Minimum Spanning Tree

Μάθημα: Εισαγωγή στους Υπολογιστές, Α’ Εξάμηνο

Πανεπιστήμιο Πατρών, Τμήμα ΗΜΤΥ, 10/1/2021

Επιβλέπων καθηγητής: κ. Χρήστος Βαλουξής

Ομάδα 28:

Γιώργος Βλησίδης (Α.Μ. 1083671)

Γιώργος Γεωργακόπουλος (Α.Μ. 1083695)

Αναστάσης Κελεσίδης (Α.Μ. 1084039)

Πάνος Λελάκης (Α.Μ. 1083712)

Στρατής Σκαπινάκης (Α.Μ. 1083850)

Ευάγγελος Τσιατσιάνας (Α.Μ. 1072369)

Περιεχόμενα

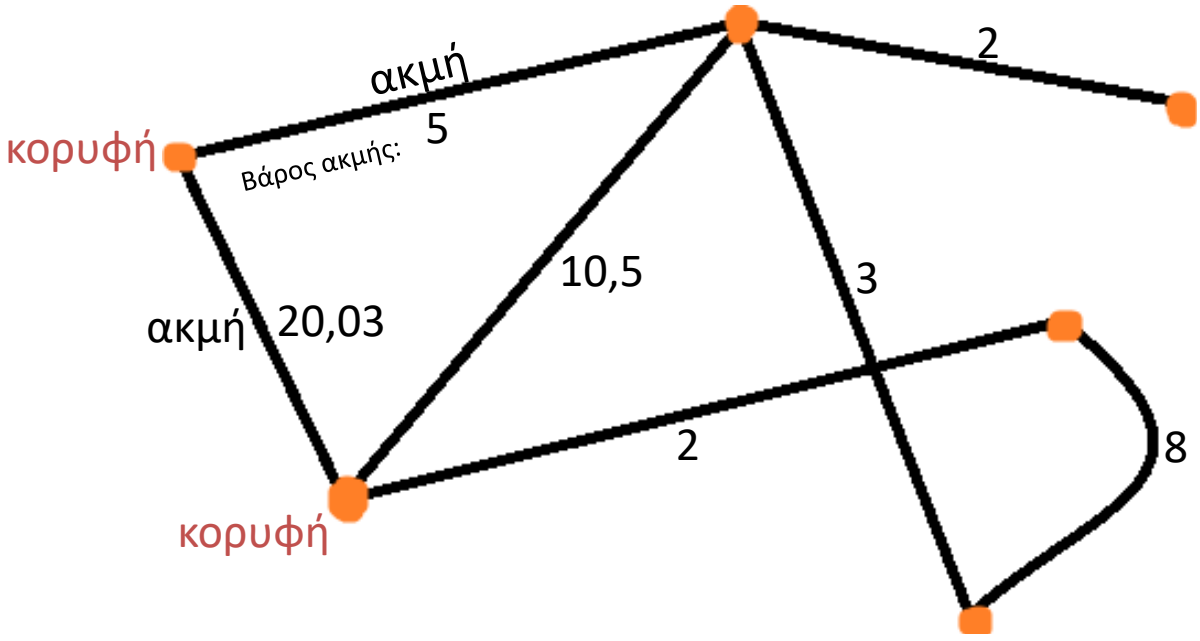
Εισαγωγή

Στα πλαίσια της ομαδικής μας εργασίας στο μάθημα Εισαγωγή στους Υπολογιστές στο Ά Εξάμηνο, ασχοληθήκαμε με την έννοια του Ελάχιστου Διασυνδετικού Δέντρου (Minimum Spanning Tree). Συγκεκριμένα, σχεδιάσαμε ένα πλήρες πρόγραμμα στην γλώσσα Python, που το υπολογίζει με δύο από τους γνωστότερους για το πρόβλημα αυτό αλγόριθμους: του Prim και του Kruskal.
Η εργασία μας αποτελείται από το πρόγραμμα (αρχείο με κατάληξη .py που βρίσκεται στο συμπιεσμένο αρχείο “code28.zip”), τα βοηθητικά αρχεία του προγράμματος (αρχεία με ονομασίες “Test ...” και με κατάληξη .txt, που βρίσκονται στο συμπιεσμένο αρχείο), την έκθεση (αρχείο με ονομασία “report” και κατάληξη .pdf), το αρχείο με τις οδηγίες εγκατάστασης (αρχείο με ονομασία “readme” και με κατάληξη .txt) και την παρουσίαση της εργασίας μας (αρχείο με ονομασία “presentation” και με κατάληξη .pdf). Ξεκινήσαμε την εργασία μας στις 30 Οκτωβρίου 2020 και την παραδώσαμε στις 10 Ιανουαρίου 2021.

Στο πρόβλημα υπολογισμού του MST, το ζητούμενο είναι να βρεθεί το δέντρο που:

1. περιέχει όλες τις κορυφές από ένα δεδομένο σύνολο κορυφών και
2. έχει το ελάχιστο συνολικό βάρος.

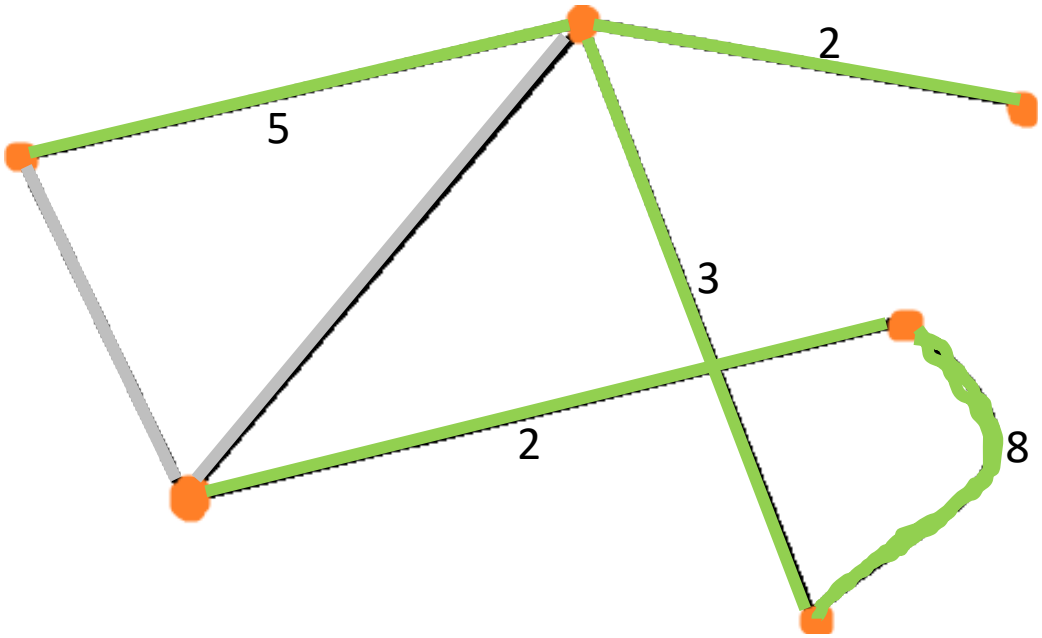
Παρακάτω δίνεται ένα παράδειγμα:



Παρακάτω εξηγούνται οι έννοιες που χρησιμοποιήθηκαν κατά τον ορισμό του προβλήματος.

