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 1

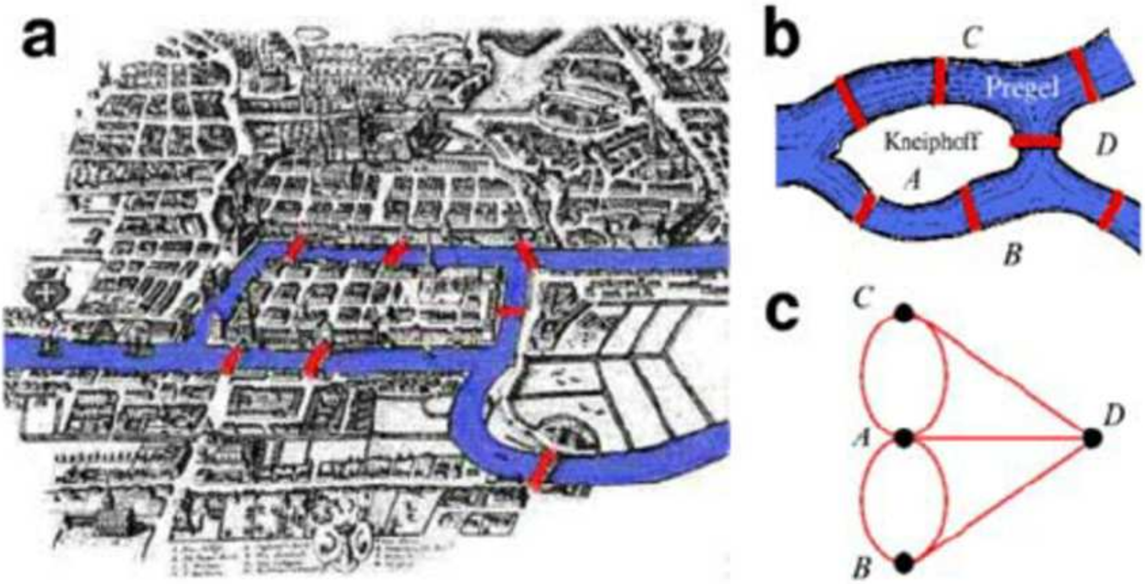
ΠΕΙΡΑΙΑΣ 2023

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

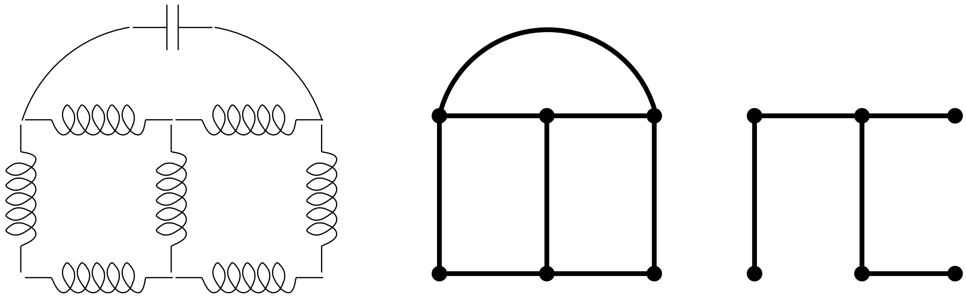
ΕΙΣΑΓΩΓΗ - ΙΣΤΟΡΙΚΟ

Euler (1736): Γέφυρες του Konigsberg

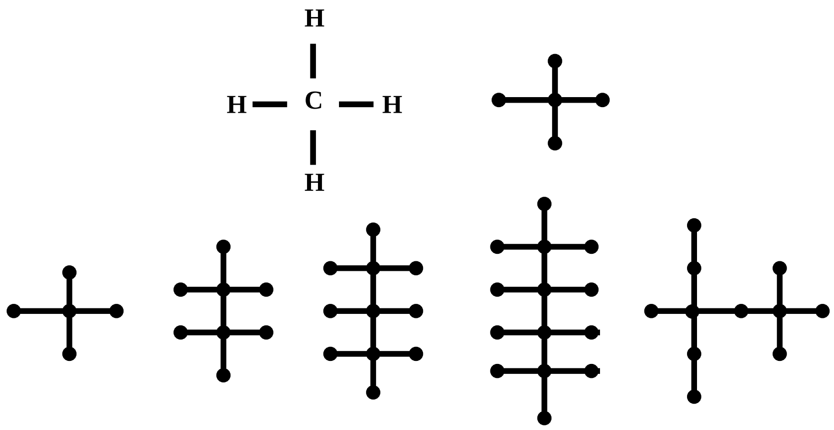
Μπορεί κάποιος να περάσει ακριβώς μια φορά από κάθε γέφυρα;



Kirchhoff (1847): Γενετικό δένδρο



Cayley (1857): Πλήθος κορεσμένων υδρογονάνθρακων C_nH_{2n+2}



Μερικές από τις εφαρμογές της Θεωρίας Γραφημάτων:

Πληροφορική (Δένδρα, Δυαδικά δένδρα, Διατεταγμένα δένδρα, Διάτρεξη (διάσχιση) δένδρων, Προγραμματισμός, Συνδεσμολογία κ.λπ.).

Αλγόριθμοι (Αλγόριθμοι γραφημάτων, Αναζήτηση πρώτα κατά πλάτος, Αναζήτηση πρώτα κατά βάθος, Τοπολογική διάταξη, Κατάταξη έργων με προθεσμίες κ.λπ.).

Διοίκηση Επιχειρήσεων (Οργανογράμματα, Κεντρικά σημεία κ.λπ.).

Οδοποιΐα (Οδικά δίκτυα - χωρητικότητα - μέγιστη ροή, Σηματοδότηση δρόμων).

Υδραυλικά (Δίκτυα - χωρητικότητα -μέγιστη ροή).

Ιστορία - Κοινωνιολογία (Γενεαλογικά δένδρα, Φιλία (γραφήματα δεσμών), Έρωτας (γραφήματα τόξων)).

ΠΡΩΤΟ ΜΕΡΟΣ: ΕΙΣΑΓΩΓΗ ΣΤΑ ΓΡΑΦΗΜΑΤΑ ΔΕΣΜΩΝ

1. ΒΑΣΙΚΟΙ ΟΡΙΣΜΟΙ

Κάθε δυάδα $G = (V(G), E(G))$, ή (V, E), ή (X, E) όπου V είναι ένα μη κενό σύνολο και E είναι ένα σύνολο από (μη διατεταγμένα) ζεύγη $\{u, v\}$, $u, v \in V$ ονομάζεται **γράφημα δεσμών** (graph), ή **απροσανατόλιστο γράφημα** (undirected graph).

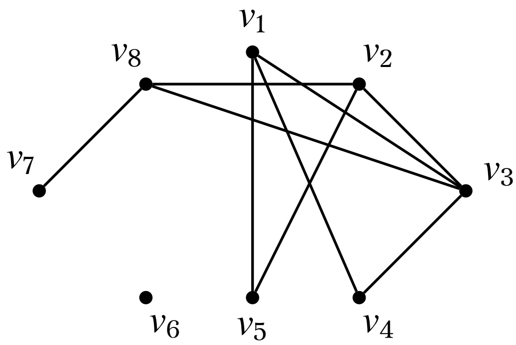
Τα στοιχεία του V καλούνται **κορυφές**, ή **σημεία**, ή **κόμβοι** (vertices, nodes, points), ενώ τα στοιχεία του E καλούνται **δεσμοί**, ή **γραμμές**, ή **χορδές**, ή **πλευρές**, ή **ακμές** (edges, lines).

