

|  |
| --- |
| Μοντελοποίηση και επίλυση του προβλήματος του περιοδεύοντος πωλητή (Travelling Salesman Problem) |
|  |
| ιανουαριοσ 2020  Ομάδα 42  Αποτέλεσμα εικόνας για πανεπιστημιο πατρων ece |

Περιεχόμενα

[Τρόπος οργάνωσης ομάδας 3](#_Toc28959685)

[Κώδικας 3](#_Toc28959686)

[Πληροφορίες περί των χρησιμοποιούμενων βιβλιοθηκών 3](#_Toc28959687)

[Γενικά 3](#_Toc28959688)

[Περί του προβλήματος… 3](#_Toc28959689)

[Οδηγίες Εγκατάστασης 4](#_Toc28959690)

[Εγκατάσταση της Python 4](#_Toc28959691)

[Εγκατάσταση των βιβλιοθηκών ‘Pymprog’ και ‘swiglpk’ 6](#_Toc28959692)

[Μέθοδοι επίλυσης του προβλήματος 9](#_Toc28959693)

[Γενικά 9](#_Toc28959694)

[1ος Τρόπος επίλυσης *(βιβλιοθήκη Pymprog)* 9](#_Toc28959695)

[Χαρακτηριστικά της Pymprog 9](#_Toc28959696)

[Μαθηματικός Προγραμματισμός *(Mathematical Programming)* 9](#_Toc28959697)

[Κώδικας 1ου Τρόπου 11](#_Toc28959698)

[2ος Τρόπος Επίλυσης *(αλγόριθμος Heinritz-Hsiao)* 12](#_Toc28959699)

[Περιγραφή του αλγορίθμου 12](#_Toc28959700)

[Κώδικας 2ου Τρόπου 13](#_Toc28959701)

[3ος Τρόπος Επίλυσης *(τυχαία διαδρομή)* 14](#_Toc28959702)

[Κώδικας 3ου Τρόπου 14](#_Toc28959703)

# Τρόπος οργάνωσης ομάδας

## Κώδικας

Τις μεθόδους 1,2 επιμελήθηκε ο Εμμανουήλ Νικολόπουλος, ενώ την ευρύτερη υλοποίηση, συνένωση και σύνταξη της εφαρμογής παρουσίασης πραγματοποίησε ο Βασίλης Μιχαηλίδης.

## Πληροφορίες περί των χρησιμοποιούμενων βιβλιοθηκών

Τις πληροφορίες για τις βιβλιοθήκες επιμελήθηκαν ο Χάρης Γαλάνης και ο Φώτης Γκολφινόπουλος.

## Γενικά

Την δημιουργία και μορφοποίηση του παρόντος ανέλαβε ο Εμμανουήλ Νικολόπουλος.

# Περί του προβλήματος…

Το πρόβλημα του περιοδεύοντος πωλητή, (Travelling Salesman Problem), είναι ένα από τα πιο γνωστά διαχρονικά προβλήματα, το οποίο λαμβάνει μεγάλες διαστάσεις στην σημερινή εποχή, όπου οι απαιτήσεις αυξάνονται, μαζί με την αύξηση της κλίμακας του παγκοσμίου εμπορίου. Ας υποθέσουμε ότι ένας πωλητής εργάζεται σε μια εταιρεία η οποία εδρεύει στην Πάτρα. Ωστόσο, στην σύγχρονη εποχή η εταιρεία θα πρέπει να επεκταθεί και σε άλλες περιοχές ώστε να είναι ανταγωνίσιμη. Έτσι ο διευθυντής αποφασίζει να στείλει τον πωλητή και σε άλλες περιοχές για να προωθήσει τα προϊόντα της εταιρείας. Ο πωλητής λοιπόν οφείλει να επισκεφτεί συγκεκριμένο αριθμό πόλεων (n πόλεις), και για να μην απουσιάσει καιρό από το σπίτι του θα πρέπει να επιλέξει την κατάλληλη σειρά με την οποία θα επισκεφτεί τις εν λόγω πόλεις. Κανονικά, υπάρχουν και άλλες παράμετροι οι οποίες επηρεάζουν την απόφασή του, όπως το κόστος των διοδίων, η ποιότητα του δρόμου, η κίνηση…αλλά οι παράγοντες αυτοί θα αγνοηθούν προς απλοποίηση του προβλήματος, θα ασχοληθούμε δηλαδή αποκλειστικά με την απόσταση των πόλεων, θεωρώντας το κόστος μεταφοράς και τον χρόνο ανάλογο της απόστασης.

