****

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΑ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ ΚΑΙ ΕΜΠΕΙΡΑ ΣΥΣΤΗΜΑΤΑ**

**ΠΡΟΕΡΑΙΤΙΚΗ ΕΡΓΑΣΙΑ θέμα 1:πλανόδιου πωλητή**

**ΕΤΟΣ 2021-2022**

**ΙΩΑΝΝΗΣ-ΙΑΣΩΝ ΜΠΟΪΔΑΝΗΣ Π19217**

**ΔΙΔΑΣΚΩΝ:Δ.ΑΠΟΣΤΟΛΟΥ**

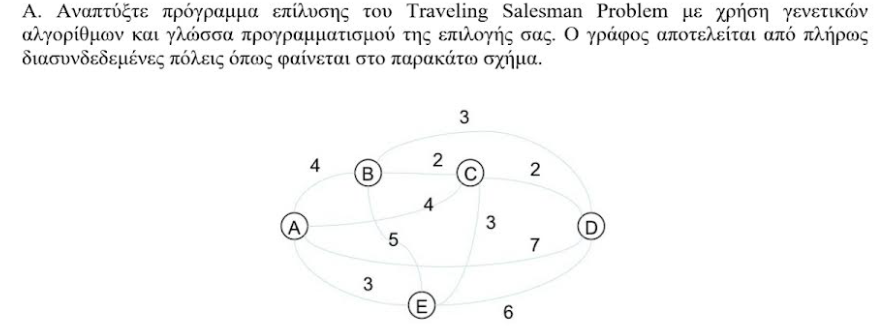
**Περιεχόμενα:**

**Ερώτημα………………σελ 1**

**Κώδικας…………………σελ 1-7**

**Εκτέλεση κώδικα…………..σελ 8**

**Παρουσίαση προβλήματος:**

****

**Αριθμώ τις πόλεις με νούμερα για να μπορέσω να κάνω εύκολο premutation χωρίς να μετατρέπω συνέχεια σε αριθμούς τα αλφαριθμιτικα. Και ταυτόχρονα κάνω ένα array με τα κόστη μετακίνησης.**

**Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές

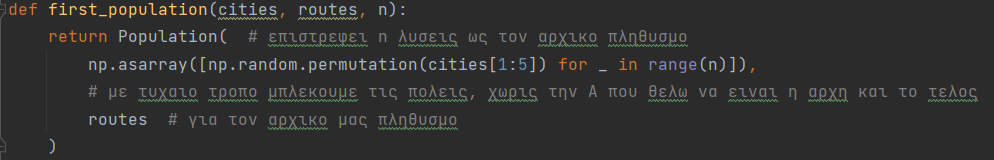
Περιγραφή που δημιουργήθηκε αυτόματα**

**Στη συνέχεια δημιουργώ μία κλάση πληθυσμός όπου μπορώ να την ενημερώνω με τα χαρακτηριστικά που θα χρειαστώ**

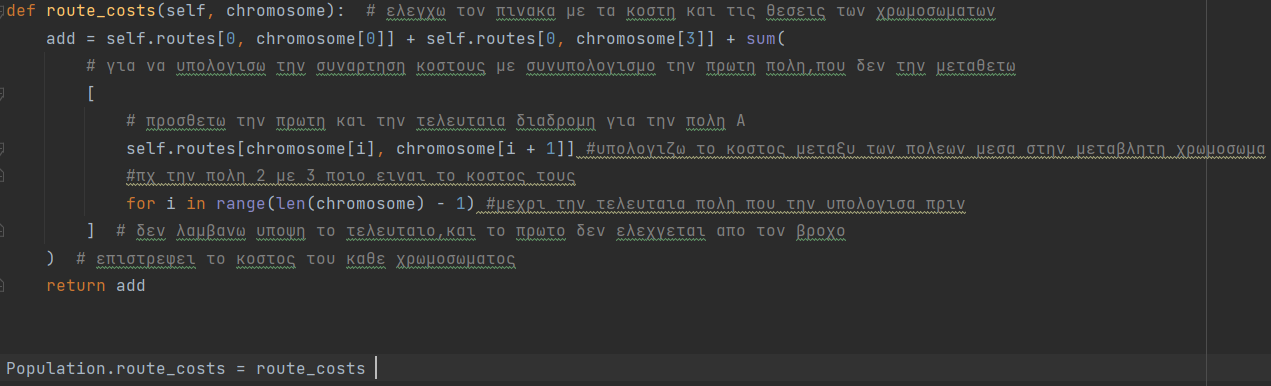
**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