Θα ασχοληθούμε εδώ με **πεπερασμένα γραφήματα**, (δηλαδή $|V| \in \mathbb{N}^*$). Το E μπορεί να είναι $\emptyset$. Συχνά γράφουμε $|V| = n$ και $|E| = m$. Ο πληθάριθμος $|V|$ ονομάζεται **τάξη** (order) του γραφήματος, ενώ ο πληθάριθμος $|E|$ ονομάζεται **μέγεθος** (size) του γραφήματος.

Αν $\{u, v\} \in E$, λέμε ότι τα u, v είναι **άκρα** του δεσμού $\{u, v\}$ ή, ισοδύναμα, ότι το u (και το v) **καλύπτει** τον δεσμό $\{u, v\}$ και τα u, v ονομάζονται **γειτονικά**. Αντίστοιχα, αν δύο δεσμοί έχουν κοινή μια κορυφή, λέμε ότι και αυτοί είναι **γειτονικοί**.

Παράδειγμα: Η δυάδα $G = (V, E)$ όπου $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ και $E = \{\{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_5\}, \{v_2, v_3\}, \{v_2, v_5\}, \{v_2, v_8\}, \{v_3, v_4\}, \{v_3, v_8\}, \{v_7, v_8\}\}$ είναι ένα γράφημα δεσμών.

Η γραφική του απεικόνιση είναι η ακόλουθη:



Αν οι u, v ταυτίζονται έχουμε ένα **βρόχο**.

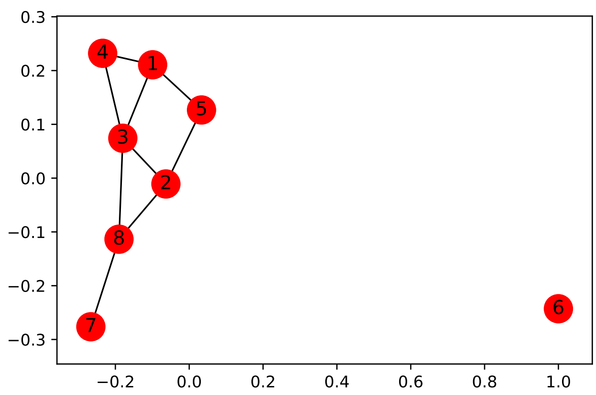
Παρατήρηση: Δεδομένου ότι σε ένα σύνολο επιτρέπεται μία μόνο εμφάνιση κάθε στοιχείου του, από τον ορισμό του γραφήματος δεσμών προκύπτει ότι σε αυτό δεν επιτρέπονται ούτε βρόχοι, ούτε πολλαπλοί δεσμοί που να συνδέουν το ίδιο ζεύγος κορυφών. Τα γραφήματα αυτά ονομάζονται **απλά γραφήματα** (simple graphs) και με τέτοια θα ασχοληθούμε, εκτός αν αναφερθεί ρητά το αντίθετο.

Στο επόμενο πρόγραμμα χρησιμοποιούμε την βιβλιοθήκη networkx της Python για να ορίσουμε το γράφημα G του πρώτου παραδείγματος.

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

G = nx.Graph() #Create an empty graph

V = [1,2,3,4,5,6,7,8] #V is the set of vertices of G
E = [[1,3],[1,4],[1,5],[2,3],[2,5],[2,8],[3,4],[3,8],[7,8]] #E is the set of edges of
    G

G.add_nodes_from(V)
G.add_edges_from(E)

print("G has order |V(G)|=",G.order(),"and size |E(G)|=",G.size())
print("V(G):",G.nodes()) #Print the nodes of G
print("E(G):", G.edges()) #Print the edges of G
for v in G:
    print("The neighbors of", v, "are:", list(G.neighbors(v)))

nx.draw_networkx(G) #Draw the graph G
plt.savefig("lect01a.eps") #Save the drawing of G
plt.show() #Show the drawing of G on screen
```

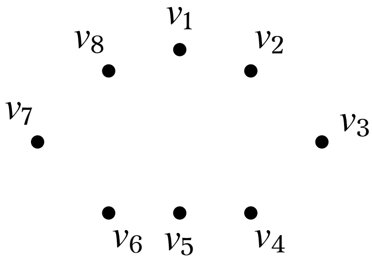
Output:

```
G has order |V(G)|= 8 and size |E(G)|= 9
V(G): [1, 2, 3, 4, 5, 6, 7, 8]
E(G): [(1, 3), (1, 4), (1, 5), (2, 3), (2, 5), (2, 8), (3, 4), (3, 8), (7, 8)]
The neighbors of 1 are: [3, 4, 5]
The neighbors of 2 are: [3, 5, 8]
The neighbors of 3 are: [1, 2, 4, 8]
The neighbors of 4 are: [1, 3]
The neighbors of 5 are: [1, 2]
The neighbors of 6 are: []
The neighbors of 7 are: [8]
The neighbors of 8 are: [2, 3, 7]
```



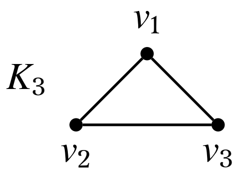
ΜΟΡΦΕΣ ΓΡΑΦΗΜΑΤΩΝ

1) **Μηδενικό γράφημα**: $G = (V, E)$ με $E = \emptyset$.
 Παράδειγμα:

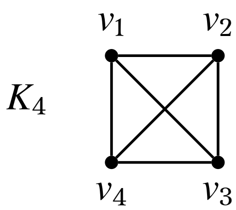


2) **Τετριμμένο γράφημα**: $G = (V, E)$ με $|V| = 1$.

3) **Πλήρες γράφημα**: $G = (V, E)$ τέτοιο ώστε για κάθε $u, v \in V$ με $u \neq v$ ισχύει ότι $\{u, v\} \in E$. Το πλήρες γράφημα με n κορυφές συμβολίζεται με K_n.
 Παράδειγμα: Το γράφημα K_3 είναι το:



ενώ το γράφημα K_4 είναι το :



Παρατήρηση: Το K_n έχει n κορυφές και $\binom{n}{2}$ δεσμούς (όσα και τα ζευγάρια του $[n]$).
 Στην βιβλιοθήκη networkx το πλήρες γράφημα με n κορυφές κατασκευάζεται χρησιμοποιώντας την μέθοδο `complete_graph(n)`, ή χρησιμοποιώντας τις επόμενες εντολές:

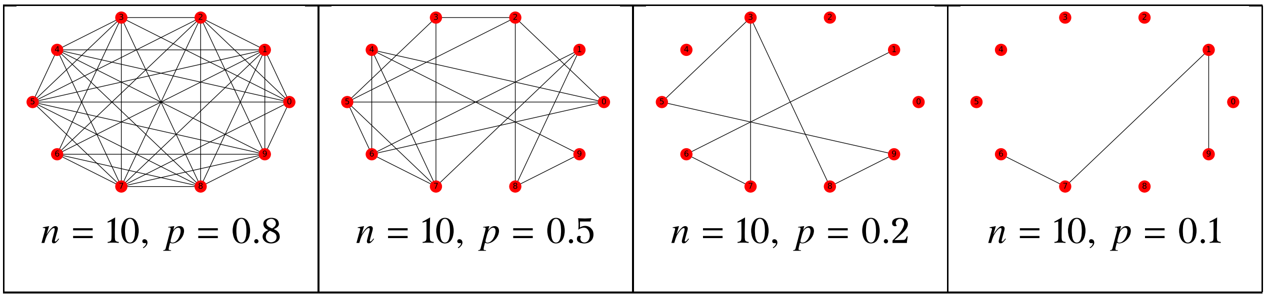
```
import networkx as nx
import matplotlib.pyplot as plt

n = 7 #number of vertices

Kn = nx.complete_graph(n)
nx.draw_circular(Kn,with_labels=True)
plt.show()