Έτσι λοιπόν το πρόβλημα διαμορφώνεται με τα εξής δεδομένα:

1. Η αφετηρία και το τέλος του ταξιδιού του πωλητή ταυτίζονται
2. Από κάθε πόλη περνάει αποκλειστικά μία φορά
3. Αναζητούμε την μικρότερη δυνατή απόσταση

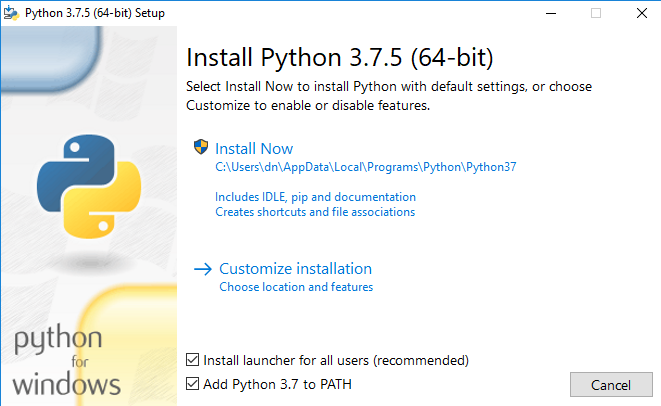
Όντας ένα από τα παλαιότερα προβλήματα με τα οποία ο άνθρωπος έρχεται αντιμέτωπος υπάρχουν αρκετές λύσεις. Ωστόσο η βέλτιστη διαδρομή μεταξύ (n πόλεων) παραμένει άγνωστη.

# Οδηγίες Εγκατάστασης

## Εγκατάσταση της Python

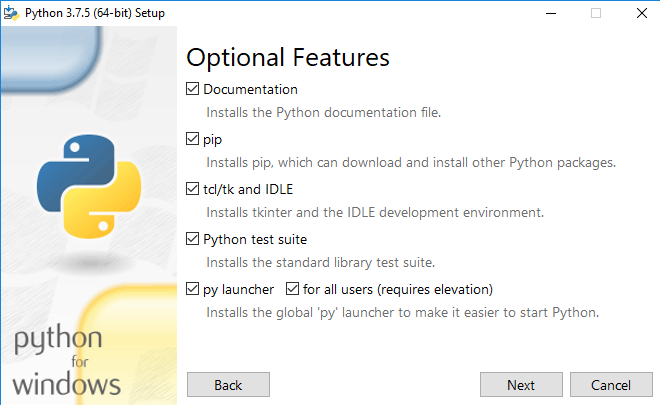
Αρχικά πρέπει να εγκατασταθεί η έκδοση 3.7.5 της Python, (η επιλογή της εν λόγω έκδοσης γίνεται με κριτήριο την προσφερόμενη σταθερότητα και προς αποφυγήν δυσκολιών εγκατάστασης). Η εγκατάσταση γίνεται μέσω του συνδέσμου: <https://www.python.org/downloads/release/python-375/> .

Μετά την λήψη του executable file επιλέγουμε την έναρξη της εγκατάστασης.



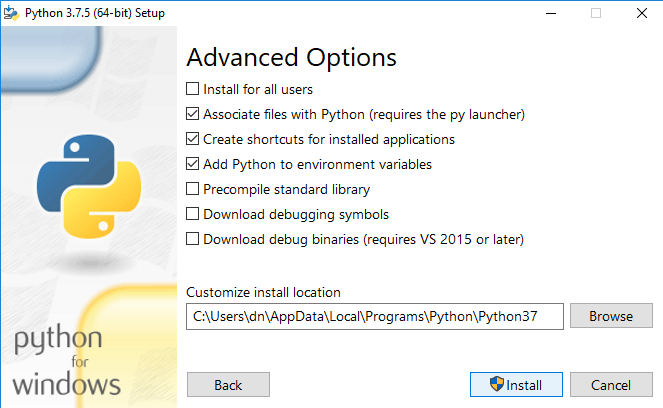
*Εικόνα 1.1.1*

Επιλέγουμε το ‘Add Python 3.7 to PATH’ και συνεχίζουμε με την εντολή ‘Customize installation’.



*Εικόνα 1.1.2*

Αφού σιγουρευτούμε ότι όλες οι επιλογές έχουν επιλεχθεί όπως στην εικόνα 1.1.2 πατάμε την επιλογή ‘Next’.



