ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ Η.Μ.Μ.Υ
ΕΡΓΑΣΤΗΡΙΟ ΕΥΦΥΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ – ΠΛΗ 417

ΑΝΑΦΟΡΑ ΥΛΟΠΟΙΗΣΗΣ ΕΡΓΑΣΙΑΣ 1

ΜΕΡΟΣ Α - "ΤΕΧΝΙΚΕΣ ΑΝΑΖΗΤΗΣΗΣ"

Σπουδαστές:

Τρίμας Χρήστος **2016030054**

Παντελής Κωνσταντίνος **2015030070**

Σκοπός Εργαστηριακής Άσκησης:

Πειραματισμός με τεχνικές αναζήτησης σε δωσμένο Grid – Maze – Κόσμο Αναζήτησης, με παρουσία ενδεχόμενων ποινών προσπέλασης. Το πρώτο μέρος υλοποιήθηκε σε Java (Java SE 13.0) ενώ το δεύτερο, η αναφορά του οποίου ακολουθεί σε ξεχωριστό .pdf σε Python (python3). Τόσο για το compilation όσο και για το run αρκούν οποιοδήποτε συμβατοί με τις δύο εκδόσεις γλωσσών compiler.

Ζητούμενες Τεχνικές Αναζήτησης:

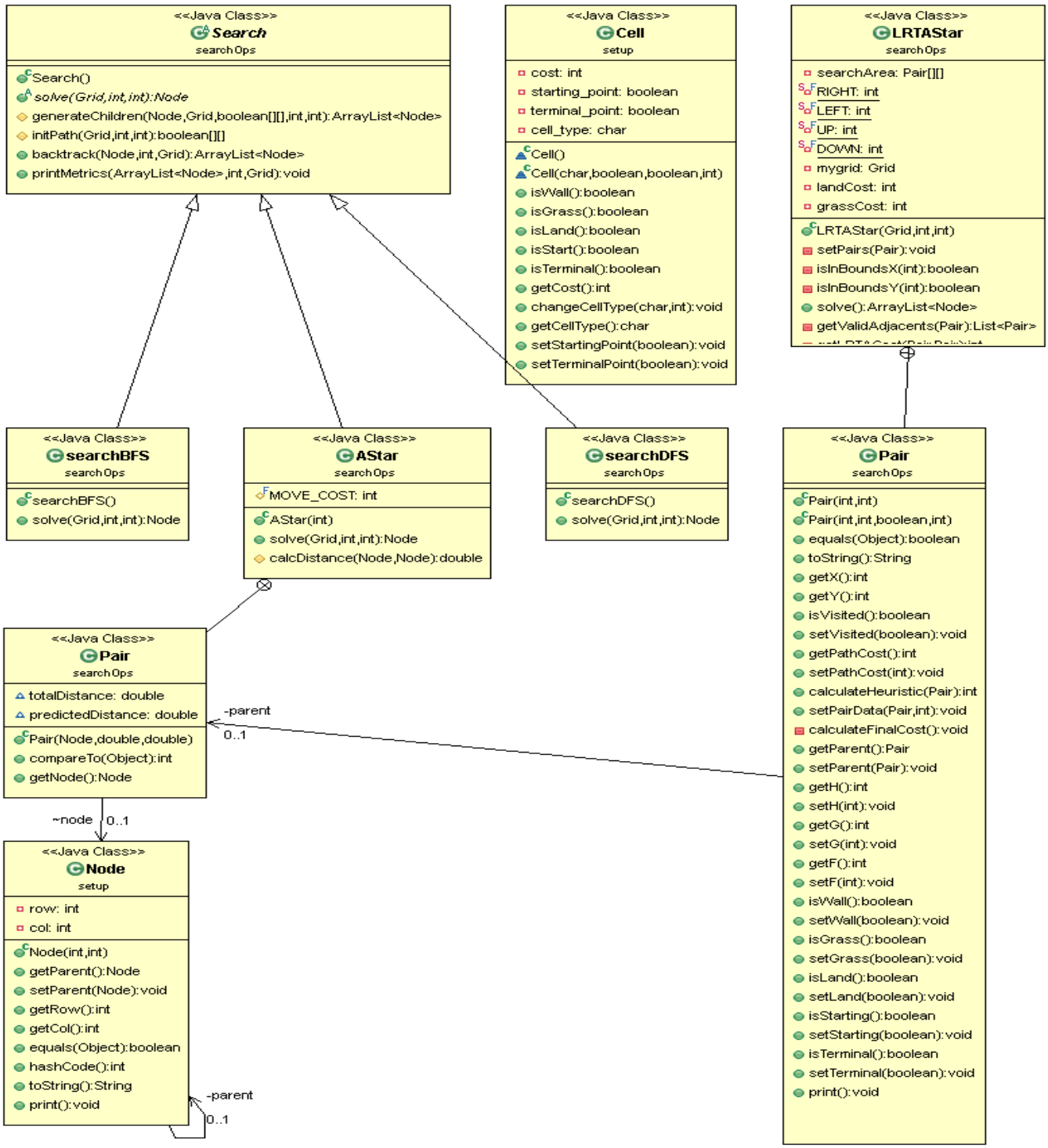
- BFS
- DFS
- A*
- LRTA* (παραλλαγή A*)