Kn = nx.Graph()
Kn.add_nodes_from(range(1,n+1))
for i in range(1,n+1):
    for j in range(i+1,n+1):
        Kn.add_edge(i,j)
nx.draw_circular(Kn,with_labels=True)
plt.show()
```

4) **Τυχαίο γράφημα**: Κάποιες φορές καλούμαστε να δοκιμάσουμε αλγορίθμους ή ιδέες μας πάνω σε διάφορα παραδείγματα γραφημάτων. Μπορούμε να φτιάχνουμε τέτοια "τυχαία" παραδείγματα χρησιμοποιώντας έτοιμες μεθόδους της βιβλιοθήκης networkx ή γράφοντας δικές μας μεθόδους, όπως στα επόμενα παραδείγματα.

Η πιο απλή ιδέα κατασκευής ενός τυχαίου γραφήματος με n κορυφές είναι το **μοντέλο των Erdős - Renyi** όπου για κάθε ζεύγος κορυφών επιλέγουμε να δημιουργήσουμε τον δεσμό που τις συνδέει με πιθανότητα p.

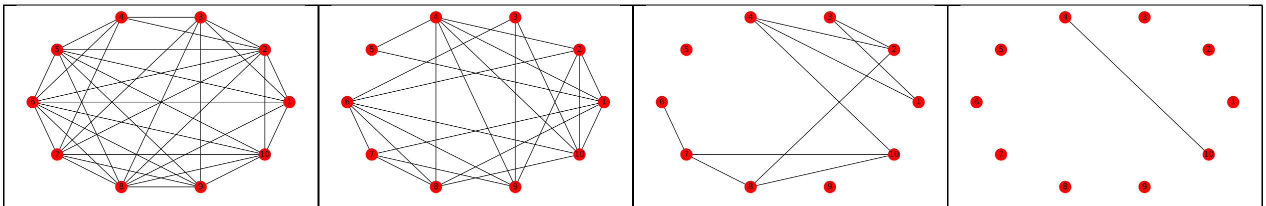
Ένα τέτοιο γράφημα προκύπτει χρησιμοποιώντας την μέθοδο `gnp_random_graph(n,p)`:

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

def random_gnp_graph(n,p,name):
    R = nx.gnp_random_graph(n,p)
    nx.draw_circular(R,with_labels=True)
    plt.savefig(name+".eps")
    plt.show()

random_gnp_graph(10,0.8,"R1")
random_gnp_graph(10,0.5,"R2")
random_gnp_graph(10,0.2,"R3")
random_gnp_graph(10,0.1,"R4")
```



Μια απλή υλοποίηση της μεθόδου `gnp_random_graph(n,p)`:

```
import networkx as nx
import matplotlib.pyplot as plt
import random #random numbers

def random_gnp_graph2(n,p,name):
    R = nx.Graph()
    R.add_nodes_from(range(1,n+1))
    for i in range(1,n+1):
        for j in range(i+1,n+1):
            if random.uniform(0,1) <= p:
                R.add_edge(i,j)
    nx.draw_circular(R,with_labels=True)
    plt.savefig(name+".eps")
    plt.show()

random_gnp_graph2(10,0.8,"R9")
random_gnp_graph2(10,0.5,"R10")
random_gnp_graph2(10,0.2,"R11")
random_gnp_graph2(10,0.1,"R12")
```

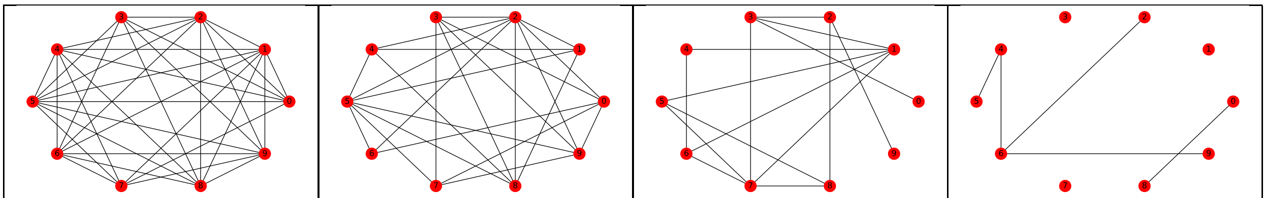


$n = 10,\ p = 0.8$ | $n = 10,\ p = 0.5$ | $n = 10,\ p = 0.2$ | $n = 10,\ p = 0.1$

Παρατήρηση: Επειδή κάθε ένας από τους $\binom{n}{2}$ πιθανούς δεσμούς επιλέγεται με πιθανότητα p έπεται ότι το τυχαίο γράφημα που προκύπτει κατά μέσο όρο αναμένεται να έχει $p\binom{n}{2}$ δεσμούς.

Στην περίπτωση όπου θέλουμε το τυχαίο γράφημα να έχει n κορυφές και ακριβώς m δεσμούς μπορούμε να χρησιμοποιήσουμε την μέθοδο `gnm_random_graph(n,m)`:

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

def random_gnm_graph(n,m,name):
    R = nx.gnm_random_graph(n,m) #0 <= m <= n(n-1)/2
    nx.draw_circular(R,with_labels=True)
    plt.savefig(name+".eps")
    plt.show()

