ΑΜ:1115201800009 ΤΥΠΑΛΔΟΣ-ΠΑΥΛΟΣ ΑΠΟΣΤΟΛΑΤΟΣ
DFS & BFS
Στο συγκεκριμενο προβλημα εχουμε 2 παραπανω δυσκολιες σε συγκριση με παραδοσιακη αναζητηση:
• δε μπορουμε να πουμε στον pacman να πηδηξει απο το ενα σημειο(state) σε αλλο.
• δε γινεται να του πουμε να προχωρησει η στριψει πανω σε ενα βασισμενοι στο δικο μας point of view γιατι πχ το δικο μας αριστερα μπορει να ειναι οποιαδηποτε κατευθυνση για αυτον. Για αυτο επισης οταν οι getsuccessor μας λεει οτι ο pacman θα παει προς μια κατευθυνση δε ξερουμε που γιατι δε ξερουμε την φορά του
Για αυτο υλοποιησα το αλγοριθμο ωστε παντα να κραταει επιπλεον πληροφορια το parent state του καθε state.προσπελασω τους parents και προσθετω τις κατευθυνσεις που δινουν στο path.αυτο εχει η πολυπλοκοτητα O(d(solution length)) και εκτελειται μια φορα ενω το dfs και το bfs O(b^d) οποτε δεν την χειροτερευει

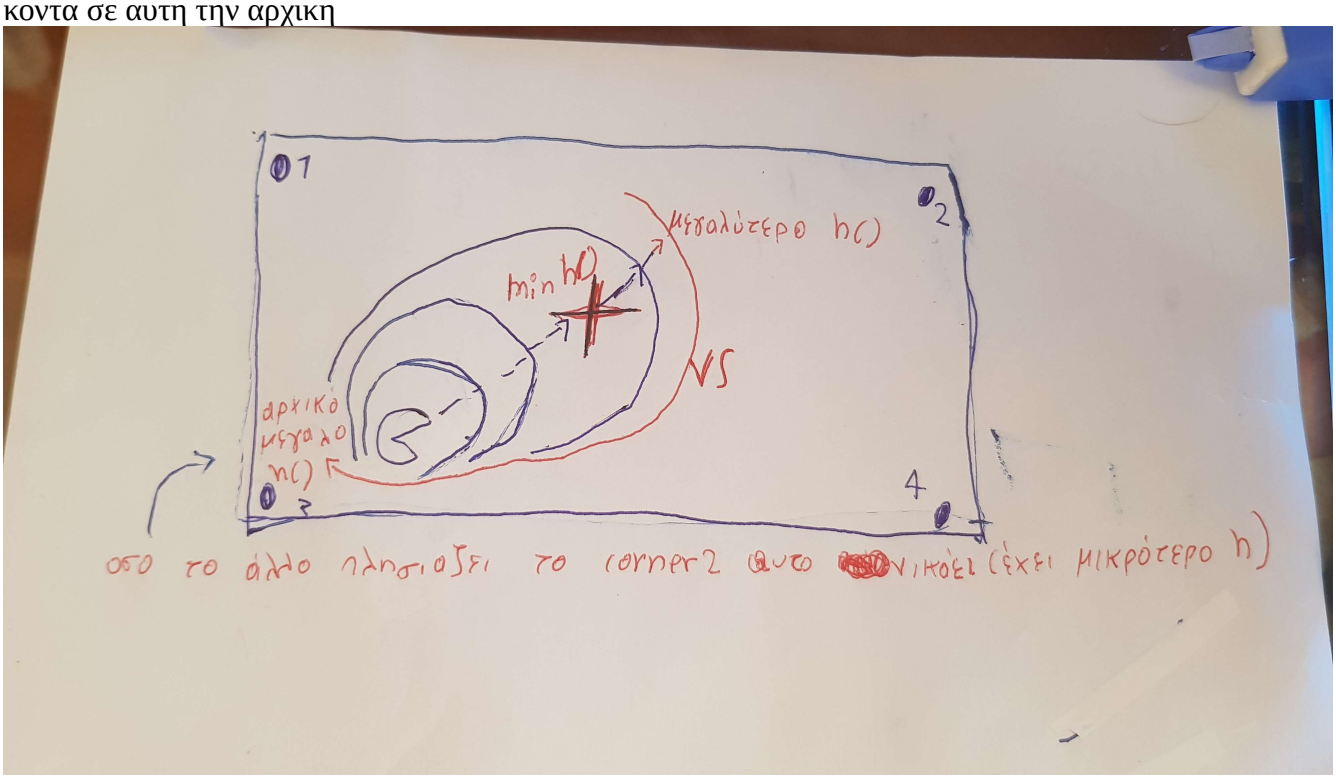
UCS:το states με το μικροτερο βαρος βγαινει απο το fringe και το βαρος αυτο ειναι η προτεραιοτητα του μεσα στο heap. Ομως στο ucs δεν μετραει μονο το βαρος της ακμη που οδηγησε στο state αυτο αλλα ολα τα βαρη των ακμων που σχηματιζουν το μονοπατια.η Pop του util.py επιστρεφει μονο το item αρα την αντικατεστησα στο search.py για να προσθετω το βαρος του μονοπατιου στο βαρος του παιδιου.αλλαξα και την update γιατι οταν ενα state ειναι στο closed δε πρεπει να γινεται push στο fringe αν δεν υπαρχει εκει.