Υλοποιήθηκε επιπλέον ένα console based menu, καθώς και γραφικοποίηση των μονοπατιών που ευρέθηκαν μέσω μεταβολής δοθέντος κώδικα. Οι εικόνες και τα πραδείγματα βασίζονται στον ***Hard_C.world***.

Δόμηση Κώδικα:

Ο κώδικας δομείται σε δύο πακέτα το ***setup*** και το ***searchOps***. Στο πρώτο κανείς θα βρεί όλο τον δοθέντα κώδικα με κάποιες τροποποιήσεις, ενώ στο δεύτερο ακολουθούν οι υλοποιήσεις των μεθόδων αναζήτησης.

Αναφορικά με τις τελευταίες ακολουθεί το UML διάγραμμά τους.



BFS (Breadth First Search):

Όπως έγινε διακριτό και απο την σειρά θεωρητικών ασκήσεων που προηγήθηκε, η απληροφόρητη αναζήτηση κατά πλάτος (BFS) δέχεται ως είσοδο τον κόμβο εκκίνησης και αναζητά διαδοχικά τους γείτονες ανα “επίπεδο” μέχρι να ευρεθεί ο κόμβος-στόχος.

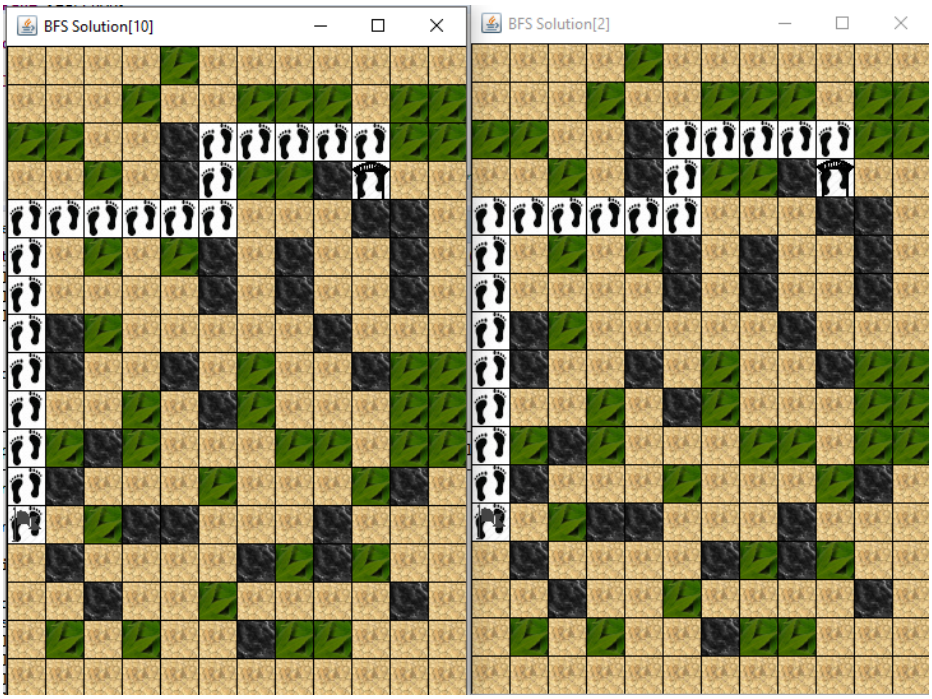
Η τιμή επιστροφής της αναζήτησης είναι το κόστος του συντομότερου μονοπατιού από την εκκίνηση ως το στόχο(τεχνικά αυτό συμβαίνει με δημιουργία σχέσων πατέρα – παιδιού σε κόμβους γεννήτορες και μία ουρά).