random_gnm_graph(10,35,"R5")
random_gnm_graph(10,25,"R6")
random_gnm_graph(10,15,"R7")
random_gnm_graph(10,5,"R8")
```

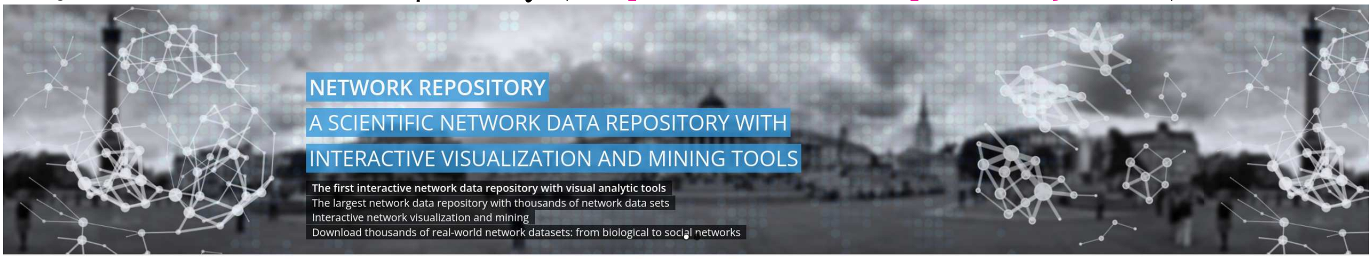


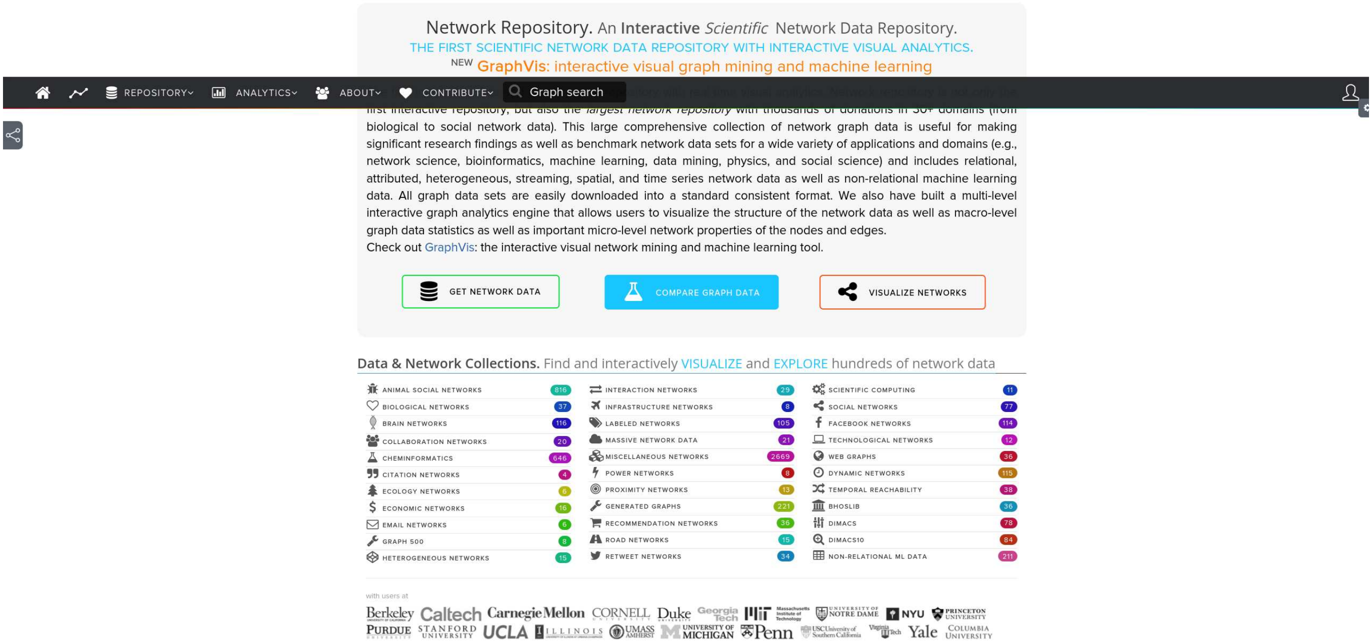
$n = 10,\ m = 35$ | $n = 10,\ m = 25$ | $n = 10,\ m = 15$ | $n = 10,\ m = 5$

Παρατήρηση (*): Μια υλοποίηση της μεθόδου `gnm_random_graph(n,m)` μπορεί να γίνει κατασκευάζοντας έναν τυχαίο υποσύνολο του $[\binom{n}{2}]$ με m στοιχεία.

ΠΗΓΕΣ ΑΝΟΙΧΤΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΓΡΑΦΗΜΑΤΑ

Υπάρχουν αρκετοί ιστότοποι με συλλογές ανοιχτών δεδομένων που αφορούν γραφήματα τα οποία εμφανίζονται σε πραγματικές καταστάσεις. Ένας τέτοιος ιστότοπος είναι το Network Repository (https://networkrepository.com/):





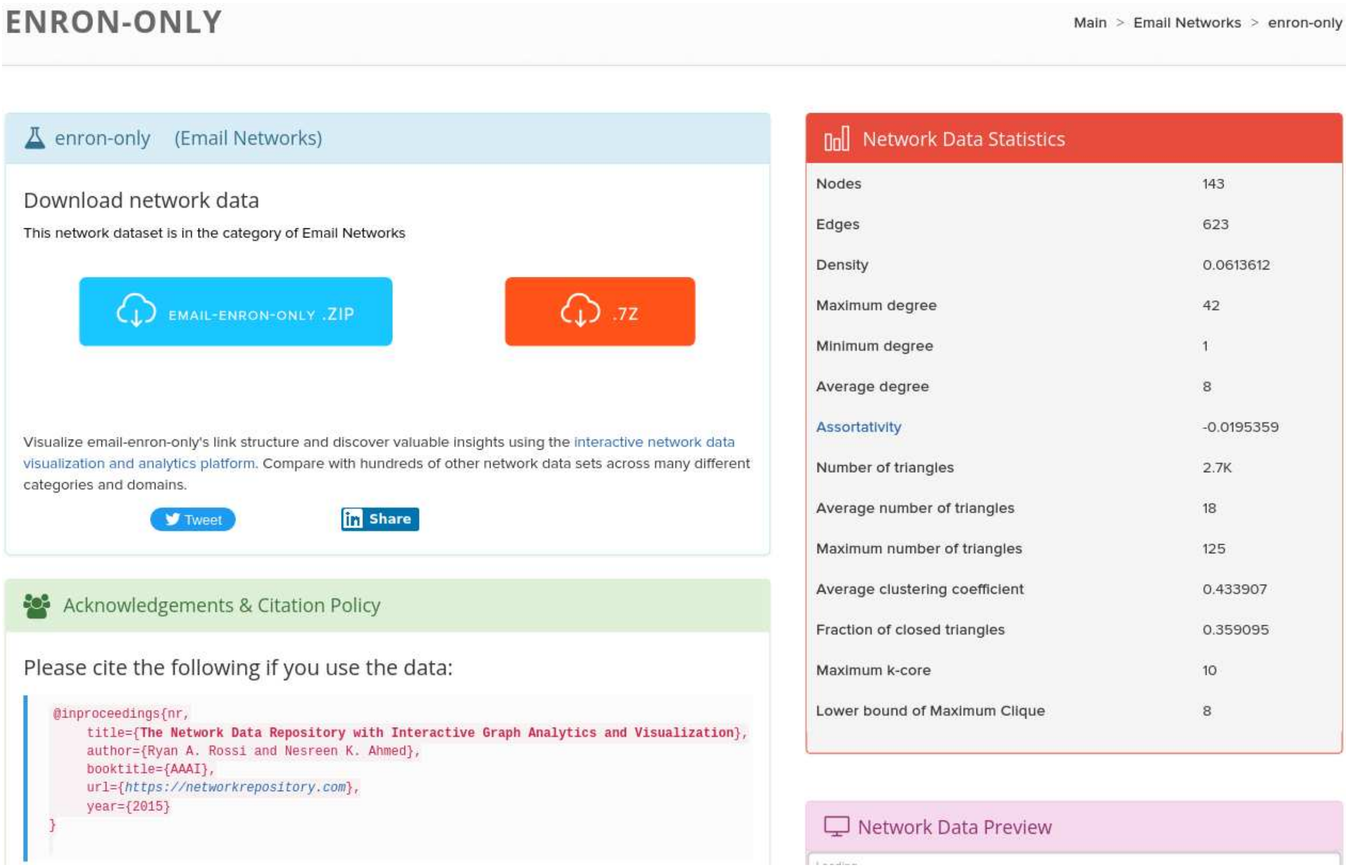
Συνήθως, τα δεδομένα που αφορούν τα γραφήματα είναι διαθέσιμα σε μορφή αρχείου κειμένου που περιέχει λίστα με δεσμούς του γραφήματος (δύο αριθμοί σε κάθε γραμμή, οι αριθμοί δηλώνουν τις ετικέτες των κορυφών που ενώνει ο δεσμός).

Η βιβλιοθήκη `networkx` έχει την μέθοδο `read_edgelist('filename.edges')` για την ανάγνωση του γραφήματος από το αρχείο `filename.edges` το οποίο περιέχει την λίστα των δεσμών του γραφήματος.

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

G = nx.read_edgelist('realgraphs/email-EU.edges')

print("Number of nodes:", G.order(), "Number of edges:", G.size())
```