- Το συνολικό βάρος του MST ισούται με το άθροισμα των βαρών όλων των ακμών στο MST.
- Λέμε ότι ένα γράφημα έχει κατεύθυνση όταν οι ακμές του έχουν κατεύθυνση, δηλαδή είναι διανύσματα.
  Τα γραφήματα με τα οποία ασχολούμαστε δεν έχουν κατεύθυνση, όπως φαίνεται στο παράδειγμα.
- Επίσης, για να μπορούμε να ορίσουμε την λύση του προβλήματος, το γράφημα πρέπει να είναι συνδεδεμένο, δηλαδή κάθε μία κορυφή του να συνδέεται με τις υπόλοιπες κορυφές μέσω κάποιας διαδρομής.
  Το γράφημα στο παράδειγμα είναι συνδεδεμένο.
- Δέντρο ονομάζεται ένα συνδεδεμένο γράφημα που δεν έχει κυκλικές διαδρομές.
  Το γράφημα στο παράδειγμα δεν είναι δέντρο, γιατί περιέχει κυκλικές διαδρομές (πχ. Το τρίγωνο που έχει ακμές με βάρη 5, 10.5 και 20.03, ή η κυκλική διαδρομή που έχει ακμές με βάρη 10.5, 2, 3 και 8)

Παρατίθεται η λύση (MST: πράσινο δέντρο) του παραδείγματος:
Το συνολικό βάρος είναι 20.



Ενότητα 1η: Τρόπος Οργάνωσης

Κατά την ανάπτυξη του κώδικα χρησιμοποιήσαμε συγκεκριμένα εργαλεία για να οργανώσουμε την εργασία μας. Αναλυτικά χρησιμοποιήσαμε:

- Το Discord για να επικοινωνούμε (λόγω των υγειονομικών συνθηκών αναγκαστήκαμε να εργαστούμε εξ αποστάσεως),
- Το GitHub για να μοιραζόμαστε τον κώδικα, να οργανώνουμε την δουλεία των υποομάδων στις οποίες χωριστήκαμε (αναφέρονται αναλυτικά παρακάτω) σε φακέλους προσβάσιμους από όλα τα μέλη και για να έχουμε online backup,
- Το Idle και το Visual Studio Code για το γράψιμο του κώδικα,
- Το OneNote για να κρατάμε σημειώσεις.

Πριν αρχίσουμε την εργασία χωριστήκαμε σε τρεις ομάδες των δύο ατόμων ώστε να μπορούμε να δουλέψουμε πιο αποτελεσματικά.
Κάθε Σαββατοκύριακο κανονίζαμε μία συνάντηση όλη η ομάδα για να ελέγξουμε και να ενώσουμε τις εργασίες που είχε κάνει κάθε υποομάδα, αλλά και για να μοιραστούν οι εργασίες της επόμενης εβδομάδας.
Χωρίσαμε το πρόβλημα σε ενότητες, και την κάθε ενότητα σε ακόμα μικρότερα προβλήματα. Κάθε υποομάδα αναλάμβανε από ένα μικροπρόβλημα.
Συγκεκριμένα οι ενότητες στις οποίες χωρίσαμε το πρόβλημα είναι:

- Οι διάφοροι τρόποι εισαγωγής δεδομένων στο πρόγραμμα,
- Ο έλεγχος και η επεξεργασία των δεδομένων αυτών,
- Η υλοποίηση των αλγορίθμων του Prim και του Kruskal στην Python για τον υπολογισμό του MST,
- Η οργάνωση του προγράμματος σε κλάσεις,
- Ο σχεδιασμός του γραφικού περιβάλλοντος του προγράμματος με τη βοήθεια της βιβλιοθήκης Tkinter.

Ανά τακτά χρονικά διαστήματα επικοινωνούσαμε με τον κ. Χρήστο Βαλουξή, ώστε να τον ενημερώσουμε για την εξέλιξη του προγράμματός μας, να του κάνουμε ερωτήσεις και να μας δώσει συμβουλές, με τις οποίες βελτιώναμε τον κώδικά μας.

Επιπλέον, την δεύτερη εβδομάδα της εργασίας μας οργανώσαμε μία συνάντηση με τον κ. Σωτήρη Χαρίτο, ο οποίος εργάζεται ως Head of Software

Development στον ΟΠΑΠ, και ο οποίος μας έδωσε πολύτιμες συμβουλές για τη διαχείριση και την οργάνωση του project μας.

Ενότητα 2η: Ρόλος Μελών

Για την εργασία μας χωριστήκαμε σε τρεις ομάδες των δύο. Παρακάτω αναφέρονται λεπτομερώς τα μέρη του κώδικα με τα οποία ασχολήθηκε η κάθε υποομάδα.

Ομάδα 1:
Αποτελείται από τους: Στρατής Σκαπινάκης, Ευάγγελος Τσιατσιάνας.
Ασχολήθηκαν με:

- Την εισαγωγή δεδομένων από έτοιμα αρχεία (Test) και με τη μέθοδο “input_file”
- Τον έλεγχο αρνητικού βάρους
- Τη μέθοδο “add_text”
- Τη μέθοδο “ioframe”

Ομάδα 2:
Αποτελείται από τους: Γιώργος Βλησίδης, Γιώργος Γεωργακόπουλος.
Ασχολήθηκαν με:

- Την κατανομή των δεδομένων στις κατάλληλες λίστες και στα λεξικά
- Τον έλεγχο για γράμματα στην εισαγωγή δεδομένων
- Τη δημιουργία τυχαίου γραφήματος και με τη μέθοδο “input_random”
- Τη μέθοδο “info_window”

Ομάδα 3:
Αποτελείται από τους: Αναστάσης Κελεσίδης, Πάνος Λελάκης.
Ασχολήθηκαν με:

- Τη δημιουργία γραφήματος μέσω πληκτρολόγησης (εισαγωγή δεδομένων) και με τη μέθοδο “input_manually”
- Τη μέθοδο “weight_check”
- Τη μέθοδο “solvable_check”
- Τη μέθοδο “credits_window”

Η εμφάνιση του γραφικού περιβάλλοντος, η μέθοδος για τη δημιουργία των κουμπιών (“buttons”) καθώς και οι μέθοδοι για την απεικόνιση του γραφήματος (“mst”, “draw_graph”, “clear_graph”), δημιουργήθηκαν από όλη την ομάδα μας κατά την διάρκεια των τακτικών ομαδικών συναντήσεων.