*Εικόνα 1.1.3*

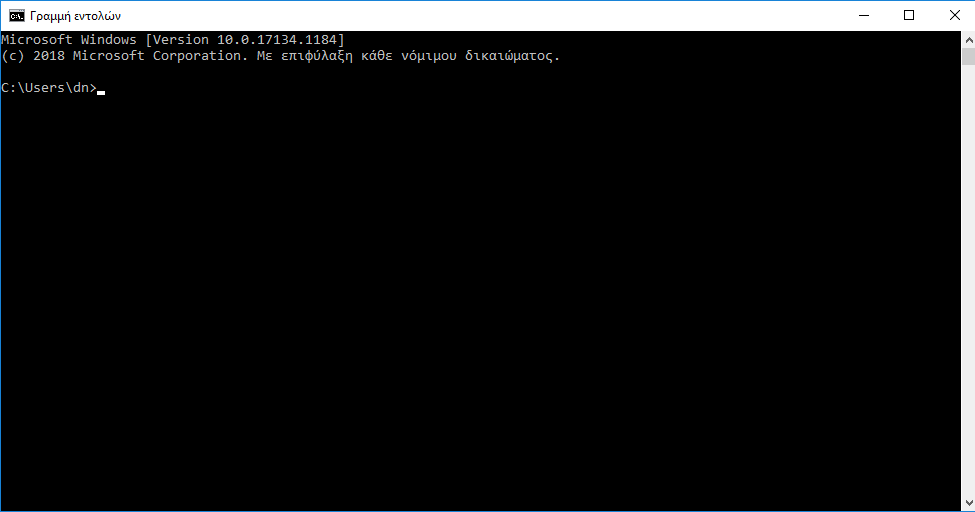
Στην συνέχεια αφού επιλέξουμε τις επιλογές της εικόνας 1.1.3 πατάμε το ‘Install’.

Με τον τις προαναφερθείσες εντολές έχουμε δημιουργήσει ένα νέο Path στον υπολογιστή μας, απαραίτητο για την συνέχεια. Προσοχή! Δεν πρέπει να υπάρχει άλλο Path της Python, όποτε σε περίπτωση που προϋπήρχε θα πρέπει να γίνει σωστό uninstall της προηγούμενης έκδοσης της Python μέσω των Windows και εκκαθάριση τυχών εναπομεινάντων Path.

## Εγκατάσταση των βιβλιοθηκών ‘Pymprog’ και ‘swiglpk’

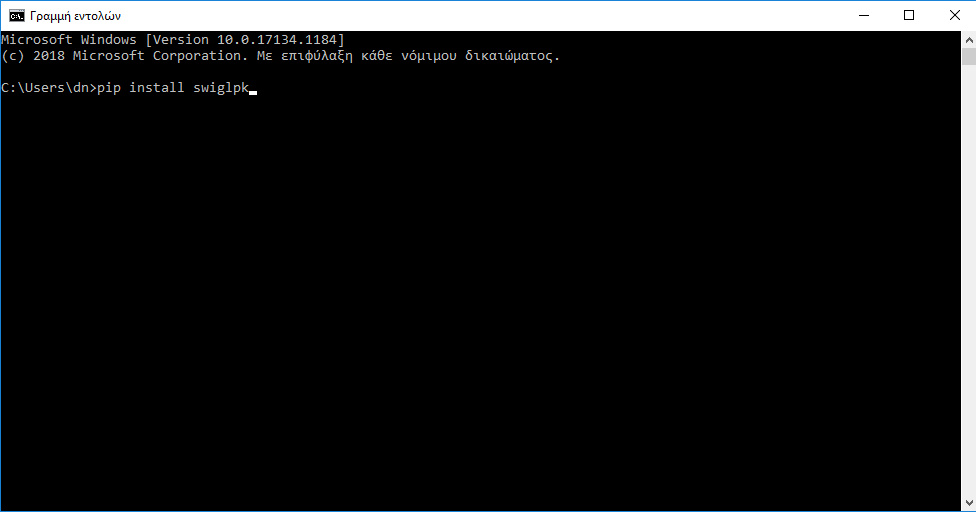
Τώρα θα πρέπει να γίνει εγκατάσταση των βιβλιοθηκών οι οποίες χρησιμοποιήθηκαν.

Ανοίγουμε μια γραμμή εντολών (Command Prompt) (εικόνα 1.2.1).