Output:

```
Number of nodes: 32430 Number of edges: 54397
```

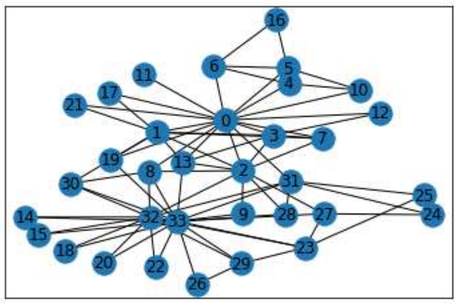


Σχήμα 1. Παράδειγμα από το Network Repository

Επίσης, αρκετά δημοφιλής είναι ο τύπος αρχείου `mtx` (matrix market file) το οποίο εκτός από την λίστα των δεσμών περιέχει το πλήθος των δεσμών και τον συνολικό αριθμό των κορυφών, επίσης μπορεί να περιέχει σχόλια. Για την ανάγνωση αρχείων `mtx` μπορεί να χρησιμοποιηθεί η μέθοδος `mmread('filename.mtx')` της βιβλιοθήκης `scipy.io`.

```
import networkx as nx
import matplotlib.pyplot as plt
from scipy.io import mmread

a = mmread('realgraphs/soc-karate.mtx')
G = nx.Graph(a)

pos = nx.layout.kamada_kawai_layout(G)
nx.draw_networkx(G,pos)

plt.show()
```

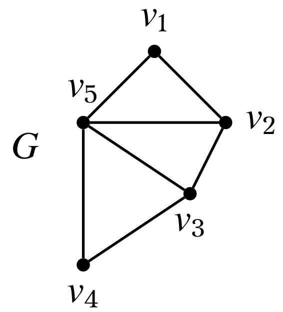
Output:

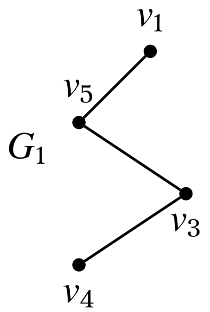


ΥΠΟΓΡΑΦΗΜΑΤΑ

1) **Υπογράφημα** (subgraph) του $G = (V, E)$: Ένα γράφημα $G_1 = (V_1, E_1)$ με $V_1 \subseteq V$ και $E_1 \subseteq E$.

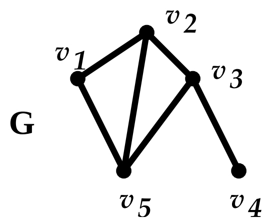
Παράδειγμα:

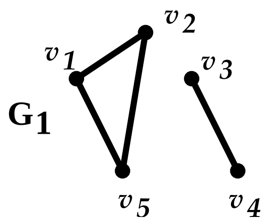




2) **Γενετικό** (ή **γενννητικό**, ή **μερικό**) **γράφημα**, ή **γράφημα ζεύξης** (spanning graph) του $G = (V, E)$: Ένα γράφημα $G_1 = (V, E_1)$ με $E_1 \subseteq E$.

Παράδειγμα:

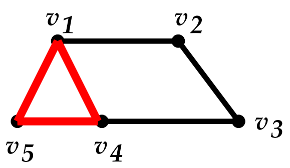


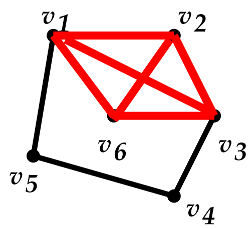


3) **Κλίκα** (clique): Κάθε πλήρες υπογράφημα του G.

Μέγιστη κλίκα: Κλίκα με το μέγιστο δυνατό αριθμό κόμβων.

Παραδείγματα:

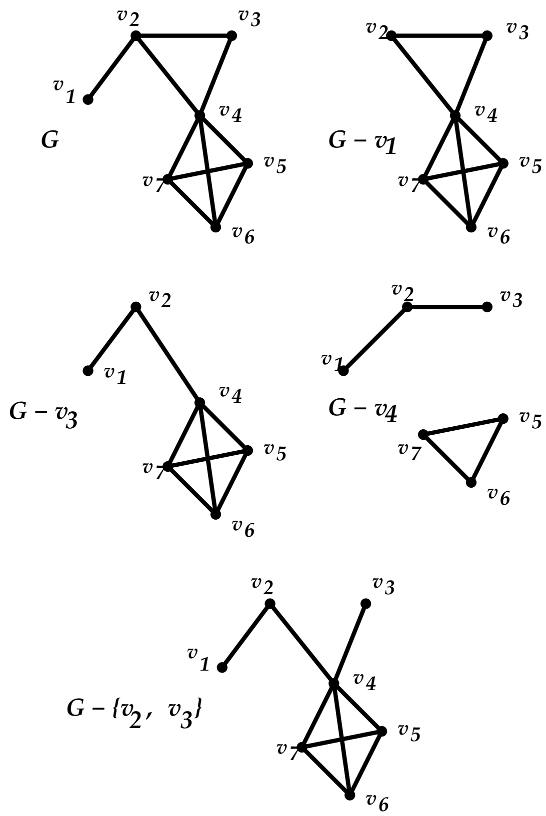




Οι μέγιστες κλίκες των δύο παραπάνω γραφημάτων είναι οι K_3 και K_4 αντίστοιχα.

4) Αν $G = (V, E)$ και $v \in V$, $e \in E$ ορίζουμε τα υπογραφήματα $\mathbf{G - v}$, $\mathbf{G - e}$ ως εξής:
$V(G - v) = V \setminus \{v\}$, $E(G - v) = E \setminus \{e_i \in E : v \in e_i\}$, ενώ
$V(G - e) = V$, $E(G - e) = E \setminus \{e\}$.

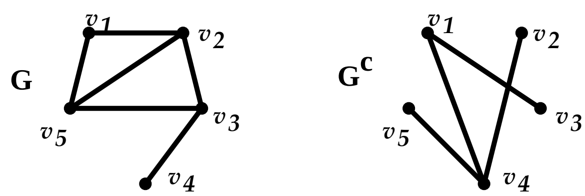
Παραδείγματα:



ΣΥΜΠΛΗΡΩΜΑ

Συμπλήρωμα (complement) G^c (ή $\overline{G}$) του $G = (V, E)$ με $|V| = n$ είναι ένα γράφημα $G^c = (V, E^c)$, με $E^c = E(K_n) \setminus E(G)$.