Για την υλοποίηση των αλγορίθμων Prim και Kruskal, χωριστήκαμε σε 2 υποομάδες:

- Αλγόριθμος Kruskal: Γιώργος Βλησίδης, Γιώργος Γεωργακόπουλος και Ευάγγελος Τσιατσιάνας
- Αλγόριθμος Prim: Αναστάσης Κελεσίδης, Πάνος Λελάκης και Στρατής Σκαπινάκης.

Υπεύθυνος για τη διαμόρφωση του εβδομαδιαίου πλάνου ήταν ο Πάνος Λελάκης ενώ η κατανομή των εργασιών γινόταν με συνεννόηση όλης της ομάδας.

Για την συγγραφή της έκθεσης κάθε μέλος της ομάδας ασχολήθηκε με διαφορετική ενότητα. Συγκεκριμένα:

- Εισαγωγή – Περιεχόμενα: Πάνος Λελάκης
- Τρόπος οργάνωσης : Στρατής Σκαπινάκης
- Ρόλος μελών: Αναστάσης Κελεσίδης
- Οδηγίες χρήσης προγράμματος : Γιώργος Γεωργακόπουλος
- Παραδείγματα χρήσης: Γιώργος Βλησίδης
- Παράρτημα κώδικα : Ευάγγελος Τσιατσιάνας

Για τη δημιουργία της παρουσίασης καθώς και για το αρχείο με τις οδηγίες εγκατάστασης η ομάδα δούλεψε συγκεντρωτικά.

Ενότητα 3η: Παραδείγματα Χρήσης

Η εύρεση του MST είναι ένα θεμελιώδες πρόβλημα με ποικίλες εφαρμογές στον σύγχρονο κόσμο. Η βασικότερη λειτουργία του εντοπίζεται στον σχεδιασμό δικτύων (network design). Επιπλέον, χρησιμοποιούνται στην εύρεση προσεγγιστικών λύσεων μαθηματικών πρακτικών προβλημάτων (πρόβλημα του ταξιδιώτη, πρόβλημα αθροίσματος υποσυνόλου κ.α.).

1. <u>Σχεδιασμός τηλεφωνικών, ηλεκτρικών, υδραυλικών, οδικών, καλωδιακών δικτύων</u>

Η απλή εφαρμογή παρατηρείται σε ένα πρόβλημα όπως ο σχεδιασμός τηλεπικοινωνιακών δικτύων. Ο στόχος μιας εταιρίας τηλεπικοινωνιών είναι να σχεδιάσει ένα δίκτυο που θα συνδέει κάθε πελάτη της (το σπίτι του πελάτη δηλαδή) με το κοντινότερο στον πελάτη κέντρο, δαπανώντας όσο λιγότερα χρήματα γίνεται. Στην συγκεκριμένη περίπτωση κάθε σπίτι αναπαρίσταται από μία κορυφή, κάθε δρόμος ή κανάλι με καλώδια από μία ακμή, και το βάρος κάθε ακμής από το μήκος του δρόμου και από την ποιότητα των καλωδίων. Το καλώδιο μπορεί να περνάει από κοινόχρηστα κανάλια, επομένως δημιουργείται ένα γράφημα, του οποίου το MST είναι η λύση του προβλήματος.

Στο κτήριο μίας επιχείρησης, επίσης, δουλεύουν πολλοί υπάλληλοι, οι οποίοι για να επικοινωνούν μεταξύ τους χρειάζονται ένα δίκτυο. Ο στόχος, στην συγκεκριμένη περίπτωση, είναι να σχεδιαστεί μια σειρά γραμμών που συνδέει όλα τα γραφεία μεταξύ τους με το ελάχιστο πιθανό κόστος.

Μπορούμε να επεκτείνουμε το παραπάνω παράδειγμα αν θέλουμε να κατασκευάσουμε ένα δίκτυο υπολογιστών, οι οποίοι συνδέονται μέσω ενός δικτύου οπτικών ινών. Στα δίκτυα υπολογιστών, πχ. το είδος εκπομπής (broadcast) αναφέρεται στη αποστολή ενός μηνύματος - πακέτου σε όλους τους δέκτες που ανήκουν στο υποδίκτυο. Στο δίκτυο Ethernet το Broadcasting επιτυγχάνεται μέσω του πρωτοκόλλου του MST.

2. Πρόβλημα ταξιδιώτη με αξιοθέατα

Το πρόβλημα εύρεσης του MST μπορεί να χρησιμοποιηθεί και για την προσεγγιστική λύση του προβλήματος του ταξιδιώτη. Αποτελεί έναν πρακτικό και επίσημο τρόπο να υπολογίζεται το συντομότερο δρομολόγιο με το οποίο ο ταξιδιώτης επισκέπτεται κάθε σημείο (περιοχή, αξιοθέατο) μόνο μία φορά. Ένα αξιοθέατο αναπαρίσταται από μία κορυφή, η διαδρομή μεταξύ δύο αξιοθέατων από μία ακμή και το μήκος και η δυσκολία της διαδρομής από το βάρος της ακμής.

3. Σχεδίαση σιδηροδρομικών και οδικών δικτύων

Στόχος είναι να επιτευχθεί η σύνδεση όσο περισσότερων πόλεων-περιοχών με το χαμηλότερο κόστος. Υποθέτουμε ότι υπάρχει μια περιφέρεια με V πόλεις οι οποίες θα συνδεθούν κατασκευάζοντας Χ αυτοκινητόδρομους για τους οποίους αναλογίζεται ένα σχετικό κόστος κατασκευής. Βελτιστοποιούμε το οδικό δίκτυο των πόλεων με τέτοιο τρόπο ώστε να μπορούμε να φτάσουμε σε μια οποιαδήποτε πόλη ενώ βρισκόμαστε σε οποιαδήποτε άλλη, με το τελικό κόστος κατασκευής των αυτοκινητόδρομων να είναι το ελάχιστο. Έτσι, υπάρχει ένας δρόμος μεταξύ οποιωνδήποτε δύο κορυφών (πόλεων) και έτσι κάθε πόλη είναι συνδεδεμένη με το δίκτυο.

