2η Εργασία στο μάθημα Τεχνητή Νοημοσύνη

Αθανασία Τουρνάκη
ΑΜ: 1115201600172

Πρόβλημα 4(Pacman Project 2):

Σε αυτήν την εργασία υλοποιήθηκαν όλα τα ερωτήματα, δηλαδή πραγματοποιήθηκαν αλλαγές στο αρχείο multiagents.py. Οι υλοποιήσεις έγιναν με βάση τις διαλέξεις και την θεωρία που έχουμε διδαχτεί στο μάθημα.

Αναλυτικά για κάθε ερώτημα:

Question 1: Reflex Agent

Έγινε επεξεργασία της ήδη υπάρχουσας κλάσης του multiagents.py, ReflexAgent και συγκεκριμένα τις μεθόδου της, evaluation Function. Η λογική της υλοποίησης είναι να υπάρχουν προσθαφαιρέσεις στο σκορ, όσο το Pacman κινείται σε States επιλέγοντας κάποιο Action, με κάποιους παράγοντες να τις επηρεάζουν, έτσι ώστε να επιτευχθεί η καλύτερη δυνατή συμπεριφορά του Pacman.

Πιο συγκεκριμένα:

i) Υπολογίζεται η απόσταση Manhattan μεταξύ του Pacman και όλων των φαγητών, επιστρέφεται η μικρότερη από αυτές και προστίθεται στο σκορ 100 προς την ποσότητα αυτή. Δηλαδή, όσο μικρότερη απόσταση έχει το Pacman από το φαγητό κατά την μετακίνηση του στην νέα του θέση, τόσο μεγαλύτερο σκορ πετυχαίνει.

ii) Αν η θέση προς την οποία κατευθύνεται το Pacman περιέχει φαγητό, του δίνονται σαν bonus +500 βαθμούς.

iii) Αν στη νέα θέση που έχει σκοπό να πάει το Pacman με την κίνηση που κάνει βρίσκεται κάποιο φάντασμα το σκορ παίρνει την τιμη -10000, έναν πολύ χαμηλό αριθμό, έτσι ώστε το Pacman να αποφύγει αυτή τη θέση και συνεπώς τον θάνατό του.

iv) Τέλος, προσδίδεται μια ποινή στο Pacman και αφαιρούνται 500 βαθμοί από το σκορ, όσο αυτό επιλέγει να μην κινείται στον λαβύρινθο.

Η συνάρτηση επιστρέφει το σκορ που υπολογίζεται από τα παραπάνω.
Οι συντελεστές έχουν προκύψει πειραματικά.

Question 2: Minimax

Υλοποιήθηκαν οι συναρτήσεις MaxValue και MinValue της μεθόδου getAction της κλάσης MinimaxAgent στο multiAgents.py. Σε αυτές τις συναρτήσεις υπάρχει μία ιδιαιτερότητα συγκριτικά με αυτές που έχουμε διδαχτεί. Επειδή, τα φαντάσματα-αντίπαλοι του Pacman είναι παραπάνω από ένα, δε μπορεί να υλοποιηθεί το κλασσικό Minimax με τους δύο παίκτες. Επομένως, έχει τροποποιηθεί ελαφρώς η λειτουργία της MinValue, έτσι ώστε να επιτρέπει την συνεργασία μεταξύ φαντασμάτων. Ένας γύρος τελειώνει όταν θα έχει παίξει το Pacman καθώς και όλα τα φαντάσματα που υπάρχουν στον λαβύρινθο.

Αναλυτικά για κάθε συνάρτηση:

MaxValue(self, gameState, depth) :

Υπολογίζονται οι νόμιμες κινήσεις του Pacman. Εαν δεν υπάρχει διαθέσιμη κίνηση, η σειρά του Pacman τελειώνει. Αρχικοποιείται μια τιμή v = -οο, όπως στην θεωρία. Αρχικοποιείται η μεταβλητή move σε 0. Για κάθε νόμιμη κίνηση, υπολογίζεται η MinValue στο gameState που προκύπτει από αυτήν την κίνηση για το πρώτο φάντασμα του ίδιου βάθους. Αν η τιμή που επιστρέφεται είναι μεγαλύτερη από την τιμή v, τότε γίνεται η νέα τιμή v και η κίνηση που οδήγησε σε αυτή την τιμή γίνεται η προτιμητέα. Αν το βάθος(που δίνεται σαν όρισμα) είναι ίσο με το 0, δηλαδή η απόφαση που θα παρθεί είναι απόφαση ρίζας, επιστρέφεται η κίνηση που επιφέρει καλύτερο σκορ, αλλιώς επιστρέφεται η τιμή που θα λάβει ο κόμβος MAX στο δέντρο παιχνιδιού.

