Αναζήτηση σε Λαβύρινθο με UCS και A* – Άσκηση 1

Η Άσκηση 1 αφορά την εύρεση της βέλτιστης διαδρομής μέσα σε έναν τυχαία παραγόμενο λαβύρινθο μεγέθους N×N, χρησιμοποιώντας τους αλγορίθμους αναζήτησης UCS (Uniform Cost Search) και A*. Ο λαβύρινθος αναπαρίσταται ως πίνακας δισδιάστατου τύπου `char[][]`, όπου κάθε κελί μπορεί να είναι είτε κενό (`'.'`), είτε εμπόδιο (`'#'`), είτε ειδικό σύμβολο όπως το Start (`'S'`), Goal (`'G'`), ή τηλεμεταφορείς (`'A'`, `'B'`).

Η κατάσταση του προβλήματος ορίζεται από το περιεχόμενο του πίνακα, την τρέχουσα θέση, το κόστος μετακίνησης και τις διαθέσιμες κινήσεις (οριζόντιες, κάθετες, διαγώνιες, και μέσω τηλεμεταφοράς). Ο χρήστης εισάγει τις θέσεις αρχής (S) και στόχου (G), καθώς και την πιθανότητα εμποδίου p. Κάθε μετακίνηση έχει κόστος 1, ενώ οι τηλεμεταφορές έχουν κόστος 2.

Ο UCS αναζητά τη διαδρομή με το ελάχιστο συνολικό κόστος χωρίς χρήση ευρετικής. Ο A*, αντίθετα, βασίζεται σε ευρετική εκτίμηση για να επιταχύνει τη διαδικασία εύρεσης λύσης. Η ευρετική που επιλέχθηκε είναι η **Chebyshev Distance** :

h(n)=max⁡(|xgoal−x|, |ygoal−y|)h(n) = \max(|x_{goal} - x|,\ |y_{goal} - y|)h(n)=max(|xgoal−x|, |ygoal−y|)

Η Chebyshev Distance μετρά το ελάχιστο πλήθος βημάτων που απαιτούνται για να φτάσει κάποιος από το σημείο `n` στον στόχο, όταν επιτρέπονται κινήσεις προς όλες τις 8 κατευθύνσεις (όπως στο πρόβλημα μας), και κάθε κίνηση έχει ίσο κόστος. Επομένως, αποτυπώνει με ακρίβεια την ιδανική απόσταση σε ένα τέτοιο περιβάλλον.

Αυτή η ευρετική είναι **αποδεκτή (admissible)** γιατί **ποτέ δεν υπερεκτιμά** το πραγματικό κόστος για να φτάσουμε από τον κόμβο `n` στο στόχο. Συγκεκριμένα, η Chebyshev Distance υποθέτει ότι ο πράκτορας κινείται ευθεία προς τον στόχο χωρίς εμπόδια, άρα επιστρέφει μια τιμή που είναι είτε ίση είτε μικρότερη από την πραγματική απόσταση με εμπόδια. Αυτό ικανοποιεί τον ορισμό της αποδεκτότητας, που απαιτεί:

h(n)≤h∗(n)h(n) \leq h^*(n)h(n)≤h∗(n)

όπου h∗(n)h^*(n)h∗(n) είναι το αληθινό κόστος διαδρομής από το `n` ως το `goal`.

Επιπλέον, η ευρετική είναι και **συνεπής (consistent)**, αφού για κάθε μετάβαση από έναν κόμβο `n` σε έναν γείτονά του `n'` ισχύει:

h(n)≤c(n,n′)+h(n′)h(n) \leq c(n, n') + h(n')h(n)≤c(n,n′)+h(n′)

με c(n,n′)=1c(n, n') = 1c(n,n′)=1, λόγω ίσου κόστους μεταξύ γειτονικών κελιών. Αυτό εξασφαλίζει ότι ο A* δεν επανεξετάζει κόμβους, κάνοντάς τον πιο αποδοτικό.

Δεν επιλέχθηκε η **Manhattan Distance** επειδή μετρά την απόσταση υποθέτοντας μόνο οριζόντια και κάθετη μετακίνηση. Σε προβλήματα όπου επιτρέπεται διαγώνια κίνηση (όπως εδώ), η Manhattan υποεκτιμά την προσβασιμότητα, οδηγώντας σε αναποτελεσματική αναζήτηση. Αντίστοιχα, η **Euclidean Distance** (ευκλείδεια απόσταση) βασίζεται σε συνεχή χώρο και δεν ταιριάζει σε πλέγματα με διακριτές κινήσεις και σταθερό κόστος. Η Chebyshev, αντίθετα, είναι η πιο κατάλληλη ευρετική για τετράγωνα πλέγματα με διαγώνια κίνηση ίδιου κόστους.

