Εργασία στην Τεχνητή Νοημοσύνη

Βασίλειος Κοκκινογένης it2021042

Ερώτηση 1: Εξηγήστε πως αναπαρίσταται ένας κόμβος του δέντρου αναζήτησης στη λύση σας

Απάντηση: Ένας κόμβος στο δέντρο αναζήτησης στη λύση μου αναπαρίσταται ως ένας συνδυασμός από (state, path). Το state αφορά τα coordinates του κόμβου (x,y), ενώ το path αφορά μια λίστα από ενέργειες που οδηγούν τον πράκτορα από την αρχική κατάσταση στον κόμβο.

Ερώτηση 2: Που οφείλεται η διαφορά στις καταστάσεις που διερευνεί ο BFS έναντι του DFS;

Απάντηση: Η διαφορά στις καταστάσεις που διερευνεί ο BFS έναντι του DFS οφείλεται στο γεγονός ότι πλέον σαν δομή δεδομένων για την αποθήκευση των κόμβων χρησιμοποιούμε την util.Queue() έναντι της util.Stack() που χρησιμοποιούσαμε προηγουμένως. Η ουρά ακολουθεί FIFO διάταξη, δηλαδή ό,τι μπαίνει πρώτο στην λίστα (τα παιδιά του κόμβου), είναι το πρώτο που θα βγει από την λίστα (και στη συνέχεια θα επεκταθεί). Αντιθέτως, η στοίβα ακολουθεί LIFO διάταξη, δηλαδή ό,τι μπαίνει τελευταίο στην λίστα, είναι το πρώτο που θα βγει από την λίστα.

Ερώτηση 3: Ποια είναι η συνάρτηση κόστους του StayEastsearchAgent και ποια του StayWestsearchAgent; Εξηγηστε γιατί οι συναρτήσεις αυτές οδηγούν τον Pacman ανατολικά (δεξιά) και δυτικά (αριστερά) αντίστοιχα.

Απάντηση: Η συνάρτηση κόστους του StayEastsearchAgent για την είσοδο σε μια θέση (x,y) είναι 1/2^x, ενώ του StayWestsearchAgent είναι 2^x. Η συνάρτηση 1/2^x οδηγεί τον Pacman ανατολικά επειδή όσο πηγαίνει πιο ανατολικά ο Pacman, το x αυξάνει και στην συνάρτησή μας το x βρίσκεται στον παρανομαστή, συνεπώς όσο αυξάνει το x τόσο μειώνεται το κόστος και ο Pacman ενθαρρύνεται στο να κινείται προς τα δεξιά. Αντιθέτως η συνάρτηση 2^x οδηγεί τον Pacman δυτικά επειδή όσο πηγαίνει πιο δυτικά ο Pacman, το x μειώνεται και στην συνάρτησή μας το x βρίσκεται στον ονομαστή, συνεπώς όσο μειώνεται το x τόσο μειώνεται το κόστος και ο Pacman ενθαρρύνεται στο να κινείται προς τα αριστερά.

Ερώτηση 4: Πόσους κόμβους απαιτεί ο UCS και ο A* στον bigMaze ; Πόσοι είναι οι αντίστοιχοι κόμβοι για τον openMaze;

Απάντηση: Ο UCS απαιτεί 620 κόμβους στον bigMaze και 682 στον openMaze. Αντίστοιχα ο A* απαιτεί 549 κόμβους στον bigMaze και 535 στον openMaze.

Ερώτηση 5: Εξηγήστε τη λογική της ευρετικής συνάρτησης που υλοποιήσατε, τον αριθμό των κόμβων που επεκτείνει, καθώς και γιατί θεωρείτε ότι είναι αποδεκτή και συνεπής.

Απάντηση: Η λογική της ευρετικής συνάρτησης που υλοποίησα έχει ως εξής:
Αρχικά υπολογίζει την Manhattan απόσταση από την τρέχουσα τοποθεσία του pacman μέχρι την κοντινότερη κουκκίδα φαγητού. Στη συνέχεια υπολογίζει την ποσότητα των κουκκίδων που έχουν μείνει και την αναθέτει σε μια μεταβλητή με όνομα food_count. Έπειτα, αν έχει αποθηκευμένο το food_count από προηγούμενο iteration στο heuristicInfo και το τωρινό food_count είναι μεγαλύτερο από το αποθηκευμένο τότε επιστρέφει κόστος άπειρο, αλλιώς αποθηκεύει το τωρινό food_count στο heuristicInfo και επιστρέφει κόστος ίσο με την απόσταση που είχε υπολογίσει αρχικά.
Η ευρετική συνάρτηση που υλοποίησα επεκτείνει μόνο 116 κόμβους (διπλό bonus εδώ). Θεωρώ ότι είναι αποδεκτή επειδή υπολογίζει την Manhattan απόσταση για την κοντινότερη κουκκίδα φαγητού, με αποτέλεσμα να μην μπορεί να υπερεκτιμήσει το κόστος που χρειάζεται για να φάει όλες τις κουκκίδες φαγητού (goal state). Επιπλέον, θεωρώ ότι είναι συνεπής επειδή ακριβώς υπολογίζει την Manhattan απόσταση για την κοντινότερη κουκκίδα φαγητού, με αποτέλεσμα η ευρετική συνάρτηση να είναι μια μονότονα μη αυξανόμενη συνάρτηση.