MinValue(self, gameState, depth, GhostNo):

Υπολογίζονται οι νόμιμες κινήσεις του φαντάσματος με index GhostNo. Αν δεν υπάρχουν κινήσεις, η σειρά του τελειώνει. Αν το φάντασμα είναι το τελευταίο, τίθεται σαν επόμενος πράκτορας(agent) το Pacman, αλλιώς το επόμενο σε σειρά φάντασμα. Αρχικοποιείται μια μεταβλητή v σε +οο. Για κάθε νόμιμη κίνηση θα υπάρχουν τρεις περιπτώσεις:

1) Αν ο επόμενος πράκτορας είναι το Pacman και το βάθος που έχουμε δώσει σαν όρισμα είναι ίσο με το βάθος του δέντρου παιχνιδιού πλην 1, δηλαδή αυτό θα είναι το τελευταίο φάντασμα που θα παίξει στο παιχνίδι, μια τιμή v1 θα λάβει την τιμή του αποτελέσματος του evaluationFunction για το επόμενο State, με βάση την κίνηση. Αυτό σημαίνει ότι θα βρισκόμαστε σε φύλλο, οπότε δε θα είχε νόημα να καλέσουμε MaxValue ή MinValue.

2) Αν ο επόμενος πράκτορας είναι το Pacman και το βάθος είναι διαφορετικό, τότε η v1 θα πάρει την τιμή που θα επιστρέψει η MaxValue στο επόμενο state(με βάση την κίνηση), για το Pacman που βρίσκεται στο επόμενο βάθος δέντρου.

3) Αν ο επόμενος πράκτορας δεν είναι το Pacman, τότε θα είναι φάντασμα. Οπότε η v1 θα λάβει την τιμή της MinValue στο επόμενο State(με βάση την κίνηση που κάνει) για το επόμενο φάντασμα του ίδιου βάθους. Αυτό εξασφαλίζει ότι κάθε φάντασμα θα μειώνει όλο και περισσότερο τη χρησιμότητα για το Pacman-MAX.

Αν η v1 που προκύπτει ανάλογα με μία από τις παραπάνω περιπτώσεις είναι μικρότερη από την v, τότε η v παίρνει την τιμή της v1.

getAction(self, gameState):

Πέρα από τους ορισμούς των παραπάνω συναρτήσεων, η getAction περιέχει και μία κλήση προς την ρίζα του δέντρου, το MAX-Pacman για βάθος 0.

Question 3: Alpha-Beta Pruning

Υλοποιήθηκαν οι συναρτήσεις MaxValue και MinValue της μεθόδου getAction της κλάσης AlphaBetaAgent στο multiAgents.py. Η υλοποίηση των συναρτήσεων είναι σε μεγάλο βαθμό ίδια με αυτή τωναντίστοιχων συναρτήσεων του ερωτήματος 2, οπότε δεν θα αναλυθούν σε βάθος.

Για τις συναρτήσεις:

MaxValue(self, gameState, depth, a, b) :

Υλοποιήθηκε όπως η MaxValue του Minimax με τη μόνη διαφορά ότι υπάρχουν δύο παραπάνω μεταβλητές a, b που κρατάνε τη μεγαλύτερη τιμή που έχει πάρει ένας κόμβος MAX και ένας κόμβος MIN αντίστοιχα. Στο σημείο του κώδικα που εξετάζεται για κάθε νόμιμη κίνηση η τιμή που θα επιφέρει, πέρα από την εκτέλεση της MinValue για το πρώτο φάντασμα του βάθους του Pacman και την σύγκρισή της με την μεταβλητή v(=-οο), γίνονται και τα παρακάτω:
α) Η μεταβλητή a λαμβάνει τη μεγαλύτερη τιμή ανάμεσα σε αυτή που έχει ήδη και αυτή της v.
β) Επιστρέφεται η v, αν η τιμή της είναι μεγαλύτερη του b.

MinValue(self, gameState, depth, GhostNo, a, b):

Υλοποιήθηκε όπως η MinValue του Minimax, αλλά με τις δύο επιπλέον μεταβλητές a, b που έχουν την λειτουργία που περιγράφηκε παραπάνω. Και πάλι, στο σημείο του κώδικα που εξετάζεται για κάθε νόμιμη κίνηση η τιμή που θα επιφέρει, μετά από την εξέταση των ίδιων περιπτώσεων της MinValue του Minimax, εκτελούνται τα παρακάτω:

α) Η μεταβλητή β λαμβάνει τη μικρότερη τιμή ανάμεσα σε αυτή που έχει ήδη και αυτή της v.
β) Επιστρέφεται η v, αν η τιμή της είναι μικρότερη του a.

getAction(self, gameState):

Πέρα από τους ορισμούς των παραπάνω συναρτήσεων, η getAction περιέχει και μία κλήση προς την ρίζα του δέντρου, το MAX-Pacman για βάθος 0 και με τιμές για τα a, b -οο και +οο αντίστοιχα.

