# ΕΚΤΕΛΕΣΗ ΑΛΓΟΡΙΘΜΟΥ

## Βήμα 1ο :

Δημιουργία Κ άδειων δρομολογίων. Με αρχικοποίηση της πρώτης και τελευταίας θέσης με τα στοιχεία του ξενοδοχείου(Υλοποίηση μέσω λίστας).

Ορισμός των #επαν επαναλήψεων έτσι ώστε να γνωρίζει ο αλγόριθμος τις επαναλήψεις που θα χρειαστεί να εκτελέσει.

## Βήμα 2ο :

Έναρξη εισαγωγής κόμβων δηλαδή αξιοθέατων στα δρομολόγια.

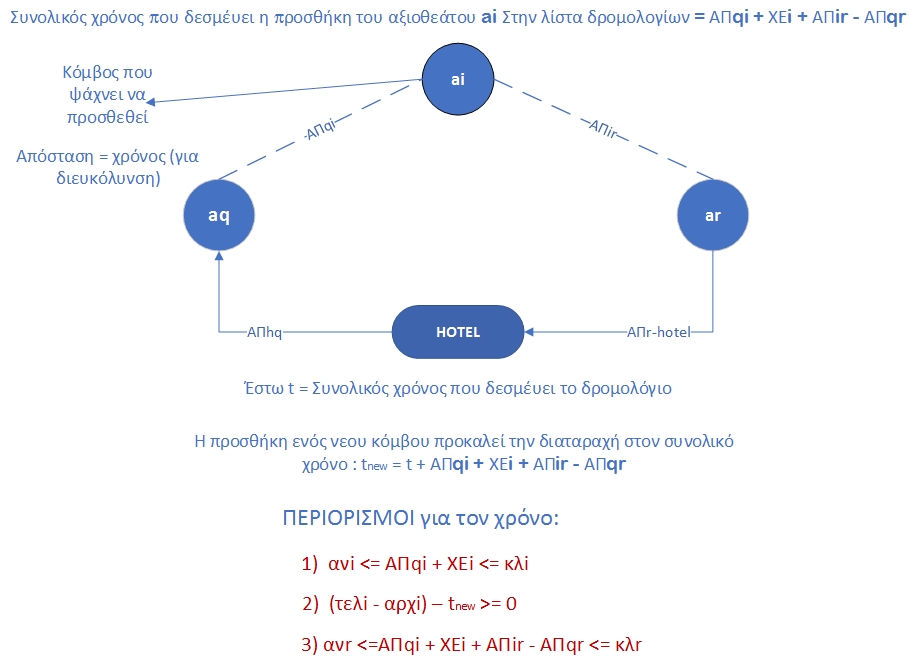
ΕΛΕΓΧΟΣ :

1) Οι νέες προσθήκες δεν ακυρώνουν ήδη υπάρχων κόμβους

των δρομολογίων. Ο έλεγχος για το αν ένα αξιοθέατο έχει

την δυνατότητα να προστεθεί στην λίστα πραγματοποιείται

σύμφωνα με το παρακάτω διάγραμμα.



Ο αλγόριθμος πραγματοποιεί τον παραπάνω έλεγχο για όλες τις υπάρχων διαδρομές και δημιουργεί μια δομή που εκφράζει όλες τις πιθανές διαδρομές στις οποίες μπορεί να εισαχθεί. Πιθανή δομή αποθήκευσης των πιθανών διαδρομών αποτελεί μια Linked List.

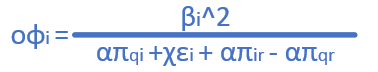
## Βήμα 3ο :

Αφού ο αλγόριθμος έχει "μπροστά" του όλες τις πιθανές διαδρομές στις οποίες μπορεί να εκχωρήσει αυτό το νέο κόμβο. Χρειάζεται να πραγματοποιήσει την παρακάτω διαδικασία.

ΕΛΕΓΧΟΣ :

1) Το βασικό κριτήριο που καθορίζει την επιλογή ενός αξιοθέατου *αi* ως το επόμενο που θα εισαχθεί σε κάποιο δρομολόγιο είναι το όφελος που προκύπτει από την εισαγωγή του στο δρομολόγιο αυτό.