**Αρχικά δεν έχουμε υποψήφιους γονείς, δεν έχουμε ελέγξει ακόμα κάποιο σκορ, άρα δεν έχουμε και κάποιο καλύτερο σκορ, στον πληθυσμό και τα κόστη των διαδρομών θα αποδοθούν μετά τιμές. Επίσης έχουμε την λίστα των λύσεων που δεν θα γίνουν γονείς και θα περάσουν στην επόμενη γενιά.**

****

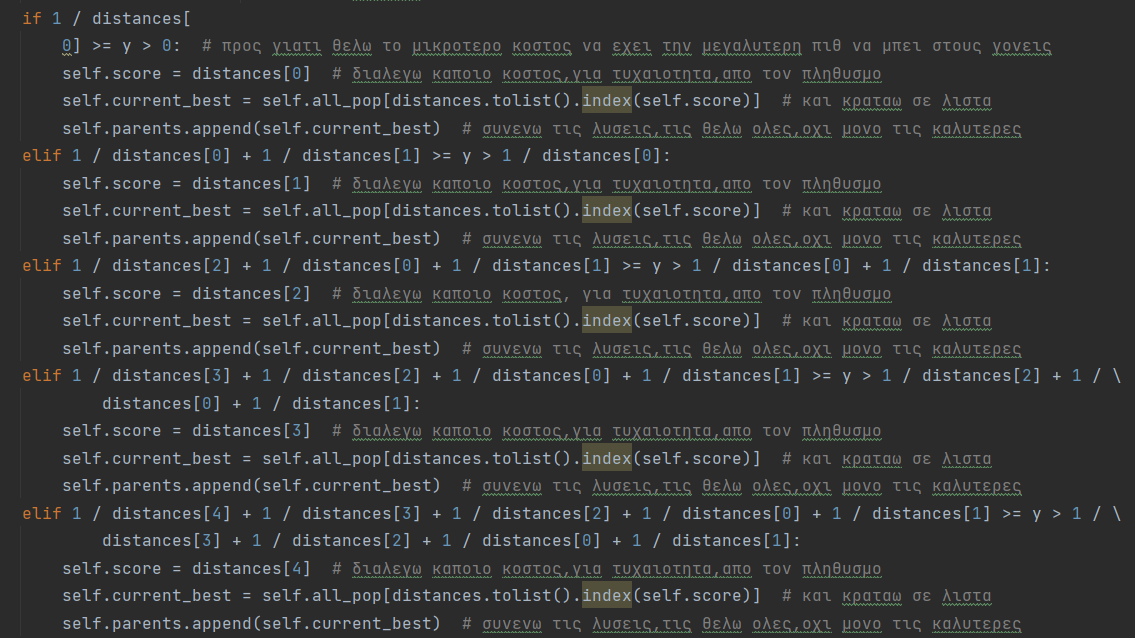
**Η πρώτη μέθοδος της κλάσης είναι για να δημιουργηθεί ο πρώτος πληθυσμός. Παίρνει ορίσματα τις 5 πόλεις(αν και θα χρησιμοποιηθούν οι 4ις), τον πίνακα με τα κόστη, και το πλήθος των υποψήφιων λύσεων που θέλουμε να έχουμε. Γίνεται μία τυχαία μετάθεση για τις πόλεις 1 2 3 και 4 δηλαδή για την β γ δ και ε αφού θέλουμε να ξεκινά και να τελειώνει στην πόλη Α, όσες φορές είναι το n και το μέγεθος του πίνακα routes. Επιστρέφει έναν πίνακα 4\*10, 4ις πόλεις για 10 υποψήφιες λύσεις.**