A*:ιδια με ucs απλα προσθετω και την τιμη της ευρετικης (h)στην προτεραιοτητα που στο ucs ηταν μονο το βαρος μονοπατιου(g) φτιαχνοντας το f=g+h της θεωριας του A*

corners problem:
το να ορισω 4 στοχους αντι για 1 δεν ειναι αρκετο γιατι οταν ο pacman φτανει σε μια γωνια ειναι ‘περικυκλωμενος’ απο θεσεις(states) που ειναι ηδη μεσα στο closed που αλγοριθμου αρα δε μπορει να κουνηθει πουθενα.οι συναρτησεις του corners δεν μπορουν να σβησουν το closed του BFS αρα πρεπει να το κανουν exploit καπως. αν στο state προσθετα ενα int ποσες γωνιες εχω βρει δε θα νομιζε οτι υπαρχει μεσα στο closed αλλα επειδη δεν ξερει ποιες γωνιες εχω βρει ο pacman θα επεφτε συνεχεια πανω στην ιδια γωνια χωρις να το ξερει.ουτε μια global λιστα που λεει ποιες γωνιες εχουν εξερευνηθει θα δουλευε γιατι τα διαφορα μονοπατια του bfs θα βρισκαν τις 4 γωνιες χωρις ενα μονοπατι να κατεχει και τις 4. αρα αποφασισα καθε state να εχει εγγραφη του ποσες γωνιες βρηκε στο bfs μονοπατι του.αυτο προγραμματιστικα ειναι ευκολο με λιστα αλλα εβαλα tuple γιατι το closed μου ειναι dictionary και δε δεχεται iterable ως κλειδι. δηλαδη το state ειναι
tuple(συντεταγμενη[0],[1:5]μια ακολουθια γωνιων που εχει βρει)