Παράδειγμα:



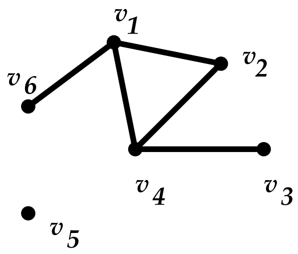
ΒΑΘΜΟΣ

Για κάθε $v \in V$ ορίζουμε $\Gamma_G(v) = \{u \in V(G) : \{v, u\} \in E(G)\}$.

Τότε $d_G(v)$, ή $d(v)$, ή $\deg(v) = |\Gamma_G(v)|$, είναι ο **βαθμός** (degree) του κόμβου v.

Δηλαδή, βαθμός του v στο G, λέγεται το πλήθος των δεσμών του G των οποίων ο v είναι άκρο, ή ισοδύναμα το πλήθος των γειτόνων του v.

Παράδειγμα:



Στο παραπάνω γράφημα, οι κόμβοι του έχουν τους ακόλουθους βαθμούς:

$$\begin{aligned} &d(v_1) = d(v_4) = 3,\\ &d(v_2) = 2,\\ &d(v_3) = d(v_6) = 1,\\ &d(v_5) = 0. \end{aligned}$$

Κάθε κόμβος βαθμού μηδέν λέγεται **μεμονωμένος** κόμβος.

Ένα γράφημα G λέγεται d-**κανονικό** αν $d_G(v) = d$, για κάθε $v \in V$.

Ο **ελάχιστος** (αντ. **μέγιστος**) **βαθμός** των κορυφών ενός γραφήματος G θα συμβολίζεται με $\delta(G)$ (αντ. $\Delta(G)$).

Έστω $G = (V, E)$ με $V = \{v_1, v_2, \ldots, v_n\}$.

Ακολουθία βαθμών του G λέγεται η πεπερασμένη ακολουθία

$$(d(v_1), d(v_2), \ldots, d(v_n)).$$

Παράδειγμα: Η ακολουθία βαθμών του παραπάνω γραφήματος είναι (3,3,2,1,1,0).

Παρατήρηση: Συνήθως γράφουμε την ακολουθία βαθμών ενός γραφήματος σε φθίνουσα σειρά.

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

G = nx.Graph() #Create an empty graph

V = [1,2,3,4,5,6] #V is the set of vertices of G
E = [[1,2],[1,4],[1,6],[2,4],[3,4]] #E is the set of edges of G

G.add_nodes_from(V)
G.add_edges_from(E)

DegreeSeq = []
for v in G.nodes:
    DegreeSeq.append(G.degree(v))
    print("Vertex",v,"has degree",G.degree(v))
DegreeSeq.sort(reverse=True)
print("Degree sequence of G:", DegreeSeq)
```

Output:

```
Vertex 1 has degree 3
Vertex 2 has degree 2
Vertex 3 has degree 1
Vertex 4 has degree 3
Vertex 5 has degree 0
Vertex 6 has degree 1
Degree sequence of G: [3, 3, 2, 1, 1, 0]
```

Οι ακολουθίες βαθμών έχουν ορισμένους περιορισμούς, δηλαδή δεν αντιστοιχούν όλες οι ακολουθίες φυσικών αριθμών σε ακολουθίες βαθμών γραφημάτων. Για παράδειγμα, δεν υπάρχει γράφημα με 5 κορυφές το οποίο έχει ακολουθία βαθμών $(6, 4, 4, 4, 4)$ διότι σε κάθε γράφημα G με 5 κορυφές ισχύει ότι $\Delta(G) \leq 4$.

Μια ακολουθία φυσικών αριθμών $(d_1, d_2, \ldots, d_n)$ ονομάζεται **γραφική** (graphical) αν υπάρχει γράφημα G με ακολουθία βαθμών την ακολουθία $(d_1, d_2, \ldots, d_n)$.

Η μηδενική ακολουθία $(0, 0, \ldots, 0)$ μήκους n αντιστοιχεί στο μηδενικό γράφημα με n κορυφές.

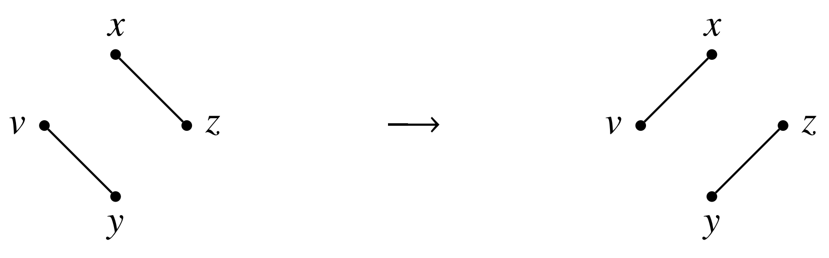
Η επόμενη πρόταση δίνει μια αναγκαία και ικανή συνθήκη για το πότε μια ακολουθία είναι γραφική.

Πρόταση 1 (Θεώρημα Havel - Hakimi).
Η φθίνουσα ακολουθία $(d_1, d_2, \ldots, d_n)$ είναι γραφική αν και μόνο αν η ακολουθία $(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n)$ είναι γραφική.

Απόδειξη. Το αντίστροφο είναι προφανές. Πράγματι, δοθέντος ενός γραφήματος με ακολουθία βαθμών d' προσθέτουμε σε αυτό μια νέα κορυφή v η οποία ενώνεται με τις d_1 κορυφές με τον μεγαλύτερο βαθμό, παίρνοντας έτσι ένα γράφημα με ακολουθία βαθμών d.

Για το ευθύ, θεωρούμε ένα γράφημα $G = (V, E)$ με ακολουθία βαθμών d και μια οποιαδήποτε κορυφή v, έστω βαθμού k. Η απόδειξη βασίζεται στην παρατήρηση ότι μπορούμε να εναλλάξουμε τους δεσμούς του G χωρίς να αλλάξουν οι βαθμοί των κορυφών του, έτσι ώστε η v να συνδέεται με τις (υπόλοιπες) κορυφές που έχουν τους k μεγαλύτερους βαθμούς.

Πράγματι, έστω ότι $x, y \in V$ με $d(x) > d(y)$ και η v έχει γείτονα τον y αλλά όχι τον x.



Επειδή $d(x) > d(y)$ υπάρχει γείτονας z της x που δεν είναι γείτονας της y.

Διαγράφοντας τις ακμές $\{v, y\}$ και $\{x, z\}$ και προσθέτοντας τις ακμές $\{v, x\}$ και $\{y, z\}$ προκύπτει ένα γράφημα G' στο οποίο οι κορυφές διατηρούν τους ίδιους βαθμούς που είχαν στο G και η v συνδέεται με την x.

Επαναλαμβάνοντας αυτόν τον μετασχηματισμό μπορούμε να πετύχουμε την ζητούμενη ιδιότητα.

Τέλος, διαγράφοντας την κορυφή v με τον μέγιστο βαθμό παίρνουμε ένα γράφημα με ακολουθία βαθμών την d'. $\square$

Παράδειγμα : Σύμφωνα με το θεώρημα Havel - Hakimi η ακολουθία $(6, 6, 4, 4, 2, 2, 2, 2)$ είναι γραφική αν και μόνο αν η ακολουθία

$$(6-1, 4-1, 4-1, 2-1, 2-1, 2-1, 2) = (5, 3, 3, 1, 1, 1, 2)$$

είναι γραφική. Η ακολουθία $(5, 3, 3, 2, 1, 1, 1)$ είναι γραφική ανν η ακολουθία

$$(3-1, 3-1, 2-1, 1-1, 1-1, 1) = (2, 2, 1, 0, 0, 1)$$

είναι γραφική. Η ακολουθία $(2, 2, 1, 1, 0, 0)$ είναι γραφική ανν η ακολουθία

$$(2-1, 1-1, 1, 0, 0) = (1, 0, 1, 0, 0)$$

είναι γραφική. Η ακολουθία $(1, 1, 0, 0, 0)$ είναι πράγματι γραφική, αφού αντιστοιχεί στο γράφημα,

άρα και η ακολουθία $(6, 6, 4, 4, 2, 2, 2, 2)$ είναι επίσης γραφική.

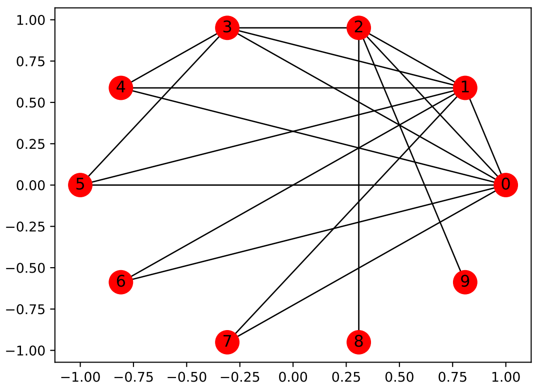
Το θεώρημα Havel - Hakimi δίνει ένα αναδρομικό κριτήριο για τον έλεγχο του κατά πόσο μια ακολουθία είναι γραφική, και ανάγει το πρόβλημα στον έλεγχο μια ακολουθίας με μήκος ένα λιγότερο από την αρχική. Μπορούμε να εφαρμόσουμε ξανά το θεώρημα στην ακολουθία $(d_2-1, d_3-1, \ldots, d_{d_1+1}-1, d_{d_1+2}, d_{d_1+3}, \ldots, d_n)$ (αρκεί πρώτα να την διατάξουμε σε φθίνουσα σειρά) και να προκύψει μια ακολουθία με μικρότερο μήκος, μέχρις ότου να καταλήξουμε σε μία ακολουθία για την οποία μπορούμε εύκολα να συμπεράνουμε αν είναι γραφική ή όχι. Οι ακολουθίες που προκύπτουν κατά την αναδρομική εφαρμογή του θεωρήματος Havel - Hakimi είτε είναι όλες γραφικές είτε καμία γραφική.

Για τον έλεγχο ύπαρξης και κατασκευής ενός γραφήματος με συγκεκριμένη ακολουθία βαθμών μπορούμε να χρησιμοποιήσουμε τις μεθόδους `is_graphical` και `havel_hakimi_graph` της βιβλιοθήκης networkx.

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

def Graph_Check(Seq):
    if nx.is_graphical(Seq):
        print("The sequence",Seq,"is graphical")
        G = nx.havel_hakimi_graph(Seq)
        nx.draw_circular(G,with_labels=True)
        plt.show()
    else:
        print("The sequence",Seq,"is not graphical")
    print("")

Seq1 = [7,7,5,5,3,3,2,2,1,1]
Seq2 = [7,7,7,3,3,2,1,1,1]

Graph_Check(Seq1)
Graph_Check(Seq2)
```

