

Ειδικά Θέματα Θεωρίας και Εφαρμογών Γραφημάτων

Πληροφορίες για το μάθημα

Ιστότοπος μαθήματος

<https://gunet2.cs.unipi.gr/courses/TMH102/>

Διδάσκοντες

- Επ. Καθηγητής Ιωάννης Τασούλας
email: jtas@unipi.gr
- Δρ. Κωνσταντίνος Μανές
email: kmanes@unipi.gr

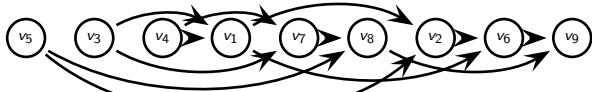
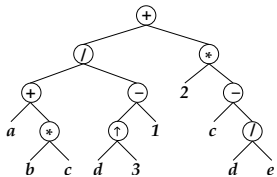
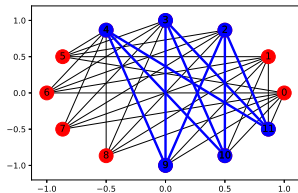
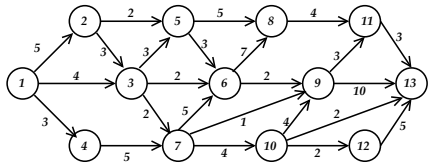
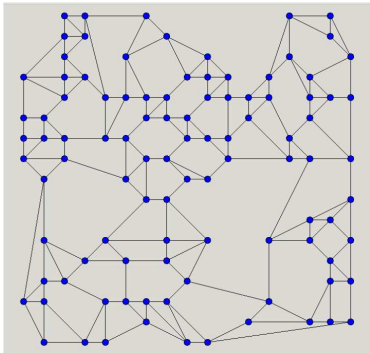
Πρόγραμμα διαλέξεων μαθήματος

Τρίτη 8-10

Περίληψη μαθήματος

Το μάθημα είναι μια εισαγωγή στην θεωρία γραφημάτων και σε μερικές εφαρμογές τους.

Τί είναι τα γραφήματα;



Ειδικά Θέματα Θεωρίας και Εφαρμογών Γραφημάτων

Ύλη μαθήματος

- Δίδονται αναλυτικές σημειώσεις που καλύπτουν πλήρως την ύλη.
- Δεν υπάρχει προαπαιτούμενη ύλη. Αρχίζουμε από την αρχή.
- Σε κάθε διάλεξη παρουσιάζονται εφαρμογές των γραφημάτων (σε Python)

Τρόπος εξέτασης

- Δύο απαλλακτικές πρόοδοι
- Πριν από κάθε πρόοδο θα γίνουν φροντιστηριακά μαθήματα.

Ειδικά Θέματα Θεωρίας και Εφαρμογών Γραφημάτων

Ενδεικτικός προγραμματισμός εξαμήνου

- | | |
|-------------|--|
| 1η διάλεξη | Εισαγωγή. Βασικές έννοιες. |
| 2η διάλεξη | Συνεκτικότητα. |
| 3η διάλεξη | Αποστάσεις. Ισομορφισμός. |
| 4η διάλεξη | Διμερή γραφήματα. Επίπεδα γραφήματα. |
| 5η διάλεξη | Χρωματικός αριθμός - Ανεξαρτησία - Κάλυψη. |
| | 1ο φροντιστήριο |
| 6η διάλεξη | Δυαδικά και διατεταγμένα δένδρα I. |
| 7η διάλεξη | Δυαδικά και διατεταγμένα δένδρα II. |
| 8η διάλεξη | Γραφήματα τόξων. |
| 9η διάλεξη | Τοπολογική διάταξη. Χρονικός προγραμματισμός. |
| 10η διάλεξη | Δυναμικά μοντέλα γραφημάτων. Ευρετικές τεχνικές. |
| | 2ο φροντιστήριο |



<https://networkx.org>

Contact

[Mailing list](#)
[Issue tracker](#)
[Source](#)

Releases

Stable (notes)

[3.1 — April 2023](#)
[Documentation](#)

Latest (notes)

[3.2 development](#)
[Documentation](#)

Archive



NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.



