του γραφήματος και έστω $v \in V$ μια κορυφή με βαθμό k.

Η v έχει k γειτονικές κορυφές $u_1, u_2, \ldots, u_k$ και κάθε μία από αυτές έχει βαθμό από 1 μέχρι k.

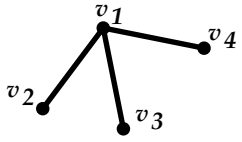


Διακρίνουμε δύο περιπτώσεις:

- Αν κάποια από τις u_1, u_2, …, u_k έχει βαθμό k, τότε το γράφημα περιέχει δύο γειτονικές κορυφές με τον ίδιο βαθμό.
- Αν καμία από τις u_1, u_2, …, u_k δεν έχει βαθμό k, τότε υπάρχουν $k - 1$ δυνατές τιμές βαθμών για τους k γείτονες της v, οπότε από την αρχή του περιστερεώνα, δύο τουλάχιστον από αυτές έχουν τον ίδιο βαθμό και κοινό γείτονα την v. □

Πρόταση 24. *Αν $|V| \geq 6$, τότε ή το G ή το G^c περιέχει τουλάχιστον ένα υπογράφημα ισόμορφο με το K_3, (δηλαδή ένα τρίγωνο).*

Απόδειξη. Έστω $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ και έστω ότι ο v_1 είναι ενωμένος με τουλάχιστον τρεις κόμβους στο G. Έστω λοιπόν: $\{v_1, v_2\}$, $\{v_1, v_3\}$, $\{v_1, v_4\} \in E$.



Αν $\{v_2, v_3\} \in E$ τότε οι v_1, v_2, v_3 είναι κορυφές τρίγωνου στο G.

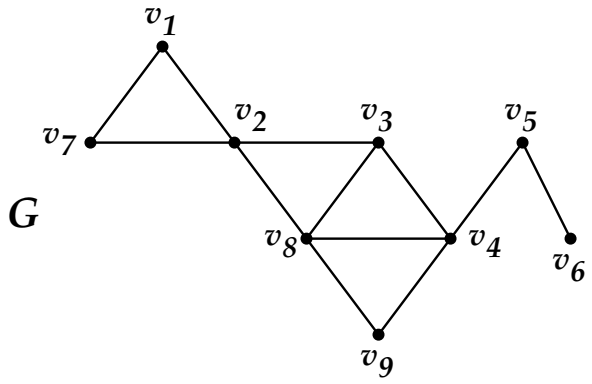
Όμοια αν $\{v_2, v_4\} \in E$, ή $\{v_3, v_4\} \in E$.

Αν τώρα $\{v_2, v_3\}$, $\{v_2, v_4\}$, $\{v_3, v_4\} \notin E$, τότε θα έχουμε $\{v_2, v_3\}$, $\{v_2, v_4\}$, $\{v_3, v_4\} \in E^c$ και άρα οι v_2, v_3, v_4 είναι κορυφές τρίγωνου στο G^c.

Αν τέλος ο v_1 είναι ενωμένος με λιγότερους από τρεις κόμβους στο G, θα είναι ενωμένος με τουλάχιστον τρεις κόμβους στο G^c. Τότε παίρνουμε το αντίστοιχο αποτέλεσμα, ξεκινώντας από το G^c. □

Ασκήσεις προς επίλυση

(1) Δίδεται το γράφημα G



Να ευρεθούν:

i) Η απόσταση $d(v_2, v_5)$.

ii) Δύο γεωδεσικά ανάμεσα στους κόμβους v_2 και v_4.

iii) Η διάμετρος $d(G)$.

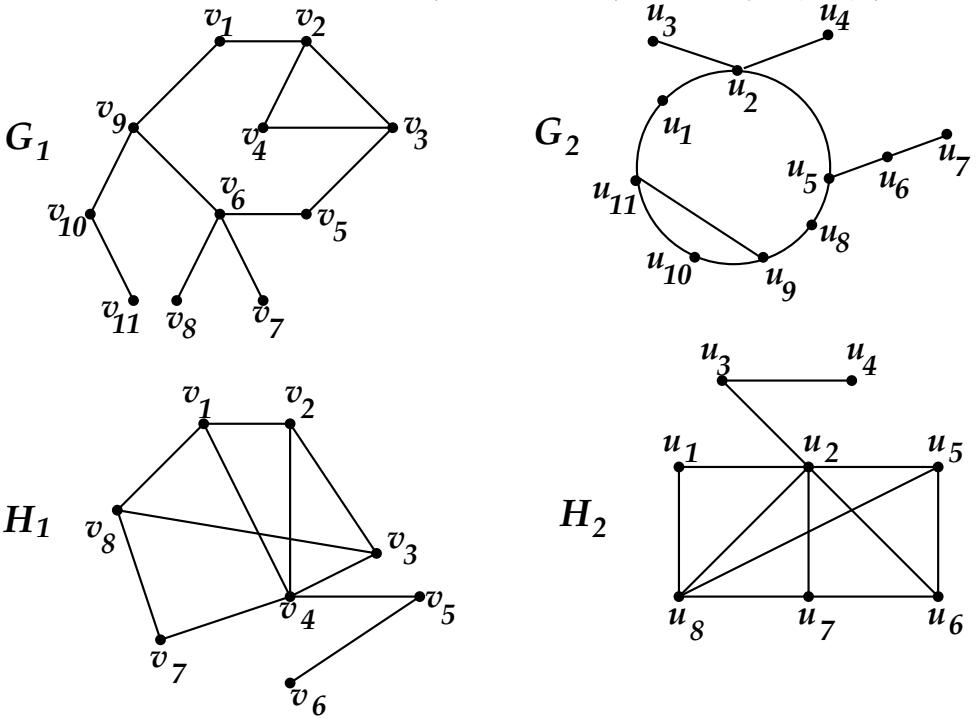
iv) Η ακτίνα $r(G)$.

