Τεχνητή Νοημοσύνη 1 – Χειμερινό 2021-2022

Εργασία πρώτη – README

Κατσαούνη Σοφία Μερόπη , sdi1800070

Q1 – Q3:

Η υλοποίηση των ερωτημάτων 1-3 βασίστηκε στον ψευδοκώδικα που μας δώθηκε.

Question 1: χρησιμοποίησα **stack**, εφόσον ο **DFS** χρειάζεται LIFO λειτουργικότητα.
Question 2: χρησιμοποίησα **queue** εφόσον ο **BFS** χρειάζεται FIFO λειτουργικότητα.
Question 3: χρησιμοποίησα **priority queue** ώστε να μπορώ να βγάζω από το σύνορο τους κόμβους με αύξουσα σειρά και σε συνδυασμό με την ευρετική συνάρτηση να γίνεται ''καλύτερη'' επιλογή μονοπατιού. Εδώ το κάθε node είναι ένα tuple που περιέχει μια κατάσταση και λίστα κινήσεων μέχρι αυτήν την κατάσταση.

Question 4:

getStartState(): Επιστρέφεται ένα tuple που περιέχει το position και ένα tuple που θα περιέχει τις γωνίες που έχει περάσει το πακμαν μέχρι αυτήν την κατάσταση. Θεωρώντας ότι η αρχική κατάσταση δεν είναι μια από τις γωνίες, το συγκεκριμένο tuple είναι αρχικά άδειο. Επέλεξα tuple γιατί το expanded set που έχω στην BFS απαιτεί ο τύπος δεδομένων να είναι hashable.

isGoalState(): Αν η λίστα με τις γωνίες φτάσει μήκος 4, το πακμαν έχει περάσει από όλες τις γωνίες και άρα έχουμε επιτυχία.

expand(): Το tuple suc_corners κρατάει όλες τις γωνίες successors, και τελικά αυτό που αποθηκεύεται στην λίστα children είναι ένα tuple όπου περιέχει άλλο ένα tuple με μια κατάσταση γωνία, και όλες τις successor γωνίες της συγκεκριμένης, το action και το κόστος αυτού του action.

Question 5: Χρειάστηκε ο υπολογισμός απόστασης στην ευρετική, οπότε από τις manhattan και την maze distance που δινόταν, επέλεξα την manhattan καθώς παράγει πιο οικονομικά αποτελέσματα και δεν μπορεί να υπερεκτιμήσει εφόσον υπολογίζει την βέλτιστη δυνατή απόσταση χωρίς τοίχους. Από άποψη αλγορίθμου βρίσκω όλες τις αποστάσεις από το state στο οποίο βρίσκεται προς όλες τις γωνίες και τελικά βρίσκω την μικρότερη απόσταση. Αυτό επαναλαμβάνεται μέχρι να περάσουμε από όλες τις γωνίες.

Question 6: Με παρόμοια λογική με αυτήν της ερώτησης 5, έπρεπε να πειραματιστώ με το ποία συνάρτηση θα επιλέξω την manhattan ή την maze distance και ποια απόσταση θα επιλέξω από το current state (την μικρότερη ή την μεγαλύτερη). Τελικά κατέληξα στην επιλογή της maze distance διότι λαμβάνει υπόψην και τους τοίχους όταν υπολογίζει την απόσταση σε αντίθεση με την manhattan, και στην επιλογή του φαγητού που βρίσκεται όσο το δυνατόν πιο μακρυά από το πακμαν. Η υλοποίηση μου καθώς και ο autograder παίρνει λίγη ώρα να εκτελεστεί (περίπου 50 sec) όποτε λιγάκι υπομονή! :)

Question 7:

findPathToClosestDot(): Εφόσον επιθυμούμε την συντομότερη διαδρομή για οποιοδήποτε φαγητό, χρησιμοποιούμε τον BFS εφόσον και οι αποστάσεις είναι μοναδιαίες.

isGoalState(): Θέλουμε να βρούμε διαδρομή για οποιοδήποτε φαγητό επομένως κάθε φαγητό αποτελεί goal state και άρα αρκεί να ελέγξουμε αν ένα state έχει φαγητό ή όχι.