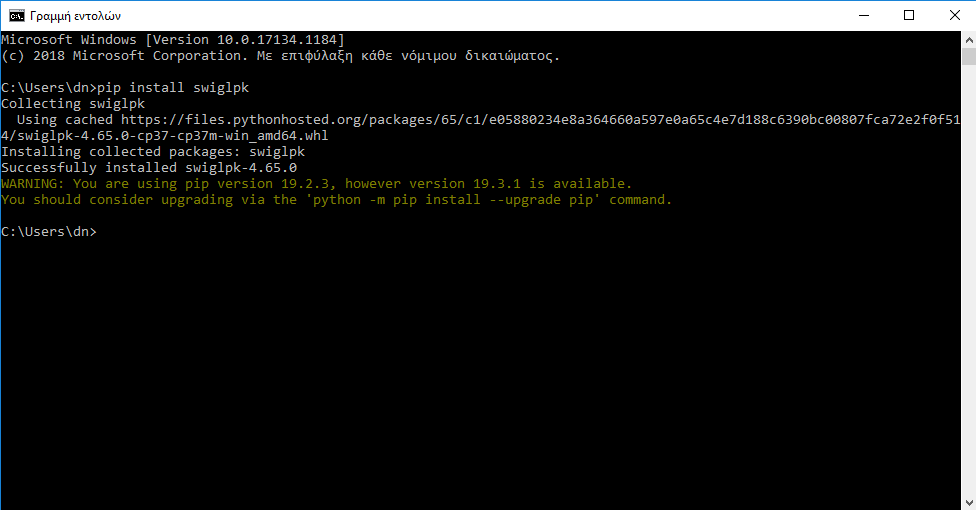
*Εικόνα 1.2.1*

Κατόπιν πληκτρολογούμε την εντολή ‘pip install swiglpk’ (εικόνα 1.2.2)

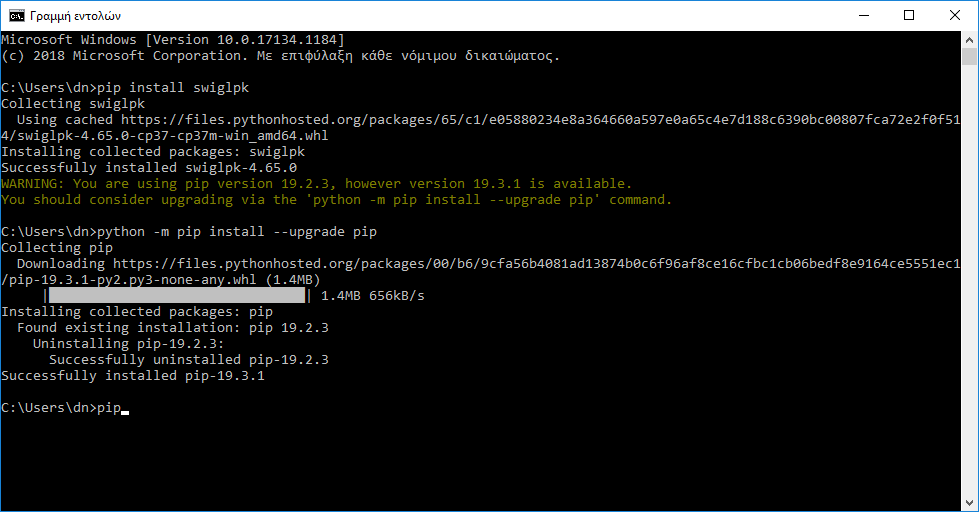


*Εικόνα 1.2.2*

Η εγκατάσταση ολοκληρώνεται. (εικόνα 1.2.3)