Για κάθε κόμβο, αναζητούμε τους πιθανούς γείτονες, δηλαδή τους κόμβους που υπάρχουν πάνω στο δοθέν grid , με μοναδικό περιορισμό να μην αντιστοιχούν σε τείχος.

Όλοι οι έγκυροι γείτονες εισάγονται μετέπειτα στην ουρά ώς δυνητικά προσπελάσιμοι. Επαναληπτικά προσπελαύνεται ο πρώτος κόμβος ουράς, εκτελώντας την προηγούμενη διαδικασία μέχρι τον τερματισμό – εύρεση στόχου.

Το ελάχιστο κόστος αναζήτησης δεν ισοδυναμεί πάντα με το συντομότερο μονοπάτι, καθώς η συγκεκριμένη αναζήτηση μας εγγυάται μεν ότι θα βρει σε κάθε περίπτωση το στόχο, δεν ελέγχει όμως το συντομότερο μονοπάτι (με κάποια ευρετική).

Αξιωσημείωτο εδώ οτι ο αλγόριθμος δεν επιλέγει κατεύθυνση κίνησης με βάση κάποια ευρετική συνάρτηση αλλά μόνο με βάση την ύπαρξη η μή επιτρεπτών γειτόνων απογόνων.




```
----------------------------------INFO:---------------------------------------------

Current path has: 9 Grass nodes 12 Land Nodes. Traverse Penalty: 30 [Grass Cost = 2]

Current path has: 9 Grass nodes 12 Land Nodes. Traverse Penalty: 102 [Grass Cost = 10]
```

Η παραπάνω μέθοδος θεωρείται αποδοτική (σε σύγκριση με την DFS) αλλά όχι βέλτιστη και σίγουρα όχι η προτιμότερη λόγω του τρόπου προσπέλασης καθότι δεν επιλέγει, παρά γεννά και προσπελαύνει με έναν όχι και τόσο “ευφυή τρόπο”.

DFS (Depth First Search):

Η απληροφόρητη αναζήτηση κατά βάθος (***DFS***) δέχεται ως είσοδο τον κόμβο εκκίνησης και αναζητά τους απογόνους προς την ίδια κατεύθυνση μέχρι να βρει τον κόμβο στόχο ή να μη μπορεί να συνεχίσει άλλο προς αυτήν (τέλος πίνακα ή κόμβος τείχος).

Όταν δεν υπάρχει άλλος κόμβος - διάδοχος προς την ίδια κατεύθυνση ακολουθείται η ίδια διαδικασία προς την επόμενη δοθείσα - επιθυμητή κατεύθυνση (τεχνικά αυτό συμβαίνει με σχέσεις πατέρα παιδιού και μία στοίβα).

