ΟΜΑΔΙΚΗ ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ ΕΙΣΑΓΩΓΗ ΣΤΟΥΣ ΥΠΟΛΟΓΙΣΤΕΣ

ΘΕΜΑ: ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΙΚΟΝΗΣΗ DEAPTH FIRST SEARCH ΑΛΓΟΡΙΘΜΟΥ ΜΕ ΧΡΗΣΗ ΒΙΒΛΙΟΘΗΚΗΣ NETWORKX ΚΑΙ MATPLOTLIB

ΕΠΙΜΕΛΕΙΑ: ΑΡΙΣ ΑΙΒΑΖΙΑΝ, ΕΜΜΑΝΟΥΗΛ ΛΕΓΑΚΗΣ, ΑΓΓΕΛΟΣ ΚΟΝΤΑΛΗΣ, ΝΙΚΟΛΑΣ ΓΙΑΝΝΗ, ΓΙΩΡΓΟΣ ΓΕΩΡΓΙΟΠΟΥΛΟΣ, ΓΙΩΡΓΟΣ ΓΚΟΝΤΕΒΑΣ.

ΤΙ ΚΑΝΑΜΕ

Αρχικά για την εκπόνηση της εργασίας μας κάναμε μια αναλυτική έρευνα για το τι ακριβώς είναι ο αλγόριθμος DFS (ή αλλιώς αλγόριθμος αναζήτησης κατά βάθος), για το πως λειτουργεί και ποιες είναι οι πρακτικές εφαρμογές του. Συγκεκριμένα, ο αλγόριθμος Αναζήτησης Κατά Βάθος (DFS - Depth-first search) επιτυγχάνει διάσχιση ή αναζήτηση σε δέντρο ή γράφημα και συνεχίζει σε βάθος του δέντρου μέχρι να φτάσει σε κόμβο. Η διάσχιση ξεκινά από τη ρίζα και εξερευνά όσο το δυνατόν περισσότερο κατά βάθος του δέντρου μέχρι να βρεθεί ο ζητούμενος κόμβος ή να καταλήξεις σε αδιέξοδο, δηλαδή σε σημείο χωρίς γειτονικούς κόμβους. Αποτελεσματικά, συνειδητοποιήσαμε ότι ο αλγόριθμος έχει πολλαπλές χρήσεις:

- Επίλυση παζλ με μια μόνο λύση (όπως λαβύρινθοι)
- Δημιουργία λαβυρίνθου
- Τοπολογική ταξινόμηση
- Εύρεση γεφυρών ενός γραφήματος
- Εύρεση συνεκτικών συνιστωσών

Πιο συγκεκριμένα, εστιάσαμε την ερευνά μας στην τοπολογική ταξινόμηση, η οποία αποτελεί και το αντικείμενο του προγράμματος μας. Ειδικότερα, τοπολογική ταξινόμηση ή αλλιώς τοπολογική διάταξη ενός Graph ονομάζεται στη Θεωρία Γράφων η γραμμική διάταξη των κόμβων, έτσι ώστε κάθε πρόγονος ενός κόμβου v προηγείται του v στη διάταξη.

ΠΩΣ ΤΟ ΚΑΝΑΜΕ

Αρχικά, δημιουργήσαμε μια ομάδα στο discord προκειμένου να έχουμε άμεση πρόσβαση σε υπολογιστή και τον κώδικα. Εφόσον συζητήσαμε για την βασική δομή του κώδικα και παρακολουθήσαμε ορισμένα βίντεο από επαγγελματίες προγραμματιστές στο YouTube καταλήξαμε στην ιδέα ότι ο αλγόριθμος πρέπει πρώτα να δημιουργεί μια κενή λίστα στην οποία αποθηκεύει τα nodes του γραφήματος αφότου πρώτα τα έχει επισκεφθεί γραμμικά. Επίσης, χρειάστηκε η εγκατάσταση των βιβλιοθηκών matplotlib και network Έπειτα μια άλλη συνάρτηση θα διαβάζει τη λίστα και θα χρωματίζει το αναλόγως το γράφημα.

Η πρώτη μας υλοποίηση, παρόλο που είχε το προτέρημα να απεικονίζει πιο αναλυτικά τον αλγόριθμο DFS, δηλαδή παρουσίαζε ακριβώς τη διαδρομή που ακολουθούσε ο αλγόριθμος, είχε τα εξής μειονεκτήματα:

1. Ήταν μεγαλύτερο σε έκταση (διακόσιες περίπου παραπάνω γραμμές)
2. Δεν ήταν αυτοματοποιημένη η διαδικασία της σχεδίασης τους γραφήματος. Συγκεκριμένα, εμείς οι ίδιοι, αντιλαμβανόμενοι της διαδρομής που ακολουθούσε ο αλγόριθμος, δίναμε την εντολή να χρωματίζει τον κάθε κόμβο ξεχωριστά με αποτέλεσμα οποιαδήποτε μελλοντική αλλαγή να απαιτεί και την ανάλογη προσθήκη στον κώδικα για τον σχεδιασμό.
3. Ήταν δυσανάγνωστος και απαιτούσε εκτενή διόρθωση και τροποποίηση.