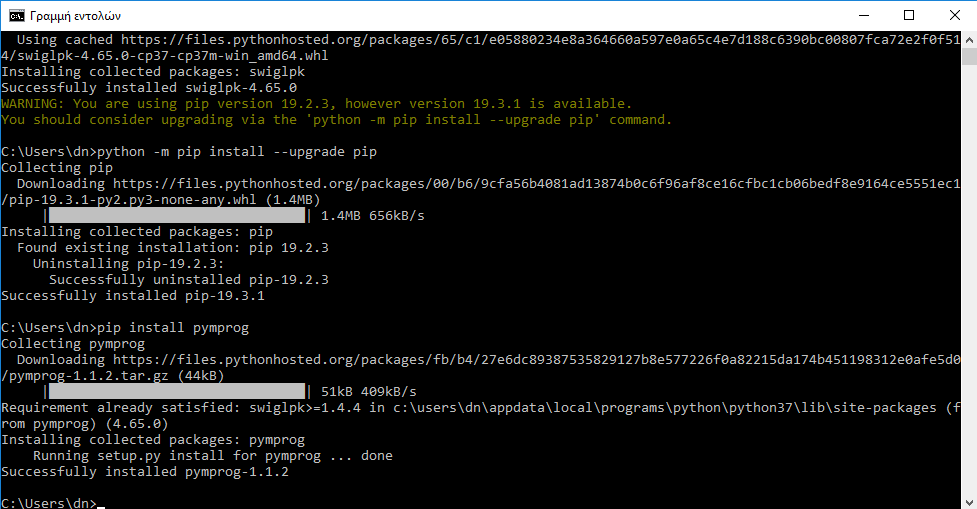
Σε περίπτωση που ζητηθεί update της ‘pip install’ (εικόνα 1.2.3) ,

*Εικόνα 1.2.3*

τότε πληκτρολογούμε την εντολή ‘python –m pip install –upgrade pip’ (εικόνα 1.2.4)

*Εικόνα 1.2.4*

Στην συνέχεια εκτελούμε την εντολή ‘pip install pymprog’ (εικόνα 1.2.5)



*Εικόνα 1.2.5*

# Μέθοδοι επίλυσης του προβλήματος

## Γενικά

Για την επίλυση του προβλήματος χρησιμοποιήθηκαν 3 διαφορετικοί τρόποι επίλυσης, οι οποίοι διαφέρουν στο ποσοστό βελτιστοποίησης της λύσης, δηλαδή στο κατά πόσο το κλάσμα της εξίσωσης 2.0.1προσεγγίζει την μονάδα.

*(εξίσωση 2.0.1)*

Ωστόσο για μεγάλο αριθμό περιπτώσεων η εύρεση της βέλτιστης απόστασης είναι πρακτικά αδύνατη, άρα θα χρησιμοποιήσουμε την εξίσωση 2.0.2,

*(εξίσωση 2.0.1)*

όπου ……….

## 1ος Τρόπος επίλυσης *(βιβλιοθήκη Pymprog)*

### Χαρακτηριστικά της Pymprog

Η PyMathProg είναι ένα εύκολο και ευέλικτο μαθηματικό περιβάλλον προγραμματισμού για την Python. Κάνει την μοντελοποίηση, την επίλυση, την ανάλυση, την τροποποίηση και τον χειρισμό γραμμικών προγραμμάτων εξαιρετικά εύκολη διαδικασία στην Python.

Σημαντικά χαρακτηριστικά τηε βιβλιοθήκης είναι:

* Αποτελεί ένα εύκολο διαδραστικό περιβάλλον
* Διαθέτει προηγμένες επιλογές λύσης
* Διαθέτει εργονομική σύνταξη για μοντελοποίηση
* Ενημερώνεται αυτόματα για τις αλλαγές παραμέτρων
* Υποστηρίζει τις Python 3 και 2
* Υποστηρίζει όλες τις μεγάλες πλατφόρμες

### Μαθηματικός Προγραμματισμός *(Mathematical Programming)*

Μαθηματικός προγραμματισμός είναι το σύνολο των μεθόδων και υπολογιστικών τεχνικών που χρησιμοποιούνται για την επίλυση μιας κατηγορίας προβλημάτων βελτιστοποίησης. Μπορούμε να ορίσουμε τον μαθηματικό προγραμματισμό ως ένα κλάδο των εφαρμοσμένων μαθηματικών που έχει ως αντικείμενο την μεγιστοποίηση ή ελαχιστοποίηση πραγματικών συναρτήσεων, κάτω από ορισμένους περιορισμούς για τις μεταβλητές.

Το ιδιαίτερο χαρακτηριστικό του είναι ότι η καλύτερη λύση σε ένα μοντέλο εντοπίζεται αυτόματα από λογισμικό βελτιστοποίησης. Ένα μοντέλο MP απαντά στην ερώτηση :

"Τι είναι καλύτερο;" αντί "Τι συνέβη;" (στατιστικά στοιχεία),

"Τι εάν;" (προσομοίωση), "Τι θα συμβεί;" (πρόβλεψη)

ή "Τι θα έκανε ένας εμπειρογνώμονας και γιατί;" (συστήματα εμπειρογνωμόνων).