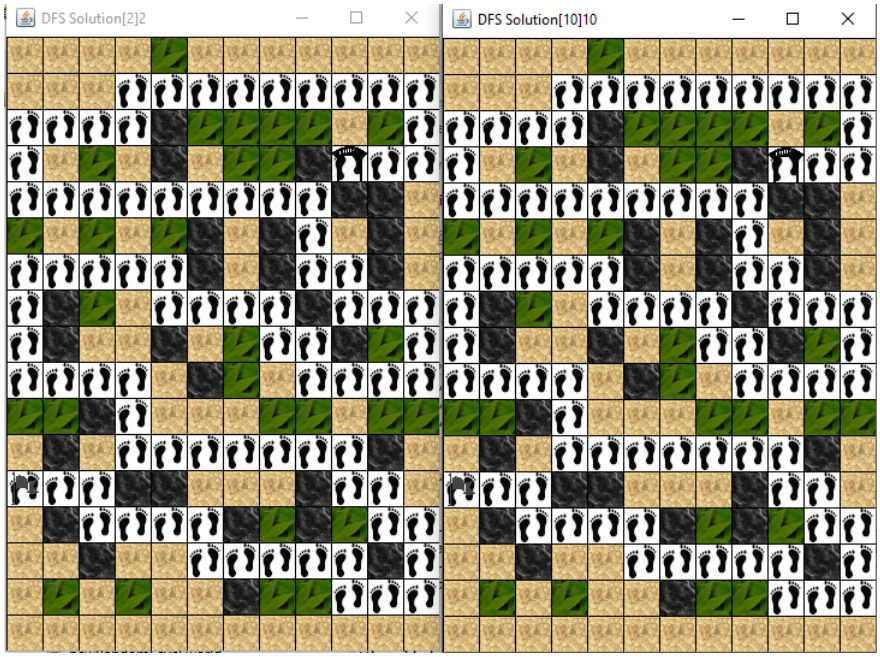
Δεδομένης επίσκεψης σε κόμβο, ο κόμβος προστίθεται στη στοίβα. Η χρησιμότητα της στοίβας είναι κομβική στην περίπτωση που ο αλγόριθμος εγκλωβιστεί και δε μπορεί να βρει έγκυρο κόμβο γείτονα. Τότε με χρήση αυτής, οπισθοδρομεί, ακολουθώντας ουσιαστικά τη διαδρομή από την οποία ήρθε και σε κάθε βήμα προς τα πίσω ψάχνει για έγκυρο κόμβο γείτονα.

Το κόστος της αναζήτησης προκύπτει από τα κόστη των κόμβων που προσπελάστηκαν αφαιρώντας τα μη έγκυρα μονοπάτια, μονοπάτια εγκλωβισμού στα οποία ο αλγόριθμος οπισθοδρόμησε.

Σε θεωρητικά πλαίσια ο συγκεκριμένος αλγόριθμος δεν εγγυάται αποτέλεσμα λόγω και χρονικής πολυπλοκότητας (ιδίως στην αναδρομική του εκδοχή – high function call overhead), η ύπαρξη όμως πινάκων με ορισμένες διαστάσεις τον βοηθά να είναι λειτουργικός στην παρούσα σε συνδιασμό πάντα και με μή αναδρομική προσέγγιση.

Σε πολλές περιπτώσεις το κόστος προσπέλασης είναι δημαντικά μεγαλύτερο από τους προς σύγκριση αλγορίθμους (***BFS, A*, LRTA****), το οποίο οφείλεται στον τρόπο προσπέλασης και γέννησης απογόνων (***UP, DOWN, LEFT, RIGHT).***

Ενδεχόμενη τροποποίηση της σειράς γέννησης ίσως να βελτίωνε την απόδοση αλλά και πάλι υπάρχει υψηλή αβεβαιότητα σχετικά με αυτό.




```
-----------------------------------INFO:-------------------------------------------------
Current path has: 21 Grass nodes 62 Land Nodes. Traverse Penalty: 104 [Grass Cost = 2]
Current path has: 21 Grass nodes 62 Land Nodes. Traverse Penalty: 272 [Grass Cost = 10]
```

A* :

Η πληροφορημένη αναζήτηση πρώτα στο καλύτερο ή αλλιώς A* αποτελεί παραλλαγή του αλγορίθμου του Djikstra δεχόμενη ως εισόδους τον κόμβο αφετηρίας και τον κόμβο στόχο χρησιμοποιώντας χρησιμοποιεί δύο συναρτήσεις ευρετικής μορφής προκειμένου να βελτιστοποιεί το μονοπάτι.

