Artificial Intelligence Project2

Σκορδούλης Κωνσταντίνος 1115 2016 00155

Σημείωση: Στα παρακάτω προβλήματα, θεωρώ ως **agent 0 ➔ pacman** και **agent 1,2,3,…,n-1 ➔ ghosts**.

Σημείωση2: Ακολούθησα την συμβουλή του site, δηλαδή αντί να χρησιμοποιήσω τις πραγματικές Manhattan αποστάσεις στις evaluation functions (Πχ distance = 5) , να χρησιμοποιώ τις **reciprocal values** (Πχ distance = 1/5) ➔ καλύτερη απόδοση.

Σημείωση3: Οι αλγόριθμοι που υλοποιούνται παρακάτω για τα Q2,Q3,Q4 , είναι αυτοί που περιγράφονται στις διαφάνειες ➔ Διαλέξεις Ενότητα 6 ➔ Διαφάνειες (17,18) , (46,47), (122). Βέβαια υπήρχαν κάποιες αλλαγές για να καλύψουμε τις απαιτήσεις των προβλημάτων (**min_players > 1**) , αλλά η δομή και ροή του αλγορίθμου είναι ίδια.

Question1:

Η evaluation function αρχικοποιείται με την τιμή 0 . Η τιμή της διαμορφώνεται από τις **Manhattan Distances** του **pacman**, από τα διαθέσιμα **Food(+)** και από τα **Ghosts(-)**. Αυτά τα δυο είναι ισοζυγισμένα, δηλαδή έχουν ίδιο factor ➔ (x1) και [x (-1)]. Οπότε η τελική τιμή διαμορφώνεται με **successorGameState.getScore() + evaluation** (return value) . (Δεν συμπερίλαβα εδώ ούτε **capsules**, ούτε **scared ghosts**).

Question2:

Καλούμαστε να υλοποιήσουμε τον **MiniMax** αλγόριθμο. Το παιχνίδι λειτουργεί ως εξής. Ο **Max player**(pacman) κάνει 1 κίνηση, μετά όλοι οι **Min_players**(ghosts) **«αντιδρούν»** σε αυτή τη κίνηση. Μόλις αντιδράσει και ο τελευταίος **Min_player ,** τότε προχωράμε στον επόμενο γύρο (**depth + 1**).

Βλέπουμε δηλαδή ότι για να αλλάξουμε depth, εκτελείται 1 φορά **Max_Value()**, ακολουθούμενη από n-1 φορές **Min_Value()**.

Ο υπόλοιπος αλγόριθμος εκτελείται όπως τον γνωρίζουμε.

Question 3:

Καλούμαστε να υλοποιήσουμε τον **AlphaBetaPruning** αλγόριθμο. Μικρές αλλαγές στον παραπάνω αλγόριθμο, απλώς προσθέτουμε τα κατάλληλα **if statements** , για να ελαχιστοποιήσουμε τους κόμβους που πρέπει να επισκεφτούμε (όπως περιγράφει ο αλγόριθμος).

Question 4:

Καλούμαστε να υλοποιήσουμε τον **ExpectiMax** αλγόριθμο. Καταρχάς, εδώ αντικαθιστώ τους **Min_κόμβους** με **Chance**_κόμβους (σαν concept το αναφέρω). Δηλαδή δεν επιλέγω ποια το min_value των παιδιών, αλλά το **Σ(P(i) * evaluation_action(i))** (άθροισμα).

Δεδομένου ότι όλες οι ενέργειες είναι **ισοπιθανες** (ομοιόμορφη κατανομή), καταλήγουμε να υπολογίζουμε τον **μέσο όρο των action(i)**.

Επομένως, η μόνη διαφορά από το Q1, βρίσκεται στο **Min_Value()**.

Question5:

Η καινούργια evaluation function, λειτουργεί όπως η προηγούμενη (αλλά με **currentGameState ➔ current_position**) και έχουμε 2 νέες προσθήκες: **scared Ghosts, Capsules**. Δηλαδή υπολογίζουμε τις **Manhattan Distances** του pacman από:

- **Food (+)** (x1)
- **Brave Ghosts (-)** (not scared ➔ dangerous) (x1)
- **Scared Ghosts (+)** (hunt them down) (x2)
- **Capsules (+)** (eating them causes, **brave ghosts ➔ scared ghosts**) (x1.5)

Πάλι χρησιμοποιώ τα **reciprocal values** των αποστάσεων (5 ➔ 1/5). Παραπάνω γράφω τον **factor/βάρος** κάθε παραμέτρου (x1 κλπ), καθώς και αν αυξάνει/μειώνει το evaluation μας.