National And Kapodistrian University of Athens
Department of Informatics and Telecommunications

Project #2

Student name: *Stylianos Bitzas (sma1100202)*

Course: *Artificial Intelligence (ΥΣ02)* – Professor: *Manolis Koubarakis*
Due date: *December 3rd, 2019*

Question 1

Να κάνετε το Pacman project P2 (http://ai.berkeley.edu/multiagent.html)

Answer. Γενικά έχω συμπεριλάβει αναλυτικά σχόλια σε όλο τον κώδικα που έχω γράψει. Θα σχολιάσω μόνο τη δόμη της λύσης και κάποιες λεπτομέρειες για τις αποφάσεις που πήρα.

- ***Για τα ερωτήματα Q1+Q5*** χρησιμοποίησα τον ίδιο αλγόριθμο με μόνες διαφορές ,ότι στο Q1 ο αλγοριθμός χρησιμοποιεί το action που λαμβάνει και παράγει ένα successor state ένω στο Q5 έχω εξαρχής το currentState , και μία συνθήκη για τα win ή lose states, όπου στο πρώτο το lose state έχει μεγαλύτερη (κατά απόλυτη τιμή) τιμή από το win state ενώ στο δεύτερο έχουν ίδια τιμή, αυτό προέκυψε με δοκίμες καθώς ήξερα ότι χρειάζομαι μεγάλες τιμές αλλά για κάποιο λόγο στο reflex πέθαινε στο τελευταίο βήμα σε καποιά που ήταν και win και lose state ταυτόχρονα και για να αποφύγω να πεθαίνει έδωσα μεγαλύτερο βάρος στο να μην πεθαίνει από το να νικάει. Η βασική ιδέα για την αξιολόγηση είναι ότι μία κατάσταση είναι κάλη αν είναι (σχετικά) μακρία από φάντασμα, κοντά σε κάψουλες, κοντά στην πιο κοντινή τροφή(και κοντά στην πιο μακρινή τροφή σε περίπτωση ίσης απόστασης βοηθάει), απομένει λιγότερη τροφή και κάψουλες και το default score είναι μεγάλο. Χρησιμοποιώ κάποιες βοηθητικές συναρτήσεις που βρίσκονται στο τέλος του module multiAgents.py με ονόματα : closestFood, closestGhost, closestCapsule, farthestFood και euclideanDistance. Το όνομα των συναρτήσεων προδίδει την λειτουργία τους αλλά υπάρχουν και σχόλια στον κώδικα. Τα υπολοίπα κομμάτια του κώδικα είναι διαδικαστικά και βασιζονται σε απλή λογική για την δημιουργία συντελεστών των βασικών τιμών και αποθηκεύση των τιμών καθώς και υπολογισμό του γραμμικού συνδυασμού των τιμών πολλαπλασιασμένων με τους συντελεστές. Επείδη συνήθως οτάν κάποιες από τις τιμές μειώνονται, τότε βρισκόμαστε σε καλύτερες καταστάσεις, έχω πάρει την αντίστροφη τιμή και διαχειρίζομαι κατάλληλα για πιθανή διαίρεση με 0. Οι συντελεστές βρέθηκαν με βάση την απλή λογική (π.χ. ο αριθμός (αντίστροφος) εναπομείναντων τροφών να είναι ο συντελέστης του κοντινότερου φαγητου κλπ.) και σταθεροποιήθηκαν οι τιμές τους έπειτα από δοκιμές και καταγράφοντας τις τίμες που δίνανε στον autograder. Με λίγα λόγια υπηρχέ η ίδεα από πίσω που βασιζόταν σε κάποιες τιμές αλλά χρειαστηκάν δοκιμές και λίγο ”duck typing” για να λειτουργήσει σωστά η συνάρτηση αξιολόγησης. Μία επέκταση με καλύτερα αποτελέσματα θα χρειαζόταν καλύτερη κατανόηση όλου του παιχνιδιού του pacman και ίσως αλλαγές σε κάποια άλλα σημεία εκτός του module multiAgents (π.χ. για να μετράει και ο χρόνος που ο pacman βρισκέται σε ένα state αρνητικά απευθείας και όχι μέσω του score).