Software for complex networks

- Data structures for graphs, digraphs, and multigraphs
- Many standard graph algorithms
- Network structure and analysis measures
- Generators for classic graphs, random graphs, and synthetic networks
- Nodes can be "anything" (e.g., text, images, XML records)
- Edges can hold arbitrary data (e.g., weights, time-series)
- Open source [3-clause BSD license](#)
- Well tested with over 90% code coverage
- Additional benefits from Python include fast prototyping, easy to teach, and multi-platform

©2014-2023, NetworkX developers.

```

import math

import matplotlib.pyplot as plt
import networkx as nx

# This example needs Graphviz and either PyGraphviz or pydot.
# From networkx.drawing.nx_pydot import graphviz_layout as layout
from networkx.drawing.nx_agraph import graphviz_layout as layout

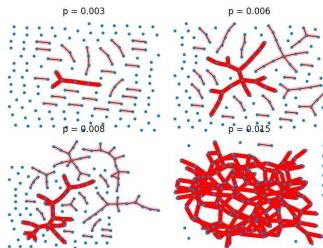
# If you don't have pygraphviz or pydot, you can do this
# layout = nx.spring_layout

n = 150 # 150 nodes
# p value at which giant component (of size log(n) nodes) is expected
p_giant = 1.0 / (n - 1)
# p value at which graph is expected to become completely connected
p_conn = math.log(n) / float(n)

# the following range of p values should be close to the threshold
pvals = [0.003, 0.006, 0.008, 0.015]

region = 220 # for pylab 2x2 subplot layout
plt.subplots_adjust(left=0, right=1, bottom=0, top=0.95, wspace=0.01, hspace=0.01)
for p in pvals:
    G = nx.binomial_graph(n, p)
    pos = layout(G)
    region += 1
    plt.subplot(region)
    plt.title(f"p = {p:.3f}")
    nx.draw(G, pos, with_labels=False, node_size=10)
    # identify largest connected component
    Gcc = sorted(nx.connected_components(G), key=len, reverse=True)
    G0 = G.subgraph(Gcc[0])
    nx.draw_networkx_edges(G0, pos, edge_color="r", width=6.0)
    # show other connected components
    for Gi in Gcc[1:]:
        if len(Gi) > 1:
            nx.draw_networkx_edges(
                G.subgraph(Gi), pos, edge_color="r", alpha=0.3, width=5.0,
            )
plt.show()

```



```

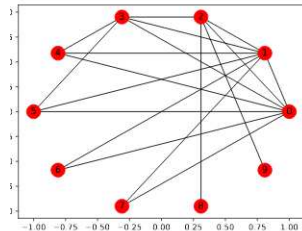
import networkx as nx
import matplotlib.pyplot as plt

def Graph_Check(Seq):
    if nx.is_graphical(Seq):
        print("The sequence",Seq,"is graphical")
        G = nx.havel_hakimi_graph(Seq)
        print("The adjacency matrix of graph having this degree sequence
is:")
        print(nx.adjacency_matrix(G).todense())
        pos = nx.circular_layout(G)
        nx.draw_networkx(G,pos)
        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)

```

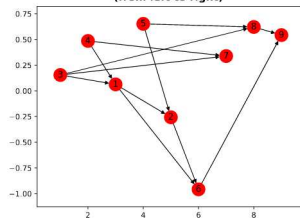


```

# A simple implementation of topological sorting algorithm
L = [] #Topological sorting of the nodes of G
H = G.copy() #Work on a copy of G
while(len(H)!=0):
    notfound = True
    for v in H.nodes():
        if (H.in_degree(v) == 0):
            L.append(v)
            H.remove_node(v)
            notfound = False # vertex v has indegree 0
            break
    if notfound: break # no vertex has indegree 0
# If L contains all the nodes of G a solution is found
if len(L) == len(G):
    print("A topological sorting of G:",L)
else:
    print("G contains a cycle")

```

Topological sorting of the vertices of G
(from left to right)



Μαθησιακά αποτελέσματα

Με την επιτυχή ολοκλήρωση του μαθήματος οι φοιτητές αναμένεται ότι:

- θα γνωρίζουν τους βασικούς ορισμούς και τα βασικά αποτελέσματα που αφορούν τα γραφήματα δεσμών και τα γραφήματα τόξων.
- θα γνωρίζουν σημαντικές εφαρμογές των γραφημάτων.
- θα γνωρίζουν πώς να υλοποιήσουν αλγορίθμους γραφημάτων στην γλώσσα Python χρησιμοποιώντας την βιβλιοθήκη `networkx`.