pacman readMe sdi1900085

Pacman Project 1: Search

Q1: DFS
Για την υλοποίηση του DFS κρατάω σε ένα tuple την τωρινη θεση και μια λιστα κινησεων που θα επιστρεψει η συναρτηση. Κρατάω σε μια λίστα τους κόμβους που εχουν ήδη επισκεφθεί, σε stack(λόγω LIFO) τους κόμβους που είναι προς επίσκεψη και, ξεκινόντας απ τον αρχικό κόμβο, αρχικά ελέγχω άμα ειναι ο στόχος και αν είναι τερματίζω την επανάληψη και επιστρέφω την λίστα κινήσεων. Έπειτα εάν δεν είναι στη λίστα visited, μπαίνει και τέλος άμα τα παιδιά του κόμβου δεν έχουν επισκεφθεί μπαίνουν στο stack και αφού γίνει pop απ το stack το επόμενο tuple επαναλαμβάνονται τα παραπάνω βήματα μέχρι να βρεθεί ο στόχος

Q2: BFS
Για τον BFS ακολούθησα τα ίδια ακριβώς βήματα με τον dfs με τη διαφορά οτι αντί για stack αποθηκεύω τα tuples σε queue λόγω FIFO για να λειτουργήσει ως BFS

Q3: UCS
Τα ίδια ακριβώς ισχύουν και για τον UCS μόνο που τώρα γίνεται χρήση priority queue στο οποίο αντί για tuples βάζω triplets καθώς προσθέτω και το κόστος το οποίο είναι και το priority κριτήριο στην priority queue

Q4: A*
Ίδιο με τον UCS μόνο που προστίθεται και ο παράγοντας της ευρετικής συνάρτησης ο οποίος συμπεριλαμβάνεται ως priority factor συνδιαστικά με το κόστος

Q5: Corners Problem
Αρχικά στην init πρόσθεσα μια λίστα που θα κρατάει σαν στοιχείο της κλάσης τις γωνίες που έχουν ήδη επισκεφθεί. Το αρχικό state που επιστρέφεται είναι ένα tuple που περιέχει την αρχική θέση και την λίστα που θα αποθηκεύονται οι γωνίες που θα επισκέπτονται.
Εάν έχουν επισκεφθεί και οι 4 γωνίες τότε έχουμε φτάσει στην επιθυμητή κατάσταση. Έπειτα για να βρούμε τα παιδιά βρίσκουμε την κάθε επόμενη θέση που δεν είναι τύχος γύρω απ την τωρινή θέση, ελένχουμε κιόλας μήπως πρόκειται για γωνία που στην περίπτωση αυτή ενημερώνουμε την λίστα με τις επισκεφθείς γωνίες στο συγκεκριμένο successor state και μετά προσθέτουμε το state αυτό στην λίστα με τους successors που θα επιστρέψει η συνάρτηση.

Q6: Corners Heuristic
Στην ευρετική για όλες τις γωνίες αρχικά κρατάω μια λίστα με τις γωνίες που δεν έχουν επισκεφθεί και μια μεταβλητή distance η οποία στην αρχή είναι 0. Άμα έχουν επισκεφθεί και οι 4 γωνίες τότε η ευρετική θα επιστρέψει 0 αλλιώς κρατάω σε μία μεταβλητή την τωρινή θέση και όσο υπάρχουν γωνίες στη λίστα not visited βρίσκω το κόστος από την τωρινή θέση ως τη κοντινότερη γωνία απ τις not_visited, το προσθέτω στην συνολική απόσταση και κρατάω και αυτήν την κοντινότερη γωνία για να την μαρκάρω ως τωρινή θέση και να την βγάλω απ τις not visited. Συνεχίζω απ αυτή τη γωνία να βρω το κόστος προς την επόμενη κοντινότερη μέχρι να βρω το κόστος για όλες

απ το τωρινό σημείο του state πηγαίνοντας κάθε φορα στην κοντινότερη γωνία το οποίο και επιστρέφω.

Q7: All Food Heuristic
Στην ευρετική για όλο το φαϊ παίρνω σε λίστα την κάθε θέση που υπάρχει φαγητό και επιστρέφω την απόσταση από την τωρινή θέση μέχρι
το πιο μακρινό φαγητό

Q8: Closest Dot Suboptimal Search
Στον έλενχο της AnyFoodSearchProblem για goal state επιστρέφω true άμα οι συντεταγμένες του state είναι συντεταγμένες φαγητού.
Στον search agent χρησιμοποιώ την υλοποίηση του BFS για την εύρεση των κουκιδων