Επομένως, κρατώντας τη βασική ιδέα της πρώτης μας προσπάθειας, επιδιώξαμε να διορθώσουμε τα προαναφερόμενα μειονεκτήματα. Έτσι, καταλήξαμε στον επόμενο μας κώδικα.

Αναλυτικότερα, ο κώδικας χωρίζονταν σε τρία σκέλη, στη δημιουργία του γραφήματος και στις δυο βασικές συναρτήσεις :

Πρώτο σκέλος: Με τη βιβλιοθήκη network δημιουργήσαμε το graph, τα nodes, τις ενώσεις τους και τις θέσεις του στο γράφημα με τη χρήση συντεταγμένων και με κέντρο αναφοράς το «Α»(θέση (0,0)).

Δεύτερο σκέλος: Η πρώτη συνάρτηση αποτελούσε τον ίδιο τον αλγόριθμο Depth First Search. Αρχικά , δημιουργούσε μια κενή λίστα (visited_node) μέσα στην οποία αποθηκεύονταν οι επισκεπτόμενοι κομβόι κατά την εκτέλεση της συνάρτησης. Εφόσον ο αλγόριθμος επισκέπτονταν έναν κόμβο και τον προσθέσει στην λίστα, αναζητά τους γειτονικούς του και ξενικά η ιδιά διαδικασία από την αρχή. Όλα αυτά γίνονταν με την χρήση των εντολών if και for.

Τρίτο σκέλος: Η δεύτερη συνάρτηση εκτελούσε τον σχεδιασμό του γραφήματος και απεικόνιζε την διαδρομή που ακολουθεί ο αλγόριθμος. Ο τίτλος του plot (παραθύρου) έδειχνε τη σειρά με την οποία τα nodes προστίθενται στην λίστα που είχε δημιουργηθεί προηγουμένως. Έπειτα δημιουργούσε το βασικό σκελετό του γραφήματος, δηλαδή σχεδίαζε τους κόμβους και τις ενώσεις τους. Αρχικά, με τη χρήση for loop ξεκινούσε και διάβαζε τη λίστα. Εφόσον διάβαζε κάθε στοιχείο της λίστας, το χρωματίζει κόκκινο, υποδεικνύοντας ότι ο αλγόριθμος εκείνη τη στιγμή βρισκόταν και διάβαζε το συγκεκριμένο κόμβο. Στη συνέχεια, ο ίδιος κόμβος χρωματιζόταν πράσινος, δείχνοντας ότι ο αλγόριθμος είχε πλέον περάσει και είχε διαβάσει τον επόμενο γειτονικό κόμβο. Η διαδικασία αυτή εκτελούταν μέχρι ώσπου βρεθεί ο τελευταίος κόμβος και εν τέλη όλοι οι κομβόι είναι χρωματισμένοι πράσινοι.