4. Εταιρίες παροχής μεταφορικών υπηρεσιών (Ταχυδρόμοι, courier κλπ.)

Στόχος της εταιρίας είναι να σχεδιάσει μία διαδρομή που να περνάει από κάθε σημείο (σπίτι) διανομής, διαδρομή τέτοια ώστε ο μεταφορέας να διανύσει τη χαμηλότερη δυνατή συνολική απόσταση. Ένα σπίτι αναπαρίσταται από μία κορυφή και ένας δρόμος από μία ακμή. Το βάρος της κάθε ακμής, επομένως, εξαρτάται από το μήκος του δρόμου και από την κίνηση στο δρόμο αυτό. Ωστόσο, η αντιμετώπιση του συγκεκριμένου προβλήματος με την εύρεση του MST δεν είναι ρεαλιστική, διότι υποθέτουμε ότι όλοι οι δρόμοι είναι διπλής κυκλοφορίας, κάτι που δεν είναι ρεαλιστικό. Για να λυθεί αυτή η λεπτομέρεια απαιτείται η εύρεση του «κατευθυνόμενου» MST, στην οποία περίπτωση κάθε ακμή έχει κατεύθυνση, όπως αυτή του δρόμου.

Ενότητα 4η: Οδηγίες Εγκατάστασης

1) Εγκατάσταση Python

Απαραίτητη προϋπόθεση ώστε να μπορέσει να εκτελεστεί το πρόγραμμα είναι ο χρήστης να έχει εγκαταστήσει οποιαδήποτε έκδοση (μορφής 3.x.y) της γλωσσάς προγραμματισμού Python στον υπολογιστή του.
Αυτό μπορεί να γίνει εύκολα μέσω της ιστοσελίδας https://www.python.org/
Επιλέξτε “Downloads” από το μενού και κατεβάστε το αρχείο που αντιστοιχεί στο λειτουργικό σύστημα που διαθέτετε.
Προτιμήστε την νεότερη έκδοση μορφής 3.x.y.
Τέλος, τρέξτε το αρχείο που μόλις κατεβάσατε και ακολουθήστε τις οδηγίες.

2) Εγκατάσταση Απαραιτήτων Βιβλιοθηκών

Το πρόγραμμα εκτελεί εντολές που ανήκουν στις ακόλουθες βιβλιοθήκες:
A) matplotlib
B) networkx

Αυτές οι βιβλιοθήκες δεν έρχονται προεγκατηστημένες με την Python. Επομένως, θα χρειαστεί να τις κατεβάσετε ξεχωριστά, όπως αναφέρεται στα ακόλουθα βήματα:
α) Στην γραμμή αναζήτησης windows πληκτρολογήστε “Γραμμή Εντολών” (στα αγγλικα: Command Prompt)

β) Όταν ανοίξετε την γραμμή εντολών, τρέξτε την εντολή: “pip install networkx”

γ) Αφότου ολοκληρωθεί η παραπάνω διαδικασία τρέξτε την εντολή: “pip install matplotlib”

3) Εκτέλεση προγράμματος

Αφότου αποθηκεύσετε το πρόγραμμα στον υπολογιστή σας και εγκαταστήσετε τις απαραίτητες βιβλιοθήκες, μπορείτε να τρέξετε το πρόγραμμα.
Αυτό μπορεί να γίνει με τους εξής τρόπους:

Α) Ανοίγοντας το αρχείο με κατάληξη .py με το περιβάλλον της επιλογής σας (ενδεικτικά: IDLE, VSCODE) και επιλέγοντας την εντολή “run”)

Β) Κάνοντας double click στο αρχείο με κατάληξη .py

Γ) Κάνοντας double click στο αρχείο με κατάληξη .exe

Επίσης, μαζί με το πρόγραμμα έχετε τη δυνατότητα να κατεβάσετε και βοηθητικά αρχεία, τα οποία περιέχουν έτοιμα test, δηλαδή με έτοιμα δεδομένα. Πρόκειται για τα αρχεία με όνομα “Test ...” και με κατάληξη .txt

Ενότητα 5η: Οδηγίες χρήσης προγράμματος

Για την κατανόηση των οδηγιών χρήσης προτείνεται η ανάγνωση της εισαγωγικής ενότητας παραπάνω στην έκθεση.

Στο πρόγραμμά μας, το γενικό (δοσμένο) γράφημα αποτελείται από όλες τις γραμμές που σχεδιάζονται στο πάνω δεξί frame, ενώ το MST μονό από τις κόκκινες.

Για να εισάγει ο χρήστης δεδομένα και να δημιουργήσει το γενικό γράφημα έχει τρεις επιλογές:

A. Καταχώρηση κορυφών και ακμών χειροκίνητα μέσω πληκτρολογίου.
Ο χρήστης εισάγει δεδομένα στη γραμμή εισαγωγής δεδομένων στο κάτω μέρος του δεξιού frame σε μορφή “x,y,z”. Έτσι, δημιουργεί μια ακμή βάρους z μεταξύ των κορυφών x και y.
Η σειρά με την οποία εισάγονται οι δύο κορυφές, x και y, δεν παίζει ρόλο, καθώς το γράφημα δεν έχει κατεύθυνση.
Όταν ο χρήστης θέλει να σταματήσει την χειροκίνητη εισαγωγή δεδομένων, πληκτρολογεί “stop”.

B. Δημιουργία τυχαίου γραφήματος.
Ο χρήστης εισάγει το πλήθος των κορυφών του τυχαίου γραφήματος καθώς και το μέγιστο πιθανό βάρος των κορυφών.

C. Εισαγωγή έτοιμων δεδομένων από αρχείο στον υπολογιστή σας.
Σε αυτή την περίπτωση ο χρήστης επιλέγει από την μνήμη του υπολογιστή του αρχείο μορφής .txt που περιέχει έτοιμη λίστα των κορυφών μαζί με τα βάρη των ακμών τους.
Εδώ πρέπει να σημειώσουμε ότι παρέχονται έτοιμα αρχεία .txt από τους δημιουργούς του προγράμματος. Βεβαίως ο χρήστης είναι ελεύθερος να δημιουργήσει νέο αρχείο .txt και να το χρησιμοποιήσει.
!Προσοχή: στο αρχείο τα δεδομένα θα πρέπει να έχουν την εξής μορφή:
x,y,z
x,y,z
...