Ο Μαθηματικός Προγραμματισμός είναι κατάλληλος για προβλήματα που συνεπάγονται ανάμειξη, επεξεργασία συνεχούς ροής, παραγωγή , διανομή και στρατηγικό σχεδιασμό. Απαντά σε ερωτήσεις όπως:

* Πόσο?
* Πότε?
* Πού?

Ποια η αιτία εφεύρεσής του?

Πριν από την βιομηχανική επανάσταση οι επιχειρήσεις ήταν μικροί οργανισμοί. Ωστόσο , από το 1760 (Βιομηχανική Επανάσταση) και μετά οι επιχειρήσεις άρχισαν να μεγαλώνουν και οι απαιτήσεις να αυξάνονται , με αποτέλεσμα οι μικροί πρώτα οργανισμοί να εξελίσσονται σε κολοσσούς επιχειρήσεων με μεγάλο εργατικό δυναμικό ,το οποίο έχει ως άμεση συνέπεια την αύξηση της πολυπλοκότητας στην διοίκησης ενός τέτοιου οργανισμού. Αυτού του είδους τα προβλήματα καθώς και η ανάγκη να βρεθεί ένας καλύτερος τρόπος να λυθούν συνέβαλαν στην δημιουργία του μαθηματικού προγραμματισμού. Επιπλέον, ο Μαθηματικός Προγραμματισμός είναι ίσως η πιο αποτελεσματική τεχνική για τη λήψη βέλτιστων αποφάσεων. Έχει μια πολύ ευρεία εφαρμογή στην ανάλυση των προβλημάτων διαχείρισης, στις επιχειρήσεις και στη βιομηχανία, στις οικονομικές μελέτες, στα στρατιωτικά προβλήματα και σε πολλούς άλλους τομείς των σημερινών μας δραστηριοτήτων.

Μαθηματικά που μπορεί να χρησιμοποιήσει ένας προγραμματιστής βάση αυτού:

1. Πρόσθεση / Αφαίρεση / Πολλαπλασιασμός / Διαίρεση
2. Modulo (%)
3. Εκθέτης (pow)
4. Απόλυτοι αριθμοί (abs)
5. Στρογγυλοποίηση
6. Πυθαγόρειο Θεώρημα. (a ^ 2 + b ^ 2 = c ^ 2)
7. Αριθμητική ακολουθία (1, 2, 3, ...)
8. Γεωμετρική σειρά (1 ^ 2, 2 ^ 2, 3 ^ 2, ...)
9. Δυαδικοί/Οκταδικοί/Δεκαεξαδικοί αριθμοί
10. Παραγοντικό
11. Συναρτήσεις
12. Αναδρομικές λειτουργίες (π.χ. ακολουθία Fibonacci, Factorial ενός αριθμού)
13. Εξαιρέσεις κυμαινόμενου σημείου (Epsilon, Division by 0, Infinity)
14. Τετραγωνική ρίζα
15. String (στην πραγματικότητα, ένα σύνολο αριθμών 8-bits)
16. Πίνακες (π.χ. Πίνακας διαίρεσης, Πίνακας καταγραφής)
17. Υπογραμμισμός (π.χ. +123, -123)
18. Σύγκριση αριθμών
19. Χειρισμός bit (Και, Ή, Xor, Όχι, Xnor, Neg, Nand, Nor)
20. Boolean αριθμητική (True ή False)
21. Λογική αριθμητική (&&, ||, ==)
22. Γεωμετρία (Hyperplane, σημεία στο άπειρο, κλπ.)
23. Τριγωνομετρία (cos, tan asin, atan2, κ.λπ.)
24. Vector (καρτεσιανό)
25. Θεωρία γραφημάτων
26. Θεωρία Αριθμών
27. Λογισμός

Μια ειδική περίπτωση του Μαθηματικού Προγραμματισμού είναι ο Γραμμικός Προγραμματισμός (LP). Σε ένα μοντέλο LP όλες οι σχέσεις είναι γραμμικές, εξ ου και το όνομα. Το LP βρίσκει εφαρμογή για δύο λόγους:

* Υπάρχουν ισχυροί λύτες "black box" οι οποίοι βρίσκουν αυτόματα την καλύτερη λύση στα προβλήματα LP
* Πολλά φαινόμενα του πραγματικού κόσμου μπορούν να προσεγγιστούν αρκετά καλά από τις γραμμικές σχέσεις.