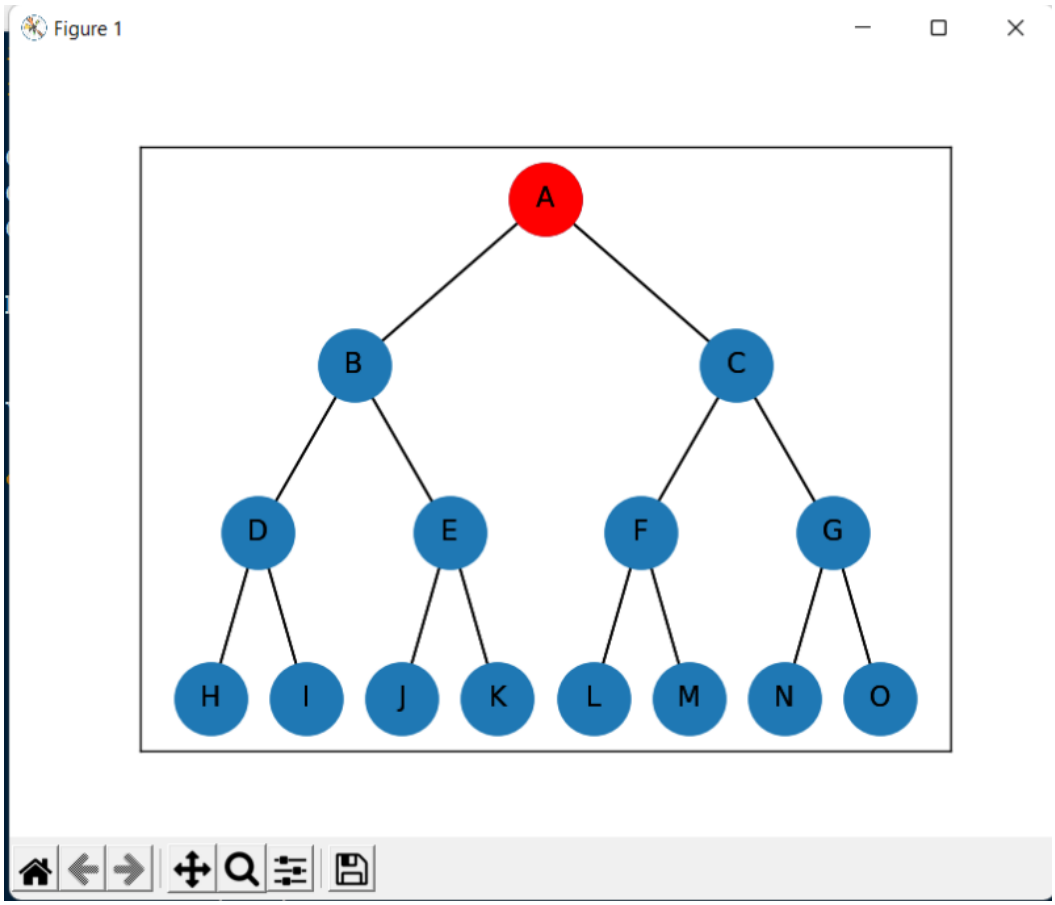
Τελικά, έπειτα από και άλλες τροποποιήσεις καταλήξαμε στο τελικό μας και πιο συμπυκνωμένο κώδικα ο οποίος αποτελείται από μόνο μια συνάρτηση. Ουσιαστικά, συνδικάσαμε τις δυο συναρτήσεις σε μια έτσι ώστε με το που ο αλγόριθμος προσθέσει ένα νέο κόμβο στη

λίστα να το ζωγραφίζει επιτόπου, προκειμένου να γίνει πιο κατανοητή και ακριβής η λογική του αλγορίθμου.

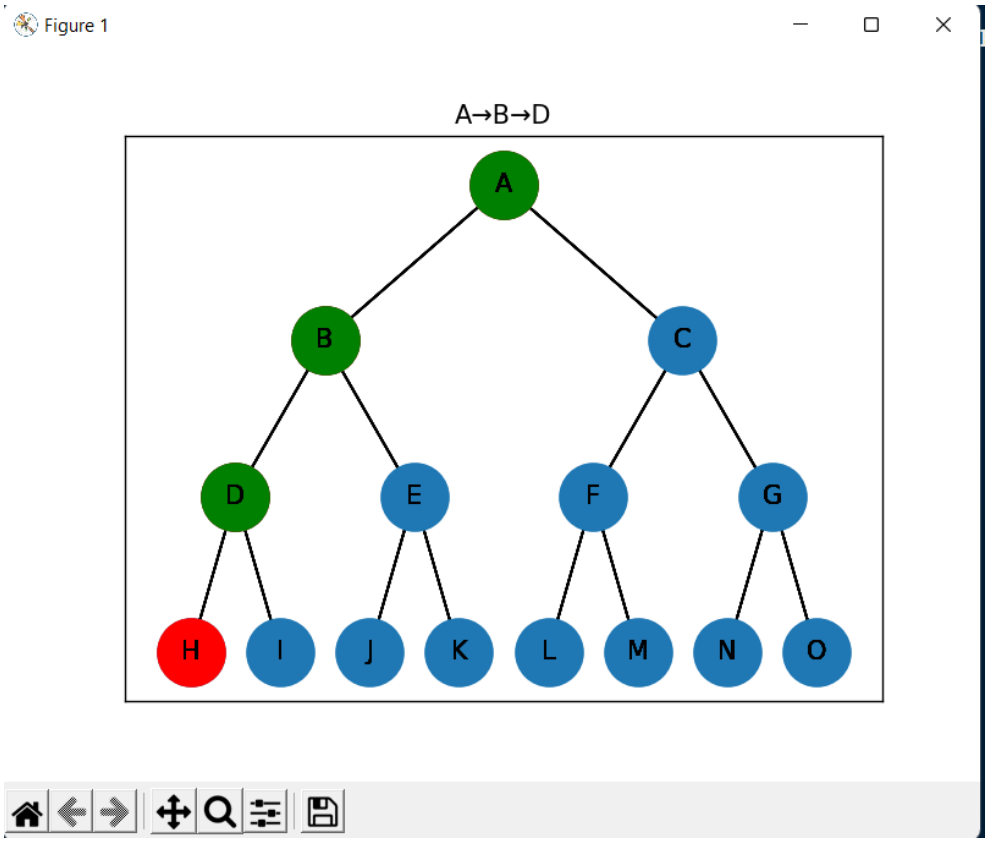
ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Με αυτόν τον τρόπο δημιουργήσαμε ένα αυτοματοποιημένο ευανάγνωστο πρόγραμμα που εντοπίζει και διαβάζει ένα γράφημα και το σχεδιάζει μόνο του. Το πρόγραμμα ουσιαστικά είναι ικανό να διαβάσει οποιοδήποτε γράφημα και να επιστρέψει την πληροφορία που είναι αποθηκευμένη σε κάθε κόμβο.