Output:

```
The sequence [7, 7, 5, 5, 3, 3, 2, 2, 1, 1] is graphical
```




```
The sequence [7, 7, 7, 3, 3, 2, 1, 1, 1] is not graphical
```

Ο ΑΛΓΟΡΙΘΜΟΣ ΤΩΝ HAVEL - HAKIMI

Επίσης, το θεώρημα Havel - Hakimi μας προσφέρει μια απλή επαναληπτική μέθοδο για να κατασκευάζουμε γραφήματα με συγκεκριμένη ακολουθία βαθμών.

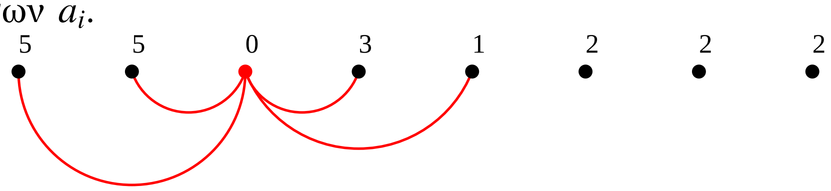
Συγκεκριμένα, αν $(d_1, d_2, \ldots, d_n)$ είναι μια γραφική ακολουθία ο αλγόριθμος κατασκευής χρησιμοποιεί μια βοηθητική ακολουθία $(a_1, a_2, \ldots, a_n)$ και λειτουργεί ως εξής:

Αρχικά θεωρούμε n κορυφές $\{v_1, v_2, \ldots, v_n\}$ βαθμού 0 και για κάθε κορυφή σημειώνουμε τον αριθμό a_i των δεσμών που απαιτούνται για να αποκτήσει βαθμό d_i.

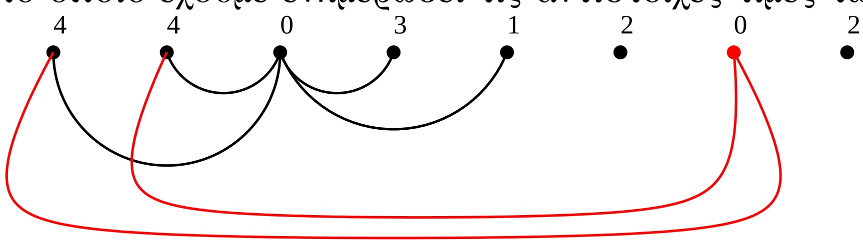
Σε κάθε βήμα επιλέγουμε οποιαδήποτε κορυφή, έστω την v_k, με $a_k > 0$, και την συνδέουμε με a_k σε πλήθος κορυφές που έχουν τις μεγαλύτερες δυνατές θετικές τιμές στην ακολουθία $(a_1, a_2, \ldots, a_n)$. Ενημερώνουμε την ακολουθία $(a_1, a_2, \ldots, a_k)$ και επαναλαμβάνουμε το βήμα αυτό μέχρις ότου όλα τα a_k γίνουν μηδενικά.

Παράδειγμα : Για να κατασκευάσουμε ένα γράφημα με ακολουθία βαθμών $(6, 6, 4, 4, 2, 2, 2, 2)$ αρχικά θεωρούμε 8 κορυφές βαθμού 0. (Στο σχήμα οι κορυφές $v_1, v_2, \ldots, v_8$ αναπαρίστανται με την σειρά από τα αριστερά προς τα δεξιά και σημειώνονται μόνο οι τιμές της ακολουθίας a_i).