****

**Η επόμενη μέθοδος υπολογίζει το κόστος των λύσεων. Δίνω ως όρισμα την ίδια την κλάσση για τα χαρακτηριστικά της και την μεταβλητή chromosome που είναι η ακολουθία των πόλεων μέσα από τον πίνακα. Αρχικά προσθέτω το κόστος της πόλη Α με την επόμενη στο χρωμόσωμα της λύσης και της τελευταίας στην ακολουθία με την πόλη Α. Πχ αν έχουμε την ακολουθία [2 1 4 3] θα υπολογίσει το κόστος της πόλης 0 με 2 (Α με Γ) και μετά τις 3 με 0(Δ με Α) κι έπειτα υπολογίζει το κόστος των ενδιάμεσων πόλεων. Επιστρέφει αυτό το άθροισμα για κάθε υποψήφια λύση.**

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

****

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

**Για την επιλογή των γονιών για την επόμενη γενιά θέλω να διαλέξω 5 γονείς για μερική ανανέωση πληθυσμού. Συνολικά θα έχω 10 υποψήφιες λύσεις άρα διαγράφω τις παλιότερες, από τον πίνακα all\_pop, και κρατάω τις 10 νεότερες(κατά την δεύτερη επανάληψη θα εκτελεσθεί το Αν). Έπειτα για κάθε υποψήφια λύση υπολογίζω μέσω της συνάρτησης route\_costs το κόστος διαδρομής. Μετά για 5 φορές(όσους νέους γονείς θέλω) θα ρίξω ένα random μεταξύ το 0 και το αντίστροφο άθροισμα των αποστάσεων και θα χωρίσω σε αντίστοιχα διαστήματα τα κόστη ώστε το μικρότερο κόστος θα έχει την μεγαλύτερη πιθανότητα να μπει στους γονείς. Για παράδειγμα αν έχω τα κόστη των υποψήφιων λύσεων σε έναν πίνακα [17 18 16] θα πάρω το άθροισμα 1/17 +1/18 + 1/16, και θα επιλέξω τυχαία έναν αριθμό από το μηδέν μέχρι αυτόν τον αριθμό και μετά θα ελέγξω αν πέφτει στο διάστημα ανάμεσα στο 0 έως 1/17, αν όχι θα ελέγξω το διάστημα 1/17 μέχρι το 1/17+1/18, στον κώδικα είναι το 1/distances[0]+1/distances[1] >= ο τυχαίος αριθμός μου >= και το προηγούμενο διάστημα μου. Θα γίνει για κάθε διάστημα ανάμεσα στα 10 μου κόστη. Όταν βρεθεί ο αριθμός στο διάστημα αυτό ενημερώνω το τρέχον σκορ self.score ότι θα είναι το κόστος του distance μετά στην θέση που είναι πχ την πρώτη θέση, έτσι θα βρει το index μέσα στην λίστα των distances,θα επιστρέψει αριθμό άρα σε αυτή την θέση θα ψάξει στον πληθυσμό all\_pop για να βρει την ακολουθία των πόλεων. Όπου θα την βάλουμε σε μία μεταβλητή και θα εν τέλει στην λίστα των γονιών. Επίσης επιστρέφει τις αποστάσεις που μπορώ να εκτυπώσω για να δούμε ότι ο πληθυσμός μένει σταθερός ανάμεσα στις γενιές. Για τον μερικό πληθυσμό επιλέγω τυχαία κάποιες λύσεις και κρατάω τα στοιχεία που μας χρειάζονται για την διαδικασία της αναπαραγωγής επόμενης γενιάς.**