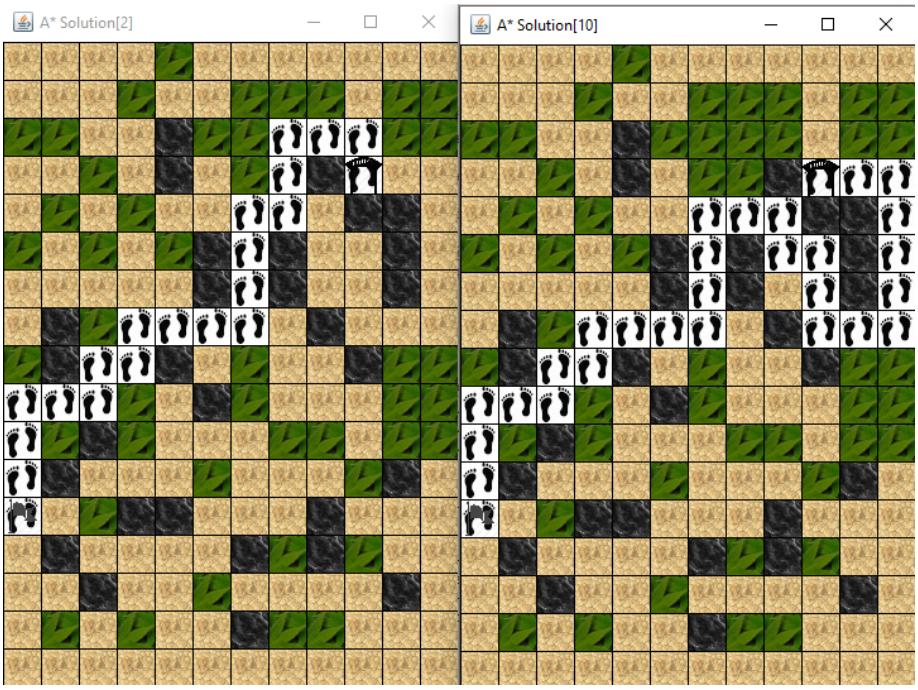
$$f(n)=h(n)+g(n)$$

Η μία εξ αυτών, ***g(n) υπολογίζει το κόστος από την αφετηρία μέχρι τον κόμβο n***, ενώ η άλλη, ***h(n) εκτιμά το κόστος από τον κόμβο n προς τον τερματικό κόμβο***.

Για κάθε κόμβο ελέγχουμε ψάχνοντας έγκυρους γείτονες και επιπλέον αξιολογούμε αν το μονοπάτι που μας οδήγησε στο συγκεκριμένο κόμβο είναι το βέλτιστο. Αν δεν είναι το αναπροσαρμόζουμε με χρήση ουράς προτεραιότητας στην οποία αποθηκεύονται οι προς επίσκεψη κόβμβοι. Η διαδικασία είναι επαναληπτική μέχρι την εύρεση του κόμβου στόχου.

Ο συγκεκριμένος αλγόριθμος επιστρέφει σε κάθε περίπτωση το βέλτιστο μονοπάτι αρκεί η ευρετική h(n) να είναι παραδεκτή (χρήζει προσοχής τυχών υπερεκτίμηση της χρησιμοποιούμενης απόστασης που θα οδηγήσει σε μη προσπέλαση πιθανώς βέλτιστου μονοπατιού).

Για μεγάλο κόστος (***grassCost = 10***) παρατηρείται διαφοροποίηση στο μονοπάτι επίλυσης, γεγονός που οφείλεται στην ευρευτική που χρησιμοποιήθηκε (απόσταση Manhattan σταθμισμένη με κόστος προσπέλασης).