Ερώτηση 6: Πως εξηγείτε τη συμπεριφορά του Pacman στην εκτέλεση του παιχνιδιού q6 παραπάνω; Γιατί ενώ εκτελούμε τον αλγόριθμο MiniMax ο πράκτοράς μας μπορεί να χάνει; Γιατί ο πράκτορας σε ορισμένες περιπτώσεις περιμένει τα φαντάσματα να πλησιάσουν;

Απάντηση: Η συμπεριφορά του Pacman στην εκτέλεση του παιχνιδιού q6 εξηγείται ως εξής: Ο Pacman όντας max player προσπαθεί να επιλέξει την κίνηση που του δίνει τη μεγαλύτερη αξία. Γι'αυτό επιλέγει να φάει τις κουκκίδες φαγητού, τις κουκκίδες που τον κάνουν να μπορεί να φάει φαντάσματα και τρώει και τα ίδια τα φαντάσματα όταν μπορεί, επειδή όλες αυτές οι ενέργειες του αυξάνουν το score. Αντιθέτως, αν φαγωθεί από τα φαντάσματα θα μειωθεί το score του, γι'αυτό και επιλέγει να τα αποφύγει. Ενώ εκτελούμε τον αλγόριθμο MiniMax ο πράκτοράς μας μπορεί να χάνει, γιατί ενώ ο Pacman μπορεί να παίζει βέλτιστα, και τα ίδια τα φαντάσματα παίζουν βέλτιστα. Τα φαντάσματα κυνηγούν τον Pacman και σε κάποια φάση τον κλείνουν γύρω από ένα εμπόδιο χωρίς αδιέξοδο (πάντως στην συγκεκριμένη περίπτωση ο Pacman θα μπορούσε να είχε φάει το power pellet για να σωθεί αν δεν έδινε τόσο έμφαση στο να φάει τις κουκκίδες). Ο πράκτορας σε ορισμένες περιπτώσεις περιμένει τα φαντάσματα να πλησιάσουν επειδή οποιαδήποτε άλλη κίνηση του φέρνει μικρότερη αξία από το να περιμένει ακίνητος.

Ερώτηση 7: Όταν ο Pacman βλέπει ότι θα χάσει, προσπαθεί να χάσει το συντομότερο δυνατό. Εξηγήστε γιατί συμβαίνει αυτό στην περίπτωση του MinimaxAgent. Πότε είναι λάθος η συγκεκριμένη στρατηγική και γιατί;

Απάντηση: Αυτό συμβαίνει στην περίπτωση του MinimaxAgent επειδή κάθε χρονική στιγμή που περνάει, το σκορ του pacman μειώνεται κατά 1. Συνεπώς, εφόσον ήδη ξέρει ότι θα χάσει, επιλέγει να χάσει όσο το πιο δυνατόν γρήγορα μπορεί ώστε να μην μειωθεί επιπλέον το σκορ του από τα δευτερόλεπτα που περνάνε. Η συγκεκριμένη στρατηγική είναι λάθος όταν ο Pacman βλέπει μεν ότι θα χάσει, αλλά κάτι τέτοιο δε συμβαίνει στην πραγματικότητα επειδή ο αντίπαλος δεν παίζει πάντα βέλτιστα, όπως στην περίπτωση του ExpectimaxAgent.

Ερώτηση 8: Εξηγήστε πως λειτουργεί η συνάρτηση αξιολόγησής σας. Επίσης, δοκιμάστε πως λειτουργεί ο πράκτοράς σας σε συνδυασμό με alpha-beta pruning. Πως διαφέρει η συμπεριφορά του πράκτορα σε σχέση με αυτόν του ερωτήματος 6;

Απάντηση: Η συνάρτηση αξιολόγησής μου λειτουργεί με τον εξής τρόπο:
Αρχικά, υπολογίζει την Manhattan απόσταση από την τρέχουσα τοποθεσία του pacman μέχρι την κοντινότερη κουκκίδα φαγητού. Έπειτα, υπολογίζει την Manhattan απόσταση από την τρέχουσα τοποθεσία του pacman μέχρι το κοντινότερο φάντασμα. Στη συνέχεια υπολογίζει τον αριθμό των κουκκίδων φαγητού που απομένουν, όπως και τον αριθμό των μεγάλων κουκκίδων (power pellets) αντίστοιχα. Τους 3 από τους 4 παράγοντες τους χρησιμοποιεί ως παρανομαστές στην διαίρεση 1 / (παράγοντας + 1) και τις διαιρέσεις αυτές τις προσθέτει, αφού πρώτα τις έχει πολλαπλασιάσει με το multiplier τους, και αφαιρεί τον 4ο παράγοντα closestGhostDistance με το multiplier του. Ο multiplier του closestGhostDistance αλλάζει με βάση αν τα φαντάσματα είναι τρομαγμένα ή όχι. Αν είναι τρομαγμένα τότε το multiplier γίνεται πολύ μεγάλο. Στο τέλος είτε επιστρέφει όλες αυτές τις προσθέσεις των διαιρέσεων (και την αφαίρεση του closestGhostDistance) μαζί με τα multipliers είτε επιστρέφει μείον άπειρο σε περίπτωση που το ghostDistance είναι μικρότερο του 2.

Η συμπεριφορά του πράκτορα σε σχέση με αυτόν του ερωτήματος 6 διαφέρει ως εξής: Ο πράκτοράς μας προτιμά να φάει power pellets και έπειτα να κυνηγήσει τα φαντάσματα, με αποτέλεσμα να αποκομίζει περισσότερους πόντους από τον πράκτορα του ερωτήματος 6 καθώς και να νικά με μεγάλο ποσοστό σε αντίθεση με τον πράκτορα του ερωτήματος 6 που χάνει συνέχεια.