Εικόνα που περιέχει κείμενο

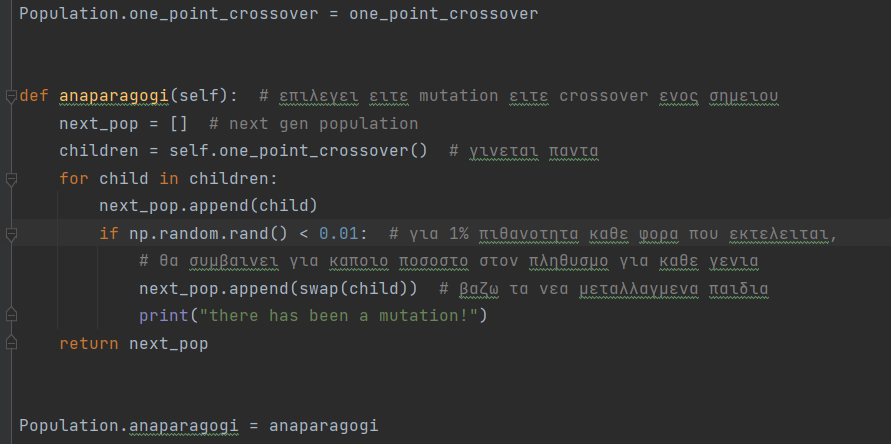
Περιγραφή που δημιουργήθηκε αυτόματα

**Η παραπάνω μέθοδος είναι η μετάλλαξη για ένα σημείο της ακολουθίας των πόλεων. Τυχαία επιλέγονται 2 σημεία και αντιμεταθέτονται , οπότε δεν υπάρχει θέμα στο να έχω δύο ίδιες πόλεις μέσα στο χρωμόσωμα της λύσης μου.**

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

**Για την διασταύρωση ενός σημείου χρειάζομαι τους γονείς, η επανάληψη θα γίνει όσοι είναι και οι γονείς 5 φορές με 1 παιδί η κάθε αναπαραγωγή άρα θα έχω 5 πληθυσμό και 5 από αυτούς που έχουν μείνει στο all\_pop, παίρνω το μέγεθος του πίνακα(όπου θα είναι 5 \* 4, 5 γονείς που διάλεξα πριν από τις 4ις πόλεις) και διαλέγω 2 τυχαίους, τους βάζω σε 2 μεταβλητές και επιλέγω μετά ένα τυχαίο σημείο(στην περίπτωση μου την μέση) για να γίνει η διασταύρωση. Έως αυτό το σημείο βάζω τις πόλεις του ενός γονέα στην άδεια λίστα του παιδιού, και μετά από το σημείο και μετά του άλλου γονέα. Συνενώνεται με την λίστα των παιδιών και μετά ξανακάνω την ίδια διαδικασία για τον υπόλοιπο πληθυσμό που δεν θα αναπαραχθεί και θα μεταβεί τυχαία στην επόμενη γενιά. Χρησιμοποίησα το self.parents γιατί είχα κάνει ήδη την διαδικασία εύρεσης των ακολουθιών των πόλεων από τα σκορ από την μέθοδο choose για να μην χρειαστεί να κάνω την ίδια επεξεργασία. Η επιλογή αυτών των γονέων είναι τυχαία όπως και η επιλογή του πληθυσμού που θα μείνει ίδιος. Κι τέλος επιστρέφεται αυτή η λίστα των παιδιών.**

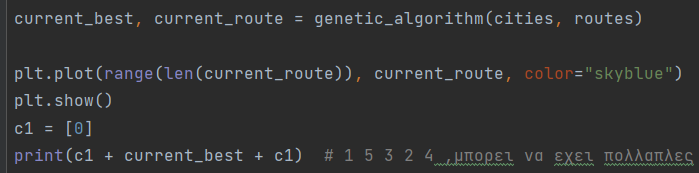
****

**Για την διαδικασία της αναπαραγωγής θα κάνω την διασταύρωση 100% των περιπτώσεων, και μετά κατά μία πιθανότητα 1% κάθε φορά που γίνεται ένα διασταύρωση θα γίνεται και μία μετάλλαξη στην λύση.**