Μόλις ολοκληρωθεί η εισαγωγή των δεδομένων ο χρήστης διαλέγει έναν από τους δύο αλγορίθμους για την εύρεση του MST.
Σημείωση: το αποτέλεσμα είναι το ίδιο, όποιος αλγόριθμος και να επιλεγεί.

Μόλις πατηθεί το κουμπί “Υπολογισμός” το πρόγραμμα υπολογίζει και εμφανίζει το MST.

Για να δημιουργήσει ο χρήστης καινούργιο γράφημα επιλέγει πάλι τρόπο εισαγωγής δεδομένων και ακολουθεί τα παραπάνω βήματα, χωρίς να κλείσει το πρόγραμμα και να το τρέξει εκ νέου.

Για να υπολογιστεί το MST του ίδιου γραφήματος με τον άλλο αλγόριθμο από αυτόν που επέλεξε ο χρήστης αρχικά, μπορεί να διαλέξει τον άλλο αλγόριθμο και να πατήσει “Υπολογισμός”. Στην περίπτωση αυτή το μόνο που αλλάζει είναι η διάταξη του γραφήματος, και όχι το αποτέλεσμα (αφού οι δύο αλγόριθμοι παράγουν το ίδιο αποτέλεσμα).

Ενότητα 6η: Κώδικας

```
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import (
   FigureCanvasTkAgg, NavigationToolbar2Tk)
import random
import tkinter as tk
from os import getcwd
import sys
import webbrowser
from PIL import ImageTk,Image


#Παλέτα χρωμάτων
NODE_COL = '#fcbf49'  # χρωματάκι νεκταρινί :)
BG_COL = '#eae2b7'
MST_COL = "#d62828"
GRAPH_COL = "#003049"


#Ορίζουμε την κλάση GUI - Γραφικό Περιβάλλον
class GUI():
   '''Σχεδιάζει το γραφικό περιβάλλον με χρήση tkinter και χειρίζεται το input
του χρήστη'''

   def __init__(self, root):   # Αρχικοποίηση τησ κλάσης
      self.root = root
      self.root.title("Minimum Spaning Tree Calculator")
      self.root.iconbitmap("test_icon121.ico")
      self.root.protocol("WM_DELETE_WINDOW", self.ask_quit)
      self.root.geometry('900x700')

      self.akmes = {}
      self.korifes = []

      self.buttons()
      self.canvas()
      self.ioframe()
```

```
        self.root.columnconfigure(1, weight=2)
        self.root.rowconfigure(0, weight=1)

        self.fr = tk.Frame(self.root, height=20, bd=2, relief=tk.SUNKEN,
bg=BG_COL)
        self.fr.grid(row=2, column=0, columnspan=2, sticky='snew')

        self.root.update()
        self.root.minsize(root.winfo_width(), root.winfo_height())


    def weight_check(self, varos): # Ελεγχος της ορθότητας των δεδομένων που
εισάγει ο χρήστης
        if not varos.replace(".", "", 1).replace("-", "").isnumeric():
            self.entry_box.delete(0, "end")
            return 1

        elif float(varos) <= 0:
            self.entry_box.delete(0, "end")
            return 2

        else:
            return 0


    def buttons(self):  # Δημιουργεί τα κουμπιά
        self.buttons_frame = tk.Frame(self.root)
        self.buttons_frame.grid(row=0, column=0, rowspan=2, sticky="nsew")
        self.buttons_frame.columnconfigure(1, weight=1)


        self.manualbutton = tk.Button(
            self.buttons_frame, text="Χειροκίνητη εισαγωγή", cursor="hand2",
command=self.input_manually, width=20)
        self.manualbutton.grid(row=0, column=0, pady=(50,5))

        self.random_button = tk.Button(
            self.buttons_frame, text="Τυχαίο γράφημα", cursor="hand2",
command=self.input_random, width=20)
```

```
        self.random_button.grid(row=1, column=0, pady=(0, 5))

        self.file_button = tk.Button(
            self.buttons_frame, text="Είσαγωγή από αρχείο", cursor="hand2",
command=self.input_file, width=20)
        self.file_button.grid(row=2, column=0, pady=(0, 60))

        self.calculate = tk.Button(
            self.buttons_frame, text='Υπολογισμός', cursor="hand2",
state='disabled', command=self.mst)
        self.calculate.grid(row=5, column=0,pady=(31.36, 0))

        self.algorithm = tk.IntVar()

        R1 = tk.Radiobutton(self.buttons_frame, text="Αλγόριθμος του Prim",
cursor="hand2",
                            variable=self.algorithm, value=1)
        R1.grid(row=3, column=0)

        R2 = tk.Radiobutton(self.buttons_frame,
                            text="Αλγόριθμος του Kruskal", cursor="hand2",
variable=self.algorithm, value=2)
        R2.grid(row=4, column=0)



        self.info_button = tk.Button(
            self.buttons_frame, text='Οδηγίες', cursor="hand2",
command=self.info_window)
        self.info_button.grid(row=6, column=0, sticky='s', pady=(40, 0))

        self.buttons_frame.rowconfigure((6,7), weight=1)

        self.credits_button = tk.Button(self.buttons_frame, text='Δημιουργοί
προγράμματος', cursor="hand2", command=self.credits_window)
        self.credits_button.grid(row=7, column=0, sticky='s', padx=10)



        self.ccimg = Image.open('creative_commons.png')

        width, height = self.ccimg.size
```

```
        new_width = 158
        new_height = int(new_width * height / width)

        self.img = self.ccimg.resize((new_width, new_height), Image.ANTIALIAS)
        self.img = ImageTk.PhotoImage(self.img)

        self.ccbutton = tk.Label(self.buttons_frame, image =self.img,
cursor="hand2")
        self.ccbutton.grid(row=8,column=0)
        self.ccbutton.bind('<Button-1>', lambda x_: webbrowser.open(
            'https://creativecommons.org/licenses/by-nc-nd/4.0/'))

        cctext = "This work is licensed under a Creative Commons Attribution-
NonCommercial-NoDerivatives\n4.0 International License."

        def del_cctext(event):
            self.text_box.configure(state='normal')
            self.text_box.delete('end-116c', 'end')
            self.text_box.insert('end', '\n')
            self.text_box.see("end")
            self.text_box.configure(state='disabled')

        self.ccbutton.bind('<Enter>', lambda y_: self.add_text(cctext))
        self.ccbutton.bind('<Leave>', del_cctext)

    def canvas(self):   # Δημιουργεί τον καμβά που εμφανίζεται το Minimum
Spaning Tree
        self.plot_frame = tk.Frame(self.root)
        self.plot_frame.grid(row=0, column=1, sticky=("nsew"))

        self.plot_frame.rowconfigure(0, weight=1)
        self.plot_frame.columnconfigure(0, weight=1)

        self.fig = plt.figure(figsize=(5, 4), dpi=100)
        self.fig.set_facecolor(BG_COL)

        self.canvas = FigureCanvasTkAgg(self.fig, master=self.plot_frame,)
        self.canvas.get_tk_widget().grid(row=0, column=0, sticky=("nsew"))

        self.Toolbar = NavigationToolbar2Tk(
```

```
        self.canvas, self.plot_frame, pack_toolbar=False)
    self.Toolbar.grid(row=1,column=0, sticky=("sew"))

def ioframe(self):  # Δημιουργεί το δεξί frame αλληλεπίδρασης
    self.io_frame = tk.Frame(self.root)
    self.io_frame.grid(row=1, column=1, sticky=("nsew"))

    self.text_box = tk.Text(self.io_frame, height=10, state='disabled')
    self.text_box.pack(fill='both', expand=True)

    self.entry_box = tk.Entry(self.io_frame, state='disabled', font='Calibri 14')
    self.entry_box.pack(fill='x', expand=True)


def add_text(self, txt):    # Εκτυπώνει το κείμενο txt στο Text widget
    self.text_box.configure(state='normal')
    self.text_box.insert('end', txt+'\n')
    self.text_box.see("end")
    self.text_box.configure(state='disabled')

def start_of_input(self):   # Αρχικοποίηση της εισαγωγής
    self.korifes = []