- ***Για το ερώτημα Q2*** η μέθοδος minimax διαχειρίζεται τις κινήσεις των παικτών. Για να ελέγχει αν ένας παίκτης είναι min ή max χρησιμοποιώ ένα δείκτη agentIndex που αυξάνεται κάθε φορά (κατά ένα) που καλείται μία συνάρτηση min ή max και ο διαχωρισμός για να ξέρουμε αν ο παίκτης είναι min ή max, δεδομένου ότι οι min κινούνται διαδοχικά, γίνεται χρησιμοποιώντας το υπόλοιπο της διαίρεσης του δείκτη με τον αριθμό των παικτών. Οπότε αν είναι 0 βρισκόμαστε στον αρχικό παίκτη max (pacman) και σε κάθε άλλη περίπτωση(1,2,...,m-2, όπου m-1 αριθμός των παικτών) σε παίκτη min (ghosts) επιπλέον σε κάθε κλήση όλων των μεθόδων υπάρχει μία μεταβλητή curDepth που κρατάει το τρέχον βάθος για να το συγκρίνει με το βάθος που θελούμε να σταματάει ο αλγόριθμος. Η μέθοδος maxValue είναι η κίνηση του max, αν ποτέ φτάσει σε τερματική κορυφή τερματίζει και επιστρέφη την τιμή . Έπειτα παράγει όλους τους απογόνους σε κάθε κόμβο(αφού θα παραχθεί ολό το δέντρο τους δημιουργώ ολόυς με ένα list comprehension και τους διατρέχω) και καλεί την συναρτήση διαχείρισης (minimax) αυξάνοντας τον δείκτη του παίκτη κατά ένα και τον δείκτη του τρέχοντος βάθους κατά ένα (μονό αφότου έχουν παίξει όλοι οι min παίκτες και παίζει δηλαδή ο max, αυξάνω το τρέχον βάθος) . Τέλος σε κάθε κίνηση max κρατάω τις τιμές των απογονών που υπολογίζονται αναδρομικά και ανανέωνω κατάλληλα με βάση τον αλγόριθμο minimax της θεωρίας ώστε να βρω τελικά την maxmin τιμή. Η μέθοδος min κάνει τα ακριβώς τα ίδια απλά για τους min παίκτες και υπολογίζει την minmax τιμή. Η min δεν αυξάνει το τρέχων βάθος ποτέ όταν καλεί την minimax μέθοδο παρά μόνο τον δείκτη των παικτών.

- ***Για το ερώτημα Q3*** χρησιμοποιώ τον κώδικα του MinimaxAgent με κάποιες αλλαγές για να γίνεται το κλάδεμα. Αρχικά οι successors παράγονται ενάς ενάς ώστε αν προκύψει κλάδεμα να επωφεληθούμε και να γλιτώσουμε την παραγωγή κόμβων παιδιών. Έπειτα χρησιμοποίουμε τις maxValue και minValue μεθόδους του AlphaBeta αλγορίθμου για να κάνουμε το κλάδεμα και να αλλάζουμε τις τιμές alpha και βήτα, που περνάνε σε κάθε μέθοδο με τα ορίσματα alpha, beta και έχουν αρχικές τιμές στην ρίζα $+\infty. -\infty$ αντίστοιχα.

- ***Για το ερώτημα Q4*** χρησιμοποιώ τον κώδικα του MinimaxAgent και αντί για minValue μέθοδο, έχω μία chance μέθοδο που λειτουργεί σαν την minValue με διαφορά ότι υπολογίζει την μέση τιμή όλων των απογόνων, χρησιμοποιώντας ένα διάνυσμα πιθανότητας που δίνει μία πιθανότητα στο $[0, 1]$ σε κάθε απόγονο πολλαπλασιασμένο με την τιμή του απογόνου και τελός αθροίζοντας όλες αυτές τις τιμές. Σε αυτήν την υλοποίηση θεώρω ότι ολοί οι απόγονοι έχουν την ίδια πιθανότητα.