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

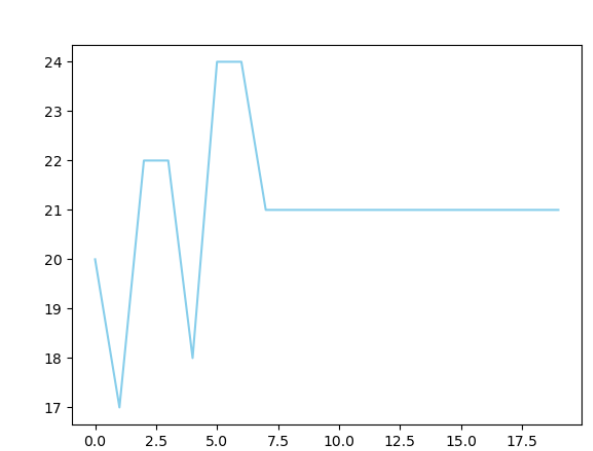
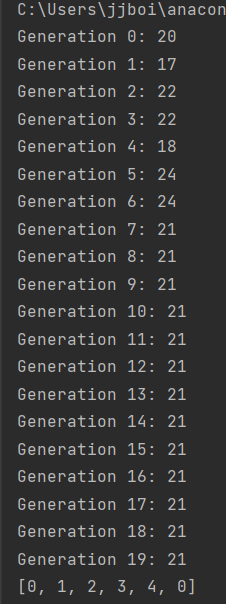
**Τέλος ο γενετικός αλγόριθμος παίρνει σαν ορίσματα τις πόλεις, τις διαδρομές και τα κόστη τους(τα οποία θα βάλω ως όρισμα όταν καλέσω τον γεννετικό αλγόριθμο) , θα εκτελεσθεί για πληθυσμό 10,κι για το πολύ 20 γενιές. Ξεκινάμε με τον αρχικό πληθυσμό καλώντας την συνάρτηση. Αρχικοπιούμε το καλύτερο μέχρι στιγμής σκορ. Και το ελέγχουμε αρχικά με το άπειρο για να βρούμε μικρότερο. Επίσης κρατώ και την καλύτερη διαδρομή που έχω βρει μέχρι στιγμής. Για όσες επαναλήψεις έχω εκτελώ την διαδικασία της εύρεσης γονέων που με την σειρά τους βρίσουν το κόστος της κάθε διαδρομής. Τους βάζω σε έναν πίνακα, και συνενώνω το σκορ στην τρέχον διαδρομή(όπου δείχνει τα βήματα που έκανε ο αλγόριθμος στον υπολογισμό των σκορ των ακολουθιών των πόλεων) αφού πλέον έχει πάρει τιμή από την συνάρτηση choose\_parents. Εκτυπώνω την τρέχων γενιά και το σκορ. Στη συνέχεια από την συνάρτηση choose\_parents επιστράφηκε το index για να βρω από τις υποψήφιες λύσεις μου την συγκεκριμένη διαδρομή οπότε αφού είχε το καλύτερο σκορ το βάζω στην μεταβλητή current\_best για να το εκτυπώσω και ενημερώνω τον μετρητή score για να λάβει την νέα μικρότερη τιμή. Στην συνέχεια καλώ την συνάρτηση με την οποία θα αναπαραχθεί ο πληθυσμός. Ελέγχω αν έχω περισσότερα από 10 παιδιά και σβήνω τα παλαιότερα από την λίστα. Και καλώ ξανά τον πληθυσμό με όρισμα τα παιδιά τώρα και όχι τον αρχικό τυχαίο πληθυσμό. Τέλος ελέγχω αν το σκορ(θα μπορούσα και το current best) είναι αρκετά καλή λύση για να σταματήσει ο αλγόριθμος στην τρέχον γενιά. Ο αλγόριθμος επιστρέφει το σκορ και την διαδρομή.**