    self.G = nx.Graph()
    self.G.clear()
    self.akmes = {}
    self.minimum_spanning_tree = 0
    self.algorithm.set(0)
    self.calculate['state']='normal'
    self.text_box.configure(state='normal')
    self.text_box.delete('1.0', tk.END)
    self.text_box.configure(state='disabled')


def input_file(self):   # Μέθοδος για εισαγωγή γραφήματος απο αρχείο
    self.start_of_input()
    self.add_text('Επιλογή αρχείου για το Δέντρο.')

    try:
        filepath = tk.filedialog.askopenfilename(
```

```
            initialdir=getcwd(), title="Επέλεξε αρχείο", filetypes=(("Text files",
"*.txt*"), ('Comma-Seperated Values', '*.csv*'), ("all files", "*.*")))

        with open(filepath, 'r', encoding='utf-8') as f:
            i = 0
            for line in f:
                try:
                    x1, x2, n = line.split(",")
                except:
                    self.add_text("Η μορφή του περιεχομένου του αρχείου δεν είναι
αποδεκτή.")
                    break

                cont = True

                i += 1

                try:
                    n = float(n)
                    if n <= 0:
                        self.add_text(
                            "Το βάρος είναι λάθος στην γραμμή "+str(i)+" :). Δεν μπορεί
να είναι αρνητικό!")
                        self.calculate['state'] = 'disabled'
                        cont = False

                except ValueError:
                    self.add_text(
                        "Το βάρος είναι λάθος στην γραμμή {} :). Δεν μπορεί να είναι
γράμμα!".format(i))
                    self.calculate['state'] = 'disabled'
                    cont = False

                if not cont:
                    break

                if x1 not in self.korifes:
                    self.korifes.append(x1)
                if x2 not in self.korifes:
                    self.korifes.append(x2)
```

```
                self.akmes.update({"{}-{}".format(x1, x2): n})

                self.G.add_edge(x1, x2, weight=n)

            if cont:
                for key in self.akmes:
                    self.add_text("Ακμή: {:s} , βάρος: {:.2f}".format(
                        str(key), self.akmes[key]))

        except FileNotFoundError:
            self.add_text("Δεν επέλεξες κάποιο αρχείο")
            self.calculate['state'] = 'disabled'
            return

    def input_random(self):  # Μέθοδος για τυχαία εισαγωγή γραφήματος
        self.start_of_input()
        self.entry_box.configure(state='normal')

        def text_input(txt):
            if self.metritis == 0:
                txt = self.entry_box.get()
                state = self.weight_check(txt)
                if state == 1:
                    self.add_text("Η είσοδος δεν πρέπει να περιέχει γράμματα")
                    return

                elif state == 2:
                    self.add_text("Η εισοδος πρεπει να ειναι θετικη")
                    return

                elif state==0:
                    self.n = int(txt)
                    self.metritis += 1
                    self.add_text(
                        "Το πλήθος των κορυφών είναι: {} \nΔώσε το μέγιστο δυνατό
βάρος των ακμών".format(self.n))
                    self.entry_box.delete(0, "end")
                    return
```

```
        if self.metritis == 1:
            txt = self.entry_box.get()
            state = self.weight_check(txt)

            if state == 1:
                self.add_text("Η είσοδος δεν πρέπει να περιέχει γράμματα")
                return

            elif state == 2:
                self.add_text("Η εισοδος πρεπει να ειναι θετικη")
                return

            elif state==0:
                g = int(txt)
                self.metritis -= 1
                self.add_text(
                    "Το μέγιστο δυνατό βάρος των ακμών είναι {} \nΤο τυχαίο
γράφημα δημιουργήθηκε!!".format(g))
                self.entry_box.delete(0, "end")
            else:
                return

        self.korifes = []
        self.G = nx.Graph()
        self.G.clear()
        self.akmes = {}

        a = 0   # a = μεγιστος αριθμος ακμων

        for i in range(1, self.n):
            a += self.n-i

        b = self.n-1    # b = ελαχιστο πληθος ακμων για συνδεδεμενο γραφο

        for i in range(1, self.n + 1, 1):
            self.korifes.append(i)

        c = random.randint(1, 4)    # c = τυχαια επιλογη επαναληψης
προσθηκης ακμων
```

```
        while True:
            for i in range(b, a+c):
                x1 = random.choice(self.korifes)
                x2 = random.choice(self.korifes)
                v = random.randint(1, g)
                if x1 != x2 and "{}-{}".format(x1, x2) not in self.akmes.keys() and "{}-
{}".format(x2, x1) not in self.akmes.keys():
                    self.akmes.update({"{}-{}".format(x1, x2): v})
                    self.G.add_edge(x1, x2, weight=v)
            if nx.is_connected(self.G) == True and self.G.order() == self.n:
                break

        for key in self.akmes:
                self.add_text("Ακμή: {:s} , βάρος: {:.2f}".format(
                    str(key), self.akmes[key]))

        self.add_text(
            "Αν θες να δημιουργήσεις νέο τυχαίο δέντρο γράψε το επιθυμιτό
πλήθος των κορυφων του\nΑν θες να εμφανίσεις το γράφημα επέλεξε
αλγόριθμο και πάτα Υπολογισμός")

    self.add_text("Δώσε το πλήθος των κορυφών του τυχαίου γραφήματος")
    self.metritis = 0
    self.entry_box.bind("<Return>", text_input)

  def input_manually(self):   # Μέθοδος για χειροκίνητη εισαγωγή
γραφήματος απο τον χρήστη
    self.start_of_input()
    self.entry_box.configure(state='normal')

    def input_manually2(txt):
        txt = str(self.entry_box.get())
        self.entry_box.delete(0, tk.END)
        if txt == "stop":
            self.entry_box.configure(state='disabled')
            return

        try:
            x1, x2, n = txt.split(",")
```

```
            if x1 == x2:
                self.add_text("Παρακαλώ δώσε διαφορετικές κορυφές")
                return

        except ValueError:
            self.add_text("Παρακαλώ δώσε διαφορετικές κορυφές")
            return

        state = self.weight_check(n)
        if state==1:
            self.add_text("Το βάρος δεν πρέπει να περιέχει γράμματα")
            return

        elif state==2:
            self.add_text("Το βάρος πρεπει να ειναι θετικό")
            return

        if x1 not in self.korifes:
            self.korifes.append(x1)
        if x2 not in self.korifes:
            self.korifes.append(x2)

        n = float(n)
        if x1 != x2 and "{}-{}".format(x1, x2) not in self.akmes.keys() and "{}-
{}".format(x2, x1) not in self.akmes.keys():
            self.akmes.update(
                {"{}-{}".format(x1, x2): round(n, ndigits=2)})
            self.G.add_edge(x1, x2, weight=round(n, ndigits=2))
            self.add_text("{}, {}, {}".format(x1, x2, n))
        else:
            self.add_text("Παρακαλώ δώσε διαφορετικές κορυφές")