Η υλοποίηση περιλαμβάνει κλάσεις `Cell`, `Node` και `NodeAStar`, που αποθηκεύουν τις πληροφορίες κάθε κατάστασης (συντεταγμένες, κόστος, πατέρας, κ.ά.). Οι μέθοδοι `solveUCS(...)` και `solveAStar(...)` χρησιμοποιούν δομές PriorityQueue για να διαχειρίζονται τους κόμβους προς επέκταση, επιστρέφοντας τη βέλτιστη διαδρομή ή null αν δεν υπάρχει λύση.

Στην έξοδο του προγράμματος εμφανίζεται ο αρχικός λαβύρινθος, η τελική διαδρομή με σύμβολο `'*'`, το συνολικό κόστος, και ο αριθμός επεκτάσεων κόμβων για κάθε αλγόριθμο. Από την πειραματική παρατήρηση, φαίνεται ότι ο A* βρίσκει τη λύση επεκτείνοντας πολύ λιγότερους κόμβους από τον UCS, χάρη στη χρήση της ευρετικής, διατηρώντας όμως τη βέλτιστη διαδρομή.

Η επιλογή της Chebyshev ως ευρετικής ήταν καθοριστική, καθώς ανταποκρίνεται άριστα στα χαρακτηριστικά του προβλήματος, οδηγώντας σε αποτελεσματική και ορθά κατευθυνόμενη αναζήτηση, με σημαντική μείωση κόστους υπολογισμού συγκριτικά με τον UCS, χωρίς να θυσιάζει την ορθότητα ή τη βέλτιστη λύση.

Παραδείγματα του κώδικα:

Παράδειγμα 1:

Δώσε μέγεθος πλέγματος N:

5

Δώσε πιθανότητα εμποδίου p (0,0 - 1,0): 0.3

Δώσε x του αρχικού κελιού S: 0

Δώσε y του αρχικού κελιού S: 1

Δώσε x του τελικού κελιού G: 4

Δώσε y του τελικού κελιού G: 2

Λαβύρινθος:

. S # # B

. . . . .

. . # # .

. . . . .

A # G # .

Βρέθηκε διαδρομή με UCS και κόστος: 4

Επεκτάθηκαν 26 κόμβοι με UCS.

. S # # B

* . . . .

* . # # .

. * . . .

A # G # .

Βρέθηκε διαδρομή με A* και κόστος: 4

Επεκτάθηκαν 9 κόμβοι με A*.

. S # # B

* . . . .

. * # # .

. * . . .

A # G # .

Παράδειγμα 2:

Δώσε μέγεθος πλέγματος N: 10

Δώσε πιθανότητα εμποδίου p (0,0 - 1,0): 0.4

Δώσε x του αρχικού κελιού S: 0

Δώσε y του αρχικού κελιού S: 3

Δώσε x του τελικού κελιού G: 4

Δώσε y του τελικού κελιού G: 8

Λαβύρινθος:

. S . # # . . B

. . . # . . .

. . . # . . . . . #

. . . # # # . #

. # # . . . . . G .

. . . # . # .

. . # # # . . .

. . # . # # . # # #

. . . . # # .

A . . . . . # . # .

Βρέθηκε διαδρομή με UCS και κόστος: 5

Επεκτάθηκαν 60 κόμβοι με UCS.

. S . # # . . B

. . * # . . .

. . . # . * * . . #

. . . # # # * #

. # # . . . . . G .

. . . # . # .

. . # # # . . .

. . # . # # . # # #

. . . . # # .

A . . . . . # . # .

Βρέθηκε διαδρομή με A* και κόστος: 5

Επεκτάθηκαν 7 κόμβοι με A*.

. S . # # . . B

. . * # . . .

. . . # . * * . . #

. . . # # # * #

. # # . . . . . G .

. . . # . # .

. . # # # . . .

. . # . # # . # # #

. . . . # # .

A . . . . . # . # .

Συμπέρασμα

Εκτός από τα επιθυμητά αποτελέσματα των παραπάνω παραδειγμάτων ,για την λειτουργία του **UCS** και **A* ,**παρατηρούμε και μεγάλη διαφορά στον αριθμό των επεκτάσεων κόμβων μεταξύ των **UCS** και **A*.**Αυτό είναι λογικό (και θεμιτό) διότι η **UCS** δεν χρησιμοποιεί ευρετική ,οπότε ψάχνει στα τυφλά χωρίς να ξέρει που είναι ο στόχος(ψάνει σε όλες τις κατευθύνσεις) με μοναδικό γνώμονα μόνο το κόστος.Ενώ η **A*** χρησιμοποιεί την **ευρετική Chebyshev Distance** που την κατευθύνει στον σωστό στόχο, ελαχιστοποιώντας τις επεκτάσεις κόμβων και αυτός ήταν ο στόχος της ασκησής τον οποίο ολοκληρώσαμε επιτυχώς**.**