### Κώδικας 1ου Τρόπου

(εισαγωγη εικονων 2.1.χ)

(εξηγηση κωδικα)

## 2ος Τρόπος Επίλυσης *(αλγόριθμος Heinritz-Hsiao)*

### Περιγραφή του αλγορίθμου

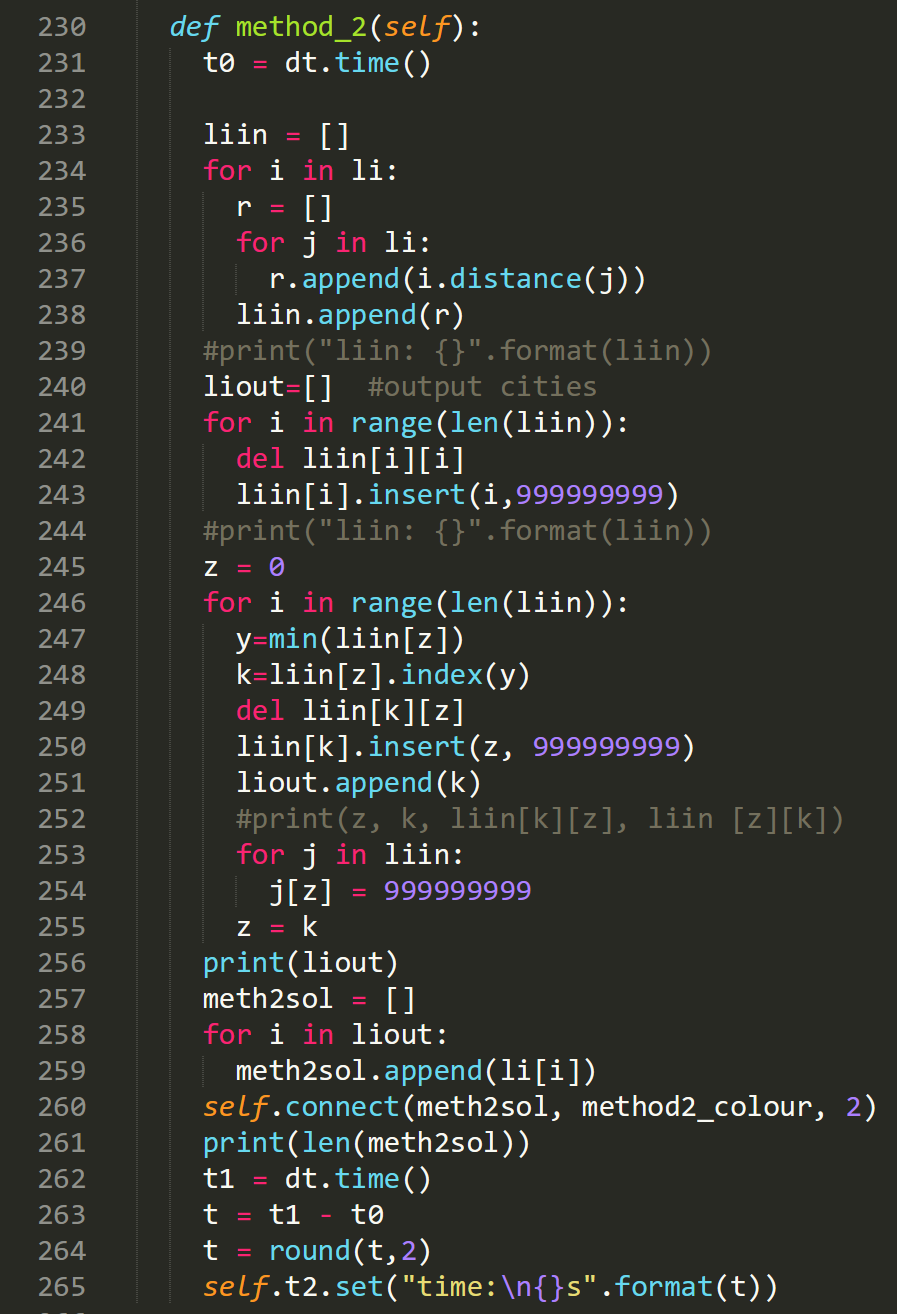
Δημιουργούμε έναν πίνακα *(πίνακας 2.2.1)* με τις πόλεις και τις μεταξύ τους αποστάσεις.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | - | A->B | A->C | A->D | A->E |
| B | B->A | - | B->C | B->D | B->E |
| C | C->A | C->B | - | C->D | C->E |
| D | D->A | D->B | D->C | - | D->E |
| E | E->A | E->B | E->C | E->D | - |

