ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

Τεχνητή Νοημοσύνη
1η Εργασία

Μανούσος Λιναρδάκης, it22064

Ερώτηση 1:

Ένας κόμβος στη λύση που έκανα αναπαρίσταται με ένα tuple, το οποίο αποτελείται από τον successor του state που επεκτείνω εκείνη την στιγμή και τις ενέργειες για να φτάσω σε αυτόν από την αρχική κατάσταση.

Ερώτηση 2:

Τρέχοντας “python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic”, παρατηρούμε ότι χρειάστηκαν 549 κόμβοι για τον a*, ενώ τρέχοντας “python pacman.py -l bigMaze -z .5 -p SearchAgent \
-a fn=ucs,heuristic=manhattanHeuristic”, χρειάστηκαν 620 για τον ucs.
Αντίστοιχα για τον openMaze, οι κόμβοι που χρειάστηκαν είναι 535 για τον a* και 682 για τον ucs.

Ερώτηση 3:

Η λογική της ευρετικής συνάρτησης που υλοποιήθηκε είναι η εύρεση της απόστασης manhattan του πράκτορα από τις γωνίες που δεν έχει επισκεφθεί ακόμα. Η απόσταση manhattan προσφέρει μία αποδεκτή και συνεπής λύση στο pacman, αφού (ο pacman) μπορεί να μετακινηθεί μόνο κάθετα ή οριζόντια. Έτσι, η manhattan θα ήταν η βέλτιστη διαδρομή αν η πίστα ήταν χωρίς εμπόδια και, αν υπάρχουν εμπόδια, η ευρετική συνάρτηση θα είναι πάντα μικρότερη του πραγματικού κόστους (το ίδιο ισχύει και για την διαφορά των ευρετικών δύο σημείων, η οποία είναι μικρότερη ή ίση από το πραγματικό κόστος μεταξύ τους).
Επίσης, επιλέγοντας το max των αποστάσεων manhattan, μειώνουμε τον αριθμό των nodes που γίνονται expand (διατηρώντας την συνέπεια και την αποδοχή της λύσης). Με αυτόν τον τρόπο, η παραπάνω ευρετική συνάρτηση επεκτείνει 1136 nodes (σύμφωνα και με τον grader).

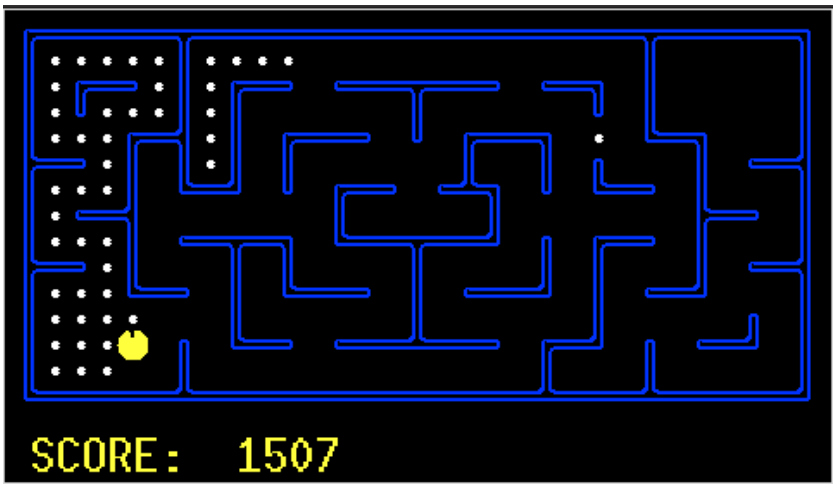
Ερώτηση 4:

Η λογική της ευρετικής συνάρτησης που υλοποιήθηκε είναι παρόμοια με αυτή του 3ου ερωτήματος, μόνο που αυτή τη φορά οι κουκίδες δεν βρίσκονται μόνο στις γωνίες της πίστας. Επιστρέφει και πάλι την μέγιστη manhattan distance, διατηρώντας την συνέπεια και αποδοχή της λύσης (όπως αποδείχθηκε και πριν). Τα nodes που επεκτείνονται με αυτό τον τρόπο είναι 9551.

Ερώτηση 5:

Ο ClosestDotSearchAgent δεν θα βρίσκει πάντα την βέλτιστη λύση, καθώς κοιτάει μόνο τις πιο κοντινές κουκίδες, με αποτέλεσμα να αποφεύγει αυτές που βρίσκονται σε πιο πλεονεκτική θέση εκείνη την στιγμή. Ένα παράδειγμα αποτελεί η συμπεριφορά του πράκτορα όταν τρέχουμε: python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5.

Παρατηρούμε το εξής:



Ο pacman έχει αφήσει την κουκίδα δεξιά, καθώς έβλεπε μόνο όσες ήταν πιο κοντά του όταν ήταν εκεί (ClosestDotSearchAgent). Στο τέλος αναγκάζεται να διασχίσει όλη την πίστα από αριστερά προς τα δεξιά, καθώς είναι η μόνη “κοντινή” κουκίδα που έχει μείνει, κάνοντας έτσι τη λύση υποβέλτιστη.