Berkeley Pacman Project 2

By: Βάιος Λύτρας

Q1: Η μέθοδος που έφτιαξα εδώ είναι πολύ απλή. Πρώτα τσεκάρουμε αν υπάρχει κάποιο φάντασμα με manhattandistance <=1 δηλαδή δίπλα στη θέση που θέλουμε να πάει ο πακμαν, αν υπάρχει τότε επιστρέφει (-άπειρο) έτσι ώστε να μην πάει σε αυτή τη θέση

ο πακμαν.Μετά τσεκάρει αν στην θέση που θέλουμε να πάμε έχει φαγητό, τότε επιστρέφει (+άπειρο) για να πάει εκεί αφού έχει φαγητό και επίσης είναι ασφαλές καθώς τσεκάραμε πριν για το αν έχει φάντασμα εκεί.Αν δεν συμβαίνει τίποτα απο τα δύο τότε επιστρέφει έναν συνδυασμό των: μανχαταν απόσταση κοντινότερου φαγητού, μανχαταν απόσταση κοντινότερου φαντάσματος, τρέχων σκορ.

Q2: Ο αλγόριθμος minimax υλοποιήθηκε όπως ακριβώς είπαμε στις διαφάνειες με τη μόνη διαφορά ότι το value το έχω σαν πίνακα γιατί θέλω να κρατάω και την κίνηση εκτός απο την τιμή μόνο έτσι ώστε να την επιστρέψω στο τέλος.Επίσης κάθε φορά που καλείται η συνάρτηση min_value (δηλ. για τα φαντάσματα) καλείται τόσες φορες όσες και τα φαντάσματα

δηλαδή για 3 φαντάσματα κάθε "βάθος¨ θα περιέχει 1 max-layer και 3 min-layers οπότε την πρώτη φορά που καλώ την max_value περνάω σαν όρισμα το self.depth*gameState.getNumAgents() για τον λόγο που προανέφερα.

Q3: Σχεδόν ίδια υλοποίηση με το Q2 αλλά προστέθηκαν τα α,β και οι έλεγχοι που χρειάζονται για να κοπούν οι άχρηστοι κόμβοι.

Q4: Ολόιδια υλοποίηση με το Q2 απλά αντι για min_value συνάρτηση τώρα υπάρχει η exp_value που αντί να επιστρέψει το min του πίνακα (όπως ανέφερα και στο 2 αποθηκεύω τις τιμές σε πίνακα), επιστρέφει τον μέσο όρο όλων των τιμών του πίνακα.

Q5: Εδώ η evaluation function δουλεύει περίπου όπως και αυτή στο ερώτημα 1 αλλά τώρα έχω προσθέσει να καταλαβαίνει πότε τα φαντάσματα είναι φοβισμένα και αν είναι τότε στην τιμή που επιστρέφει ΠΡΟΣΘΈΤΕΙ την μανχαταν απόσταση του κοντινότερου φαντάσματος αντί να την αφαιρεί. Κανονικά την αφαιρεί γιατι θέλουμε να είμαστε όσο πιο μακριά από τα φαντάσματα γίνεται