Question 4: Expectimax

Υλοποιήθηκαν οι συναρτήσεις MaxValue και ChanceValue της μεθόδου getAction της κλάσης ExpectimaxAgent στο multiAgents.py. Η υλοποίηση των συναρτήσεων είναι σε μεγάλο βαθμό ίδια με αυτή τωναντίστοιχων συναρτήσεων του ερωτήματος 2, οπότε δεν θα αναλυθούν σε βάθος.

Αναλυτικά κάθε συνάρτηση:

MaxValue(self, gameState, depth, a, b) :

Υλοποιήθηκε όπως η MaxValue του Minimax με τη μόνη διαφορά ότι η μεταβλητή v1 κρατάει την τιμή που επιστρέφεται από την κλήση της ChanceValue και όχι της MinValue.

ChanceValue(self, gameState, depth, GhostNo):

Υλοποιήθηκε όπως η MinValue του Minimax, αλλά ενώ οι περιπτώσεις είναι ίδιες όπως στην MinValue, εδώ κάθε αποτέλεσμα που φέρνει μια νόμιμη κίνηση προστίθεται σε μια μεταβλητη v(που αρχικοποιείται σε 0) και στο τέλος υπολογίζεται το μέσο σκορ ως η διαίρεση της τελικής τιμής της μεταβλητής v προς τον αριθμό των νόμιμων κινήσεων. Αυτό συμβαίνει διότι η ChanceValue αθροίζει για όλες τις κινήσεις την πιθανότητα να γίνει αυτή η

κίνηση επί το σκορ που θα επιφέρει. Σε αυτήν την περίπτωση, όλες οι κινήσεις είναι ισοπίθανες εξ'ου και ο κοινός παρονομαστής.

getAction(self, gameState):

Πέρα από τους ορισμούς των παραπάνω συναρτήσεων, η getAction περιέχει και μία κλήση προς την ρίζα του δέντρου, το MAX-Pacman για βάθος 0.

Question 5: Evaluation Function

Υλοποιήθηκε η συνάρτηση betterEvaluationFunction του multiagents.py. Αξιολογείται κατά την κλήση της το σκορ που θα επιστραφεί αν το Pacman φτάσει στο State για το οποίο καλείται ανεξάρτητα από την κίνηση για να φτάσει σε αυτό. Η λογική της υλοποίησης είναι και σε αυτήν την evaluation function να υπάρχουν προσθαφαιρέσεις στο σκορ, με κάποιους παράγοντες να τις επηρεάζουν, έτσι ώστε να επιτευχθεί η καλύτερη δυνατή συμπεριφορά του Pacman.

Πιο συγκεκριμένα:

i) Υπολογίζεται η απόσταση Manhattan μεταξύ του Pacman και όλων των φαγητών, επιστρέφεται η μικρότερη από αυτές και προστίθεται στο σκορ 10 προς την ποσότητα αυτή. Δηλαδή, όσο μικρότερη απόσταση έχει το Pacman από το φαγητό κατά την μετακίνηση του στην νέα του θέση, τόσο μεγαλύτερο σκορ πετυχαίνει.

ii) Εξετάζονται δύο περιπτώσεις:

α) Αν όλα τα φαντάσματα που βρίσκονται στον λαβύρινθο είναι τρομαγμένα (0 not in scaredTimes), τότε υπολογίζεται η απόσταση Manhattan μεταξύ του Pacman και όλων των φαντασμάτων, επιστρέφεται η μικρότερη από αυτές και προστίθεται στο σκορ 60 προς την απόσταση αυτή. Δηλαδή ο Pacman επιβραβευεται σε μεγάλο βαθμό όσο πλησιάζει σε ένα τρομαγμένο φάντασμα. Προφανώς, με βάση την υλοποίηση, το Pacman θα ενδιαφέρεται περισσότερο να φάει ένα τρομαγμένο φάντασμα από ότι ένα φαγητό.

β) Αν υπάρχει έστω και ένα μη τρομαγμένο φάντασμα, τότε προτεραιότητα του Pacman γίνεται να καταναλώσει μια κάψουλα. Αυτό γίνεται υπολογίζοντας την απόσταση Manhattan μεταξύ του Pacman και όλων των κάψουλων, επιστρέφοντας την μικρότερη από αυτές και προσθέτοντας στο σκορ 55 προς τη μικρότερη απόσταση. Έτσι επιβραβεύεται όσο πιο κοντά βρίσκεται σε μία κάψουλα και άρα προτιμά αυτό το State.

Η συναρτηση επστρέφει το σκορ που υπολογίζεται από τα παραπάνω και το προσθέτει στο ήδη υπάρχον σκορ.
Οι συντελεστές έχουν προκύψει πειραματικά.