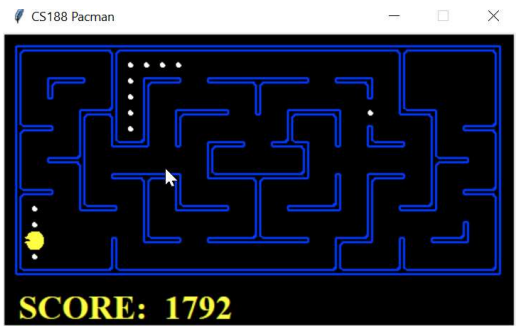
cornersheurestic:
οι ευρετικες που ειδαμε στο astar συγκριναν αποσταση απο το ενα goal οποτε πηρα αποστασεις απο τα 4 goals μας(αυτα που δεν εχουν βρεθει ηδη για ευνοητο λογο).αρχικα χρησιμοποιησα την manhattan και μου εδωσε 1357 expanded κομβους. Για να το παω κατω απο 1200 το εφτιαξα μια δικια μου eyclidian distance στον πατο του searchagents.py γιατι το util ειχε μονο μανχατταν αλλα εγινε χειροτερο με 1528 expanded κομβους. στο πατο του searchagents.py ειδα την maze distance που χρησιμοποιει την δικια μου bfs για να υπολογισει την ακριβες μηκος μονοπατιου απο ενα σημειο του gameboard σε ενα αλλο οχι προσεγγιστικο μηκος οπως η Manhattan και ευκλειδια. Μια συναρτηση ειναι παραδεκτη οταν δεν υπερβαλλει για την αποσταση 2 σημειων και αυτο ισχυει γιατι η maze distance ισουται με αυτη την τιμη. Συνεπης συνθηκη h(n)-h(n’)<=c(n,a,n’) επισης ισχυει γιατι ακομα και για το το worst case(n,a,n’)=1 (κατι που ισχυει για καθε ακμη pacman) ισχυει h(n)-h(n’)=1 γιατι το

n’ χρειαζεται να κανει μονο ενα βημα για να παει το n και επειτα να χρειαστει h(n) βηματα για να φτασει στον στοχο.
Επιστρεφω τη μεγαλυτερη των αποστασεων γιατι θελω ο astar να αποφευγει θεσεις που ειναι μακρια απο γωνια.αρα προσπαθει να παει στη μεση του gameboard για να ειναι κοντα σε ολα τα corners αλλα οταν πρεπει να κανει expand μακρια απ’τη μεση φτιαχνει με τον ανταγωνισμο των ευρετικων ενα κυκλο με κεντρο την αρχικη θεση του Pacman κανοντας τον να φτασει πρωτα τα corners που ειναι κοντα σε αυτη την αρχικη



foodheuretic:
ειναι ιδια λογικη με το προηγουμενο προβλημα αλλα αντι για 4 στοχους, εχουμε οσους ειναι στο Foodlist. η maze distance παιρνει 5/4 στον autograder αλλα ειναι υπερβολικα αργη
για αυτο χρησιμοποιησα το dictionary ωστε οταν βλεπω ζευγαρι συντεταγμενων του οποιου εχω υπολογιζει στο παρελθον το μηκος διαδρομης το αποθηκευω για να το ανακτω σε O(1) πολυπλοκοτητα αντι να το ξαναυπολογιζω με O(b^d) αφου αυτη η αποσταση δεν προκεται να αλλαξει .
8)
ειναι προφανες οτι το φαγητο που ειναι πιο κοντα στον pacman ειναι αυτο που βρισκεται στο υψηλοτερο επιπεδο σκεφτομενοι οτι η θεση του ειναι το starting state.η bfs μας ταριαζει επειδη εξεταζει πληρως καθε επιπεδο αρχιζοντας απ’τα ψηλοτερα.Για οποιο φαγητο συναντησει πρωτο θα επιστρεψει μονοπατι αρκει μονο να αλλαξει το goal state που μας ζητειται ωστε να τερματιζει η bfs για καθε κουκιδα. Εμεις γραφουμε το προβλημα για μια κουκιδα και η `registerInitialState` το τρεχει μεχρι να μη μεινει καμια αλλη κουκιδα

η εντολη `python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5` μας δινει ενα παραδειγμα οπου δεν επιλεγται η βελτιστη κουκιδα



αλλο ενα κλασσικο παραδειγμα ειναι το:
…….
----------.
…….. .
οπου ο Pacman αντι να φαει την κατω δεξια κουκιδα οταν φτασει κοντα της παει στις πανω και κανει ολοκληρη τη διαδρομη παλι για να την φαει.