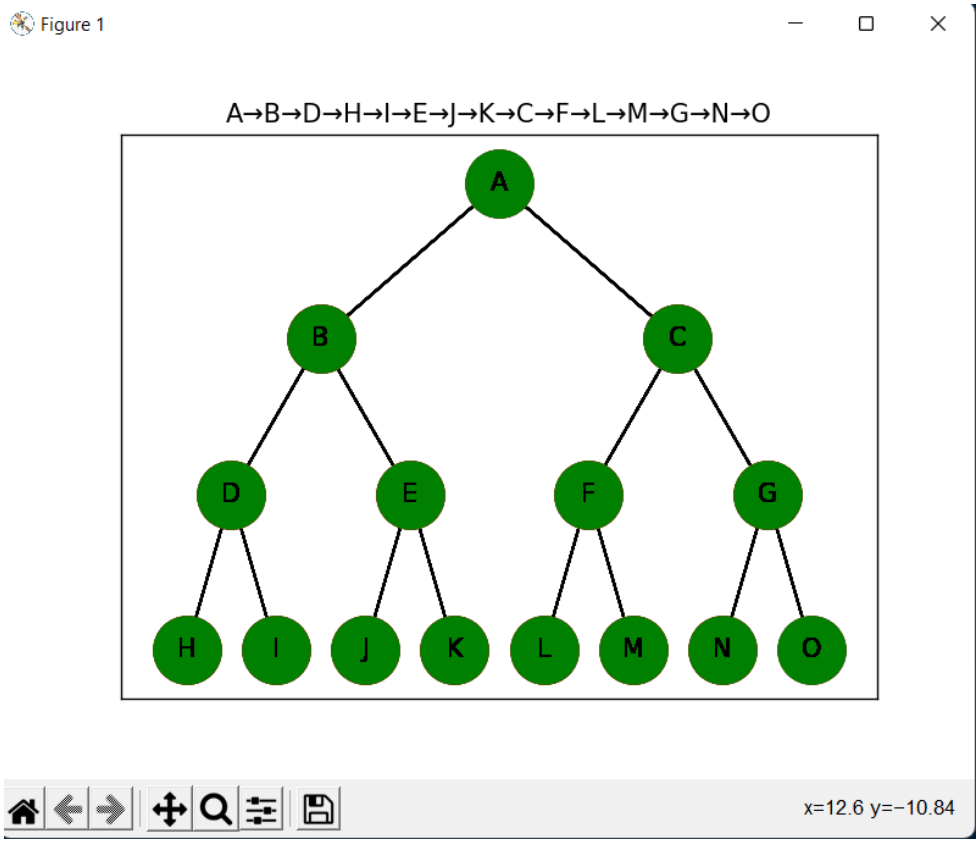
Στην αρχη:



Κατά την εκτέλεση(πράσινο το έχει επισκεφθεί και στο κόκκινο βρίσκεται τώρα):



Η τελική εικόνα:



ΒΙΒΛΙΟΓΡΑΦΊΑ

https://el.wikipedia.org/wiki/%CE%91%CE%BD%CE%B1%CE%B6%CE%AE%CF%84%CE%B7%

CF%83%CE%B7_%CE%9A%CE%B1%CF%84%CE%AC_%CE%92%CE%AC%CE%B8%CE%BF%CF

%82 https://www.programiz.com/dsa/graph-dfs

https://www.youtube.com/watch?v=Fn27mVVF9_s

https://www.youtube.com/watch?v=7fujbpJ0LB4

https://networkx.org/documentation/stable/auto_examples/basic/plot_simple_graph.html

?highlight=networkx%20draw_networkx%20graph

https://networkx.org/documentation/latest/tutorial.html

https://www.youtube.com/watch?v=CPQeSmDGiOQ&ab_channel=MohammadT.Irfan

https://youtu.be/SiXjTkGFwng

https://www.geeksforgeeks.org/using-matplotlib-for-animations/

https://www.youtube.com/watch?v=dOKHY_PUvqU