v) Το κέντρο του G.

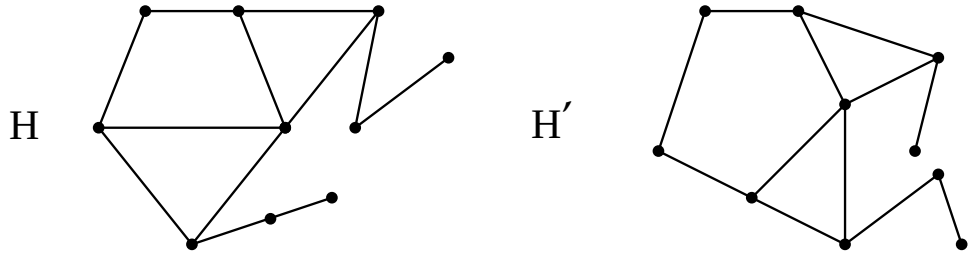
vi) Το περιφερειακό σύνολο του G.

vii) Το βαρύκεντρο του G.

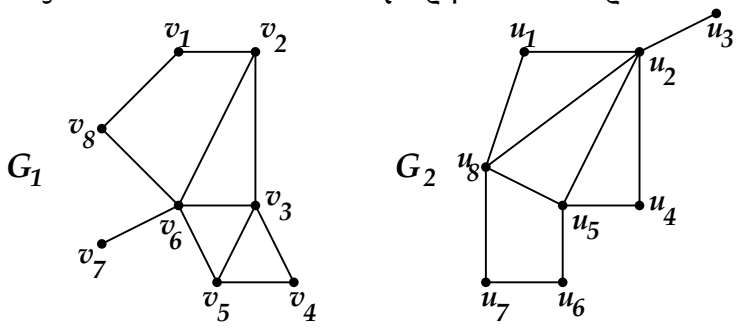
(2) Να εξετασθεί αν είναι ισόμορφα τα παρακάτω ζεύγη γραφημάτων.

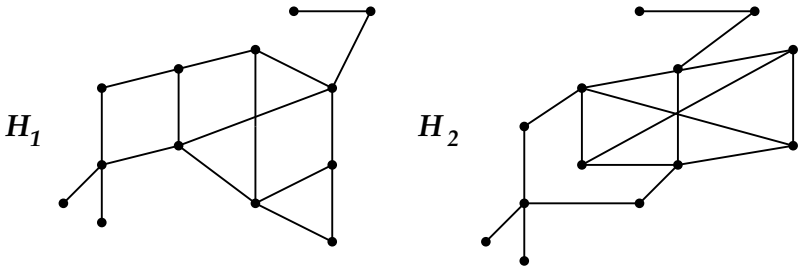


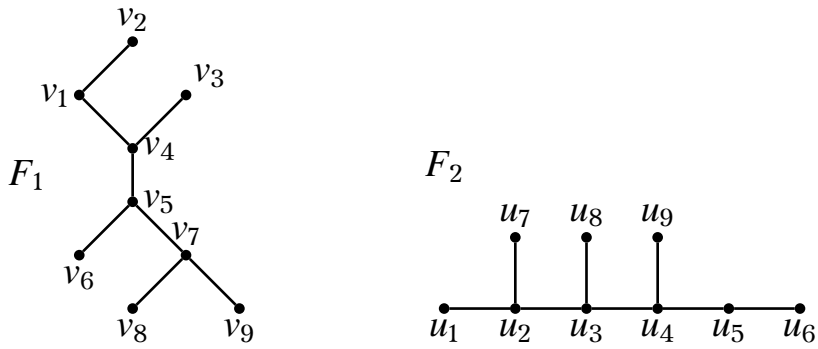
(3) Να δειχθεί ότι δεν είναι ισόμορφα τα παρακάτω γραφήματα:



(4) Να εξετασθεί αν είναι ισόμορφα τα παρακάτω ζεύγη γραφημάτων.



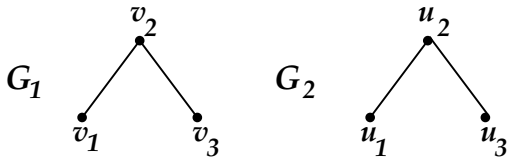




(5) Για τα γραφήματα

i)

ii)



να ορισθούν τα $G_1 \cup G_2$, $G_1 + G_2$, $G_1 \times G_2$, $G_1(G_2)$.