Read me Project 1 Pacman

Άρης Τσιλιφώνης ΑΜ 1115201700170

Αρχικά στο search.py ,οι αναζητήσεις υλοποιήθηκαν με generic graph search όπως στην φωτογραφία.

Ο Γενικός Αλγόριθμος Αναζήτησης σε Γράφους

```
function GraphSearch(problem,QueuingFn)
returns a solution, or failure
    closed ← an empty set
    fringe ← MakeQueue(MakeNode(InitialState[problem]))
    loop do
        if fringe is empty then return failure
        node ← RemoveFront(fringe)
        if GoalTest[problem] applied to State[node] succeeds then
            return node
        if State[node] is not in closed then
            add State[node] to closed
            fringe ← QueuingFn(Expand(node, problem), fringe)
    end
```

Ο κάθε κόμβος είναι της μορφής (state,priority,direction) ενώ η συνάρτηση προσθήκης κόμβου είναι της μορφής (fringe,node,priority) .Παρόλο που δεν χρειάζεται το priority για τα απλά stack,queue ,θέλουμε η συνάρτηση να είναι generic.Οπότε θα έχει την ίδια μορφή για όλα τα push.Ουσιαστικά η pop είναι ίδια για όλες τι δομές ,οπότε αυτό που διαφοροποιείται είναι η pop.Τέλος η αναζήτηση

A* χρειάζεται στον υπολογισμό του priority να λάβει υπόψη και το heuristic.Όσον αφορά το search agents.py το υλοποίησα όσο πιο απλά μπορούσα.Τα σχόλια είναι πολύ επεξηγηματικά.Το μόνο που θέλω να αναφερθώ είναι το corner heuristic και το food heuristic.Το πρώτο το έκανα με manhattan distance και το δεύτερο με maze distance λαμβάνοντας υπόψη τον αλγόριθμο bfs.Εκείνος είναι πολύ πιο αποδοτικός για τον υπολογισμό της απόστασης αφού υπάρχουν πολλά food στον board.Τέλος επιστρέφω max τιμή για να είναι παραδεκτή η ευρετική. (να μην κάνει overestimate).Προφανώς είναι και consistent και τα δύο.Τέλος για το tricky search , παίρνει λίγη ώρα (περίπου 1 λεπτό καθυστέρηση) , όμως βγάζει σωστό αποτέλεσμα στον autograder.