Η συνάρτηση που παριστά το όφελος της προσθήκης ενός κόμβου σε μία από τις διαδρομές όπου το επιτρέπουν είναι :



Ο παρανομαστής του παραπάνω κλάσματος παριστά την διαταραχή η μάλλον αύξηση που προκαλεί η προσθήκη του κόμβου αi στο δρομολόγιο

Ο αλγόριθμος πρέπει να ελέγξει το κάθε δρομολόγιο στο οποίο έχει πρόσβαση και να υπολογίσει το όφελος που προκύπτει άμα προστεθεί ο παραπάνω κόμβος. Ο αλγόριθμος θα πρέπει να προσθέσει το όφελος που υπολόγισε για κάθε διαδρομή και να το προσθέσει σε μια κατάλληλη δομή

Στη συνέχεια θα ψάξει και θα βρει το μεγαλύτερο όφελος από την δομή των οφελών και τελικά θα προσθέσει τον κόμβο στην αντίστοιχη λίστα που υπολόγισε ότι θα παρουσιαστεί το μεγαλύτερο όφελος .

Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι να γεμίσουν όλες οι διαδρομές.

## Βήμα 4ο :

Πρόβλημα εγκλωβισμού σε τοπικό μέγιστο :

Ο αλγόριθμος μας ενδέχεται να εγκλωβιστεί σε μία τοπικά μέγιστη λύση. Για να εξηγήσουμε αυτό το φαινόμενο θα χρησιμοποιήσουμε ένα παράδειγμα.

Έστω ότι υπήρχαν 100 πιθανά αξιοθέατα που μπορεί να επισκεφτεί ο τουρίστας. Αλλά ο τουρίστας χωρούσε μόνο 50 επισκέψεις συνολικά στο πρόγραμμά του. Έστω ότι τα 50 πρώτα αξιοθέατα τα οποία θα επεξεργαζόταν ο αλγόριθμος είχαν χαμηλότερες βαθμολογίες από τα υπόλοιπα 50 που βρίσκονται στο αρχείο. Οπότε όταν ο αλγόριθμος εκτελούσε τις πρώτες 50 επαναλήψεις θα είχε γεμίσει όλες τις διαδρομές με αξιοθέατα(εφ όσων αυτό ήταν εφικτό σύμφωνα με τους περιορισμούς). Ο αλγόριθμος θα νόμιζε πως η λύση που βρήκε είναι η βέλτιστη παρ' όλα αυτά αυτό που έχει επιτύχει είναι το να εγκλωβιστεί σε μια τοπικά "καλύτερη" λύση .

Για να αποφύγουμε το παραπάνω σενάριο έχουμε την δυνατότητα να "βάλουμε"

ως τελευταίο βήμα στον αλγόριθμό μας να πραγματοποιεί έναν αριθμό από διαγραφές στα ήδη υπάρχων δρομολόγια. Πιο συγκεκριμένα από κάθε δρομολόγιο επιλέγουμε τυχαία ένα αξιοθέατο και π είναι μια τυχαίως γενόμενη τιμή ( οι τιμές που μπορεί να πάρει το π είναι 0 ή 1). Και με αρχή το αξιοθέατο που έχει επιλεχθεί αρχίζουμε και διαγράφουμε π\*(πλήθος των αξιοθέατων) αξιοθέατα με κατεύθυνση προς το ξενοδοχείο. Αν τα αξιοθέατα που πρέπει να διαγραφθούν είναι περισσότερα από αυτά που μεσολαβούν μέχρι το ξενοδοχείο, η διαγραφή συνεχίζει και στα αξιοθέατα που βρίσκονται μετά το ξενοδοχείο. Άρα με λίγα λόγια δεν λαμβάνουμε υπόψη μας τον κόμβο του ξενοδοχείου.

Μετά τις διαγραφές ο αλγόριθμος μας προχωρά στην επόμενη επανάληψη.