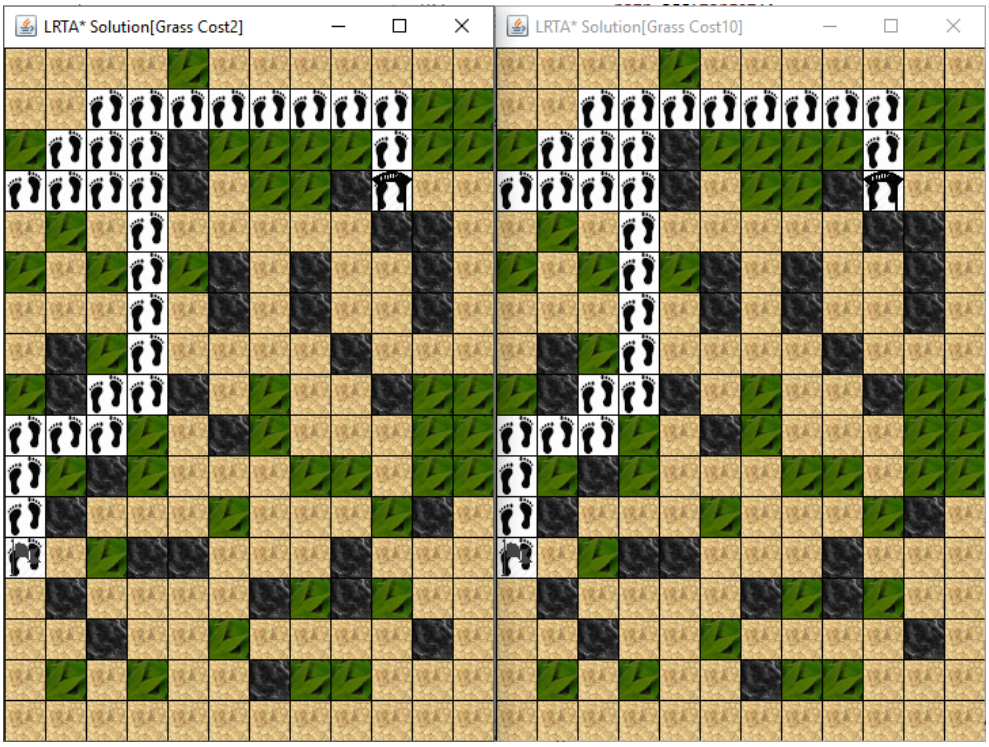

```
-----------------------------------INFO:-------------------------------------------------

Current path has: 4 Grass nodes 17 Land Nodes. Traverse Penalty: 25 [Grass Cost = 2]
Current path has: 1 Grass nodes 28 Land Nodes. Traverse Penalty: 38 [Grass Cost = 10]
```

Εν γένει ο παραπάνω αποτελεί τον πιο αποδοτικό ,σε όρους κόστους, αλγόριθμο, γεγονός που οφείλεται και στην κατασκευή της ευρετικής αυτού η οποία επιλέχθηκε να σταθμίζει την απόσταση Manhattan με το βάρος του κόστους προσπέλασης μονοπατιού. Με αυτόν τον τρόπο επιτυγχάνεται βελτιστοποίηση κόστους διαδρομής για δεδομένη λύση και εξασφαλίζεται η παραδοχή της αυτής ευρετικής αποφεύγοντας τυχούσα υπερεκτίμηση.

LRTA* :

Η πληροφορημένη online αναζήτηση πρώτα στο καλύτερο (LRTA*) είναι όμοια στην υλοποίηση αλλά και αλγοριθμικά με την προηγούμενη (A*) με τη βασική διαφορά ότι ο αλγόριθμος γνωρίζει κάθε φορά μόνο τα κόστη των γειτονικών μονοπατιών - κόμβων, από τον κόμβο που βρίσκεται.

Αυτό οδηγεί στο συμπέρασμα παρότι συμβαίνει επιλογή μονοπατιού με βάση κριτήριο εύρεσης δεν υπάρχει δυνατότητα βελτίωσης λόγω της online φύσης του, σε αντίθεση με τον A*. Εδώ δεν αναμένεται αλλοίωση στην διαδρομή συμμετρική της αύξησης κόστους.




```
----------------------------------INFO:-----------------------------------------------

Current path has: 8 Grass nodes 23 Land Nodes. Traverse Penalty: 39 [Grass Cost = 2]
Current path has: 8 Grass nodes 23 Land Nodes. Traverse Penalty: 103 [Grass Cost = 10]
```