            return

    self.entry_box.bind("<Return>", input_manually2)
    self.add_text(
        'Δώσε τις ακμές (μορφή: κορυφή1,κορυφή2,βάρος)\n(Πληκτολόγησε
stop για να ολοκληρώσεις τις εισόδους σου)')
    self.akmes = {}
    self.korifes = []
    self.G = nx.Graph()
```

```
   def mst(self): # Υπολογισμός του Minimum Spaning Tree με χρήση του καταλληλου αλγοείθμου απο την κλάση MST
      if self.algorithm.get() == 0:
         self.add_text('Παρακαλώ επιλέξτε πρώτα τον αλγόριθμο επίλυσης.')

      elif MST.solvable_check(self, self.G, self.korifes):
         if self.algorithm.get() == 1:
            self.add_text('Το Ελάχιστο Διασυνδετικό Δέντρο του Prim')
            self.minimum_spanning_tree = MST.prim_graph(self, self.akmes)
            self.draw_graph()

         elif self.algorithm.get() == 2:
            self.add_text('Το Ελάχιστο Διασυνδετικό Δέντρο του Kruskal')
            self.minimum_spanning_tree = MST.kruskal_graph(self, self.akmes)
            self.draw_graph()

   def draw_graph(self):   # Σχεδιασμός γραφήματος στον καμβά
      plt.clf()

      colors = nx.get_edge_attributes(
         self.minimum_spanning_tree, 'color').values()

      weights = nx.get_edge_attributes(
         self.minimum_spanning_tree, 'weight').values()

      pos = nx.spring_layout(self.minimum_spanning_tree)

      labels = nx.get_edge_attributes(self.minimum_spanning_tree, 'weight')

      nx.draw(self.minimum_spanning_tree, pos, edge_color=colors, width=3,
            with_labels=True, node_color=NODE_COL, node_size=400)

      nx.draw_networkx_edge_labels(
         self.minimum_spanning_tree, pos, alpha=1,
bbox=dict(boxstyle='Circle,pad=0.1', facecolor=BG_COL, edgecolor=BG_COL),
edge_labels=labels, rotate=False)

      self.fig.set_facecolor(BG_COL)
```

```
        self.canvas.draw()

        #self.calculate['state'] = 'disabled'

    def ask_quit(self): # Εξοδος απο το πρόγραμμα
        if tk.messagebox.askokcancel("Έξοδος", "Θέλεις να τερματίσεις το
πρόγραμμα;"):
            self.root.destroy()
            sys.exit()

    def credits_window(self):   # Εμφάνιση δημιουργών προγράμματος
        self.creditsWindow = tk.Toplevel(self.root)
        self.creditsWindow.title("Δημιουργοί προγράμματος")
        self.creditsWindow.iconbitmap("test_icon121.ico")
        line1 = str("Αυτό το πρόγραμμα δημιουργήθηκε από τους:")
        line2 = str("Βλησίδης Γεώργιος\nvlisidisge002@gmail.com")
        line3 = str("Γεωργακόπουλος Γεώργιος\ngeorgegeo248@gmail.com")
        line4 = str("Κελεσίδης Αναστάσης\nkelesidis123@gmail.com")
        line5 = str("Λελάκης Πάνος\nplelakis@gmail.com")
        line6 = str("Σκαπινάκης Στρατής\nskapinakiss@gmail.com")
        line7 = str("Τσιατσιάνας Ευάγγελος\nvagtsiats12@gmail.com")
        self.creditsText = tk.Text(self.creditsWindow, height=28, state='normal')
        self.creditsText.pack()
        self.creditsText.insert(tk.INSERT, line1)
        self.creditsText.insert(tk.INSERT, '\n' + '\n' + line2)
        self.creditsText.insert(tk.INSERT, '\n' + '\n' + line3)
        self.creditsText.insert(tk.INSERT, '\n' + '\n' + line4)
        self.creditsText.insert(tk.INSERT, '\n' + '\n' + line5)
        self.creditsText.insert(tk.INSERT, '\n' + '\n' + line6)
        self.creditsText.insert(tk.INSERT, '\n' + '\n' + line7)
        self.creditsText.configure(state='disabled')

    def info_window(self):  # Εμφάνιση οδηγιών χρήσης του προγράμματος
        self.infowindow = tk.Toplevel(self.root)
        self.infowindow.title("Οδηγίες")
        self.infowindow.iconbitmap("test_icon121.ico")

        self.info_text = tk.Text(self.infowindow, width=100)

        with open('info.txt', 'r', encoding='utf-8') as f:
```

```
            for line in f:
                self.info_text.insert('end', line)

        self.info_text.pack()
        self.info_text.configure(state='disabled')

        self.infowindow.resizable(0, 0)


#Ορίζουμε την κλάση MST
class MST():
    '''Περιέχει μεθόδους που υπολογίζουν το Minimum Spaning Tree και μεθόδους ελέγχου'''

    def solvable_check(self, G, korifes):   # Έλεγχος καταλληλότητας του γραφήματος
        solvable = True
        if len(korifes) > 2:
            if nx.is_connected(G) == False:
                GUI.add_text(self, "Σφάλμα. Το γράφημα δεν είναι συνδεδεμένο.")
                solvable = False
        else:
            GUI.add_text(self, "Σφάλμα. Ο οριθμός των κορυφών δεν επαρκεί για την δημιουργία γραφήματος.")
            solvable = False

        return solvable

    def prim_graph(self, dict1):    # Αλγόριθμος του Prim για τον υπολογισμό του Minimum Spaning Tree
        self.dict1 = dict1

        cnctd_nodes = []
        korifes = []

        H = nx.Graph()
        D = nx.Graph()

        for edge in self.dict1.items():
            x1, x2 = edge[0].split("-")
```

```
    H.add_edge(x1, x2, color='#003049',
            weight=round(edge[1], ndigits=2))

    if x1 not in korifes:
        korifes.append(x1)
    if x2 not in korifes:
        korifes.append(x2)

r = random.choice(korifes)
cnctd_nodes.append(r)

while True:
    nbrs = {}
    for i in cnctd_nodes:
        for edge in self.dict1.items():
            if i in edge[0]:
                nbrs.update({edge[0]: edge[1]})

    nbrs = dict(sorted(nbrs.items(), key=lambda item: item[1]))

    for babushka in nbrs.keys():  # love_babushka
        add_edge = [babushka.split("-")[0], babushka.split("-")[1]]

        try:
            if D.has_edge(add_edge[0], add_edge[1]):
                continue

            D.add_edge(add_edge[0], add_edge[1])
            H.edges[add_edge[0], add_edge[1]]['color'] = "#d62828"

            nx.find_cycle(D)

            H.edges[add_edge[0], add_edge[1]]['color'] = "#003049"
            D.remove_edge(add_edge[0], add_edge[1])

        except:
            for n in add_edge:
                if n not in cnctd_nodes:
                    cnctd_nodes.append(n)
```

```
                break

            if D.order() == len(korifes):
                break

        return H

    def kruskal_graph(self, dict1): # Αλγόριθμος του Kruskal για τον υπολογισμό του Minimum Spaning Tree
        self.dict1 = dict1

        self.dict1 = dict(sorted(self.dict1.items(), key=lambda item: item[1]))

        H = nx.Graph()
        D = nx.Graph()

        for edge in self.dict1.items():
            x1, x2 = edge[0].split("-")

            H.add_edge(x1, x2, color='#003049',
                       weight=round(edge[1], ndigits=2))

            try:
                D.add_edge(x1, x2)
                H.edges[x1, x2]['color'] = "#d62828"

                nx.find_cycle(D)

                H.edges[x1, x2]['color'] = "#003049"
                D.remove_edge(x1, x2)

            except:
                continue

        return H


#Ορίζουμε την κλάση Βασικό Πρόγραμμα
class Main():
    '''Βασικές λειτουργίες προγράμματος'''
```

```
    root = tk.Tk()
    GUI(root)
    root.mainloop()


Main()
```