****

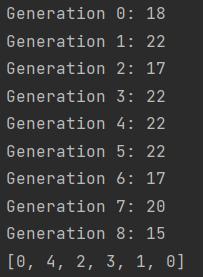
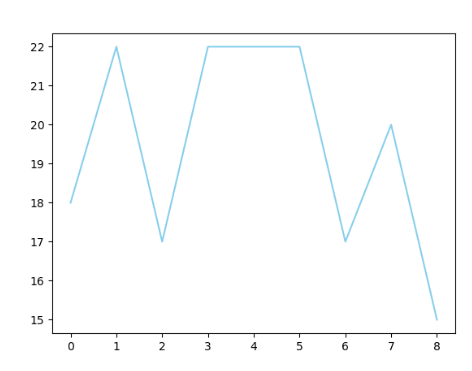
**Ώρα να εκτελέσουμε τον γενετικό αλγόριθμο με αρχικά ορίσματα τον πίνακα των πόλεων και τα κόστη των διαδρομών, θα μου επιστρέψει την καλύτερη διαδρομή των πόλεων, και το σκορ από τις συνολικές πόλεις που εξέτασε για να αναπαρασταθεί γραφικά. Το καλύτερο σκορ θα εκτυπωθεί στην κονσόλα. Το γράφημα έχει ως άξονα των Χ όσες το πλήθος των σκορ(όσες και οι γενιές) και Υ την ακολουθία των σκορ. Κι τέλος δείχνω την καλύτερη διαδρομή συμπεριλαμβανόμενη και της πόλης Α.**

**Εκτέλεση Κώδικα**

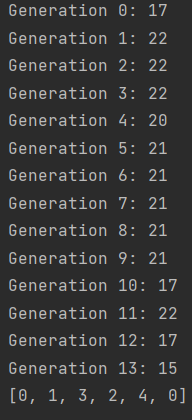
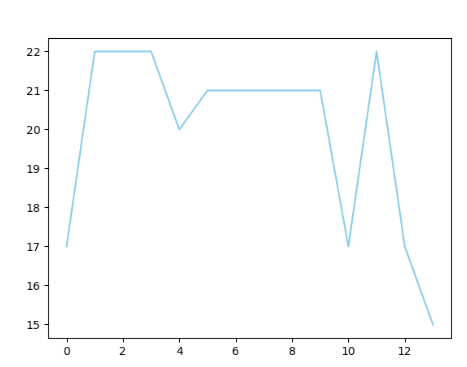
**Μίας σχετικά όχι τόσο καλής λύσης**

**** ****

Μίας αρκετά καλής λύσης σε λιγότερες γενιές



Και μίας αλλης αρκετά καλής λύσης

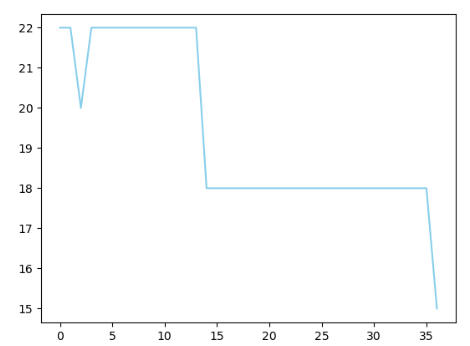


Παρατηρώ ότι ο αλγόριθμος μπορεί να αποκλείνει σε κάποια σημεία διότι δεν επιλέγω κάθε φορά τους καλύτερους γονείς με το μικρότερο σκορ, μπορεί να επιλεχθούν γονείς στην choose parents που να μην έχουν καλά γονίδια, ή στην διαδικασία της αναπαραγωγής να επιλεχθούν 2 γονείς τυχαία κι ο ένας ή και οι δύο που να μην βελτιώσουν το σκορ. Ή να γίνει κάποια μετάλαξη που να αλλιώσει αρνητικά ένα γονίδιο. Ή παρόλο που πήγαινε καλά ο αλγόριθμος να έμενε ένας γονέας με μη βέλτιστα γονίδια στην λίστα των γονιών και να επιλέχθηκε σε μεταγενέστερη γενία αν δεν ήταν αρκετά παλιός να διαγραφεί ή αρκετά άτυχος ώστε να μην επιλεχθεί ως γονέας και να μην περάσει στην μερική ανανέωση πληθυσμόυ.

Επίσης επειδή είναι σπάνιο θα μπορούσαμε να δούμε πότε έγινε μία μεταλλαξη.



Για να μην αποκλείνει τόσο ο αλγόριθμος και χρειαστει να τον ξανατρέξουμε μπορούμε να αυξήσουμε τις γενιές

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Σε βάθος χρόνου θα βρίσκει σχεδόν πάντα καλό αποτέλεσμα αν αυξήσουμε αρκετά τις γενιές πχ χίλιες.