*πίνακας 2.2.1*

Κάνουμε την παραδοχή ότι ο πωλητής αρχίζει από το σημείο Α, και αναζητούμε το σημείο i (A,B,C,D,E,…) το οποίο είναι πιο κοντά στο Α, δηλαδή το ελάχιστο Α->i από τον πίνακα 2.2.1. Στην συνέχεια αναζητούμε την ελάχιστη απόσταση από το σημείο i στο σημείο j (B,C,D,E,… εκτός του σημείου i) , δηλαδή το ελάχιστο i->j από τον πίνακα. Προσοχή: κάθε φορά πρέπει να εξαιρούμε από τον πίνακα την πόλη στην οποία ο πωλητής έχει ήδη πάει. Με αυτόν τον τρόπο έχουμε δημιουργήσει μία προσεγγιστική λύση όπου κάθε φορά εντοπίζει το πιο κοντινό σημείο στην πόλη στην οποία βρίσκεται εκείνη την στιγμή ο πωλητής.

### Κώδικας 2ου Τρόπου



*Εικόνα 2.2.2*

Στην εικόνα 2.2.2 παρουσιάζεται ο κώδικας ο οποίος χρησιμοποιείται για την 2η μέθοδο. Αρχικά το *def method\_2* λαμβάνει μία λίστα *liiin* η οποία περιέχει n αντικείμενα κλάσης *Point*. Στις γραμμές 234-238 η *liin* γεμίζεται με n λίστες που περιέχουν n αντικείμενα ώστε κάθε σημείο *liin[i][j]* να αντιστοιχεί στο ίδιο σημείο του πίνακα αποστάσεων. Στις 241-243 αντικαθίσταται η απόσταση του σημείου i από τον εαυτό του, με μια πολύ μεγάλη απόσταση (αδύνατο να υπάρχει μεγαλύτερη), ώστε ο αλγόριθμος να μπορεί να προσχωρήσει στο επόμενο κοντινότερο σημείο.

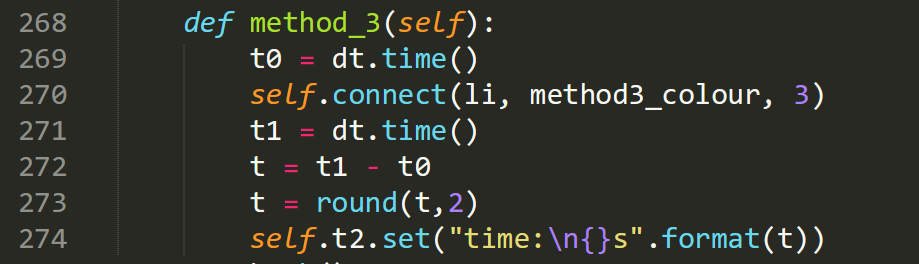
Στις 245-255 επιλέγουμε να ξεκινήσουμε από το σημείο *z=0* και για το μήκος της *liin* επαναλαμβάνουμε την εξής διαδικασία :

* Βρίσκουμε την απόσταση (y) του κοντινότερου σημείου (k) στο z
* Χρησιμοποιώντας την μέθοδο *index* βρίσκουμε την θέση του πίνακα αποστάσεων του σημείου *k*
* Στις 249-250 αντικαθιστούμε την απόσταση του *k* από το *z* στον πίνακα *liin[k]* με ένα πολύ μεγάλο νούμερο ώστε να αποφευχθεί η παλινδρόμηση.
* Στις 253-254 αντικαθιστούμε τις αποστάσεις όλων των υπολοίπων σημείων από το *k* με ένα πολύ μεγάλο νούμερο ώστε να αποφευχθεί πάλι η παλινδρόμηση.
* Προσθέτουμε το *k* στο *liiout* και δίνουμε στο *z* την τιμή *k,* ώστε να συνεχίσει η διαδικασία από το *k.*

## 3ος Τρόπος Επίλυσης *(τυχαία διαδρομή)*

Ο πωλητής αποφασίζει να μην αναζητήσει την μια καλύτερη εκδοχή και αφήνει την τύχη να επιλέξει για αυτόν…

### Κώδικας 3ου Τρόπου



*εικόνα 2.3.1*

Για τον 3ο  τρόπο απλά χρησιμοποιείται η μέθοδος *connect* για την ένωση των πόλεων σύμφωνα με τον τυχαίο τρόπο εισαγωγής τους στην λίστα *li* από την βιβλιοθήκη *random*.