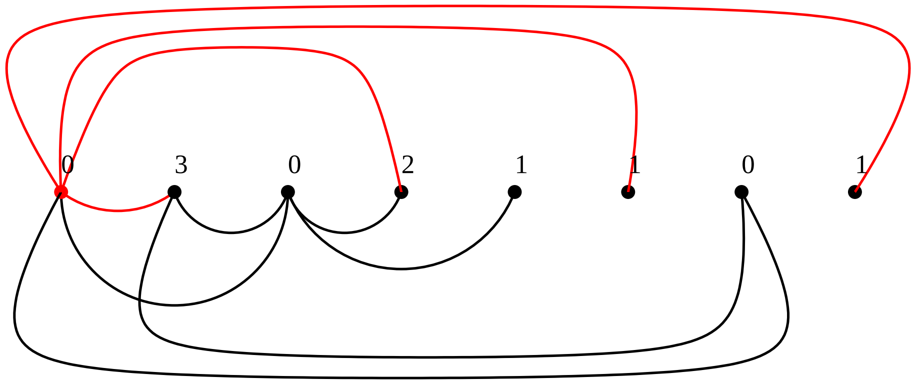
Επιλέγουμε (αυθαίρετα) την κορυφή v_3 (που είναι μαρκαρισμένη με κόκκινο). Η v_3 έχει $a_3 = 4$ οπότε την συνδέουμε με τις 4 κορυφές οι οποίες έχουν τις μεγαλύτερες δυνατές τιμές της ακολουθίας a_i, εδώ είναι οι κορυφές v_1, v_2, v_4 και v_5, οπότε προκύπτει το επόμενο γράφημα στο οποίο έχουμε ενημερώσει τις αντίστοιχες τιμές των a_i.

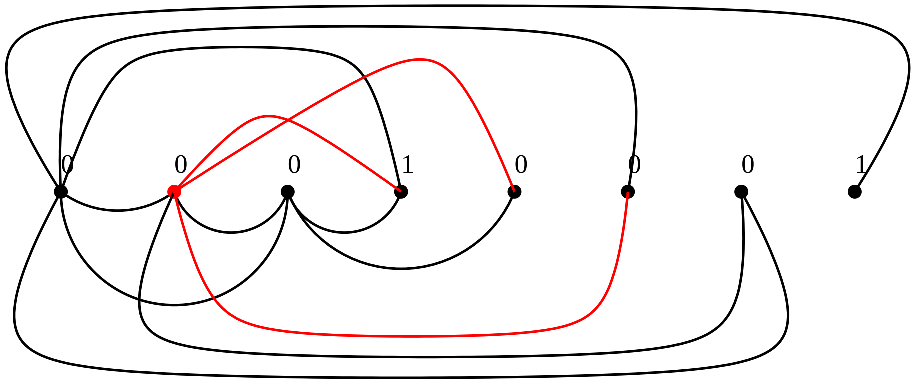


Στην συνέχεια, επιλέγουμε (αυθαίρετα) την κορυφή v_7 για την οποία $a_7 = 2$ και την συνδέουμε με τις 2 κορυφές οι οποίες έχουν τις μεγαλύτερες δυνατές τιμές της ακολουθίας a_i, εδώ είναι οι κορυφές v_1, v_2, οπότε προκύπτει το επόμενο γράφημα στο οποίο έχουμε ενημερώσει τις αντίστοιχες τιμές των a_i.

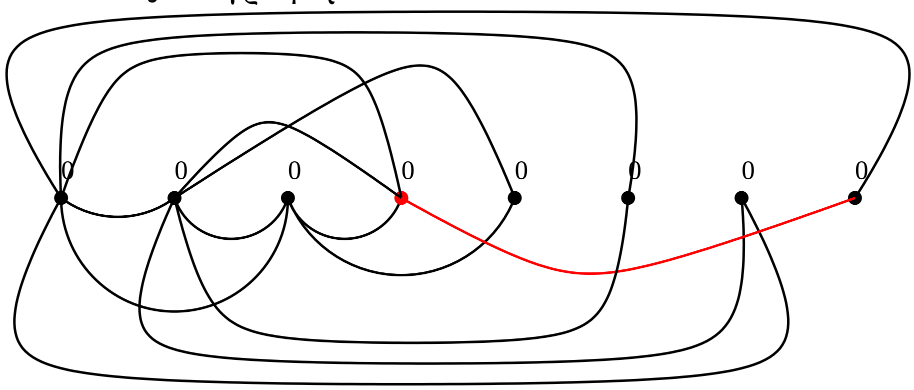


Συνεχίζουμε με τον ίδιο τρόπο, επιλέγοντας κορυφές v_k με $a_k > 0$, οπότε προκύπτουν διαδοχικά τα γραφήματα:





και τέλος το γράφημα

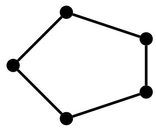
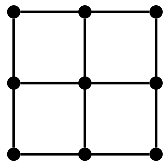


το οποίο έχει ακολουθία βαθμών $(6, 6, 4, 4, 2, 2, 2, 2)$.

Παρατηρήσεις : Στην περίπτωση όπου η ακολουθία $(d_1, d_2, \ldots, d_n)$ δεν είναι γραφική ο αλγόριθμος θα αποτύχει διότι θα υπάρχουν θετικά a_k αλλά δεν θα υπάρχουν αρκετές διαθέσιμες κορυφές για να συνδεθεί η κορυφή v_k.

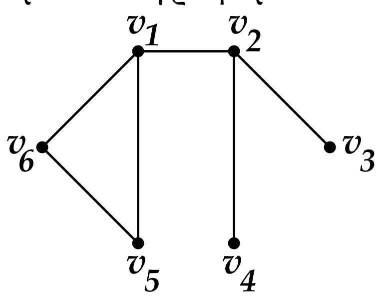
Ασκήσεις προς επίλυση

(1) Χρησιμοποιώντας την βιβλιοθήκη networkx κατασκευάστε τα παρακάτω γραφήματα

C_5  $L_{3,3}$ 

(2) Ένα γράφημα G έχει 20 κορυφές.
   i) Ποιος είναι ο ελάχιστος και ο μέγιστος δυνατός βαθμός των κορυφών του;
   ii) Ποιος είναι ο ελάχιστος και ο μέγιστος δυνατός αριθμός των δεσμών που περιέχει;
   iii) Αν το G έχει 50 δεσμούς, πόσους δεσμούς έχει το συμπλήρωμα του;

(3) Να βρεθεί το συμπλήρωμα των γραφημάτων

G 

$H = K_6$

(4) Να κατασκευασθεί
   i) ένα 2-κανονικό γράφημα με 10 κορυφές.
   ii) ένα 3-κανονικό γράφημα με 10 κορυφές.

(5) Να εξετασθεί, αν υπάρχουν, γραφήματα δεσμών G που έχουν τις παρακάτω ακολουθίες βαθμών.
   i) $(11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0)$
   ii) $(5, 5, 4, 4, 3, 3, 2, 2, 1, 1)$.
   iii) $(5, 5, 3, 3, 3, 1)$
   iv) $(6, 3, 3, 3, 3, 3, 3)$
   v) $(3, 3, 3, 3, 1, 1, 1, 1, 1, 1)$
   vi) $(3, 3, 3, 3, 2, 2, 2, 2, 2)$.
   vii) $(4, 4, 2, 2, 2, 2, 2, 2, 2, 2)$.

(6) Να δειχθεί ότι για κάθε $n \geq 2$ υπάρχει γράφημα δεσμών G με $2n$ κορυφές και ακολουθία βαθμών $(n, n, n-1, n-1, \dots, 3, 3, 2, 2, 1, 1)$.