Πηγές

Εισαγωγή στο πρόβλημα:

https://en.wikipedia.org/wiki/Minimum_spanning_tree

Αλγόριθμος Kruskal:

https://www.youtube.com/watch?v=Yo7sddEVONg&feature=share

https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

https://stackabuse.com/how-to-sort-dictionary-by-value-in-python/

Αλγόριθμος Prim:

https://www.youtube.com/watch?v=Uj47dxYPow8&feature=share

https://en.wikipedia.org/wiki/Prim%27s_algorithm

https://bradfieldcs.com/algos/graphs/prims-spanning-tree-algorithm/

https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/

Εισαγωγή στις βιβλιοθήκες networkx και matplotlib:

https://www.youtube.com/watch?v=Q-qKKQLNLPo&list=LL&index=47

https://www.youtube.com/watch?v=sGAT2npnNLc&list=LL&index=46

Περισσότερα για τη βιβλιοθήκη networkx:

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

http://avinashu.com/tutorial/pythontutorialnew/NetworkXBasics.html \

https://stackoverflow.com/questions/35683302/python-networkx-detecting-loops-circles

Περισσότερα για τη βιβλιοθήκη Tkinter:

https://stackoverflow.com/questions/50422735/tkinter-resize-frame-and-contents-with-main-window

https://www.daniweb.com/programming/software-development/threads/262029/tkinter-interactive-terminal-as-widget

https://stackoverflow.com/questions/59164314/how-can-i-create-a-small-idle-like-python-shell-in-tkinter

https://stackoverflow.com/questions/53580507/disable-enable-button-in-tkinter

Περισσότερα για τη βιβλιοθήκη matplotlib:

https://matplotlib.org/3.3.3/api/_as_gen/matplotlib.pyplot.clf.html

Χρώματα:

https://stackoverflow.com/questions/25639169/networkx-change-color-width-according-to-edge-attributes-inconsistent-result

https://www.color-hex.com/color-palettes/

Μετατροπή προγράμματος σε εκτελέσιμη μορφή (.exe):

https://www.pyinstaller.org/

https://pyinstaller.readthedocs.io/en/stable/usage.html

Σχεδίαση του εικονιδίου του προγράμματος:

https://docs.microsoft.com/en-us/windows/win32/uxguide/vis-icons

Μήνυμα εξόδου από πρόγραμμα:

https://www.semicolonworld.com/question/44508/how-do-i-handle-the-window-close-event-in-tkinter

Απεικόνιση γραφήματος matplotlib στο παράθυρο Tkinter:

https://matplotlib.org/3.1.0/gallery/user_interfaces/embedding_in_tk_sgskip.html