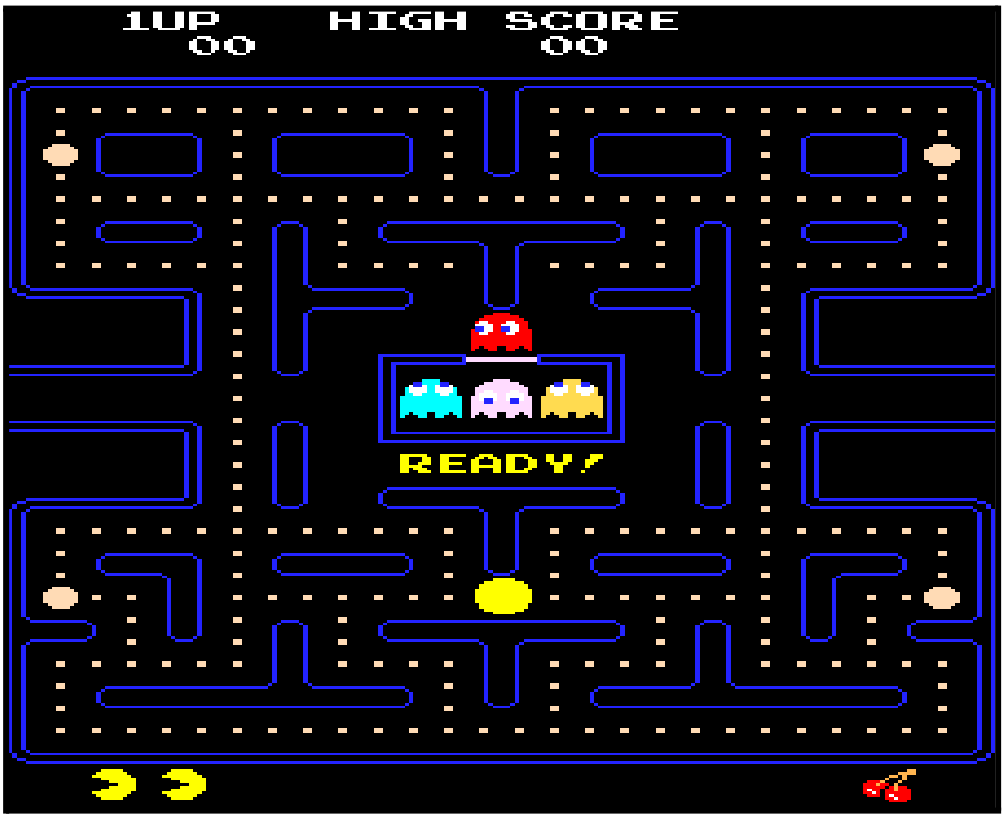
ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΜΑΤΙΚΗΣ

Σκουλούδης Ιωάννης-2021091

1η Εργασία στο μάθημα Τεχνητή Νοημοσύνη



Περιεχόμενα:

Ερώτηση 1:

Εξηγήστε πως αναπαρίσταται ένας κόμβος του δέντρου αναζήτησης στη λύση σας.

Κάθε κόμβος στο δέντρο αναζήτησης αναπαρίσταται ως μια πλειάδα που αποτελείται από το state,δηλαδή την κατάσταση που βρίσκεται στον λαβύρινθο όπου είναι μια πλειάδα με συντεταγμένες x,y και το path που είναι μια λίστα με κινήσεις που πρέπει να κάνει για να βρεθεί στο state.

Ερώτηση 2:

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

Ο DFS υλοποιήθηκε με την χρήση της στοίβας,η οποία θα χρησιμοποιήσει πρώτα τον τελευταίο κόμβο που τις προστέθηκε,ενώ ο BFS υλοποιήθηκε με την χρήση της ουράς που χρησιμοποεί τους κόμβους που προστέθηκαν πρώτοι.

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

Ο StayWestAgent έχει συνάρτηση κόστους 2^x και ο StayEastAgent $\frac{1}{2^x}$.Προκειμένου να ελαχιστοποιήσει το κόστος ο StayWestAgent θα κινηθεί αριστερά καθώς αριστερά μειώνεται το x και επομένως και το κόστος η συνάρτηση κόστους 2^x είναι ανάλογη του x.Έτσι και ο StayEastAgent θα κινηθεί δεξιά για να αυξήσει το x αφου η συνάρτηση κόστους $\frac{1}{2^x}$ είναι αντιστρόφως ανάλογη του x.

Ερώτηση 4:

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

Με την χρήση του **bigMaze** ο UCS απαιτεί 620 και ο A* απαιτεί 549,ενώ με τον **openMaze** ο UCS απαιτεί 682 και ο A* 535

bigMaze:

```
PS F:\Projects\python projects\project_1> python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 549
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
PS F:\Projects\python projects\project_1> python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=ucs
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 210 in 0.0 seconds
Search nodes expanded: 620
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
PS F:\Projects\python projects\project_1>
```

openMaze:

```
PS F:\Projects\python projects\project_1> python pacman.py -l openMaze -z .5 -p SearchAgent -a fn=ucs
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
PS F:\Projects\python projects\project_1> python pacman.py -l openMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 535
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
PS F:\Projects\python projects\project_1>
```

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

Η ευρετική συνάρτηση που υλοποιήθηκε χρησιμοποιεί την απόσταση manhattan για να υπολογίσει την απόσταση μεταξύ της θέσης που βρίσκεται ο pacman και τις θέσεις όλων των κουκίδων και στο τέλος επιστρέφει την μεγαλυτερη απόσταση,επεκτείνοντας 9.000 κόμβους. Είναι αποδεκτή και συνεπής,γιατι δεν ξεπερνά το πραγματικό κόστος για να φτάσει τον στόχο του

Ερώτηση 6:

Πως εξηγείτε τη συμπεριφορά του Pacman στην εκτέλεση του παιχνιδιού q1 παραπάνω; Γιατί ενώ εκτελούμε τον αλγόριθμο MiniMax ο πράκτοράς μας μπορεί να χάνει; Γιατί ο πράκτορας σε ορισμένες περιπτώσεις περιμένει τα φαντάσματα να πλησιάσουν;

Η συμπεριφορά του pacman εξηγείτε με την υλοποίηση του MiniMax.Ο πράκτορας μας μπορεί να χάσει λόγω του βάθους των ενεργειών που μπορεί να κοιτάξει που στην περίπτωση αυτή είναι 1 ή απο την συνάρτηση αξιολόγησης.Προκειμένου να ελαχιστοποιήσει την χειρότερη περίπτωση ο pacman δεν θα κουνηθεί και θα περιμένει να πλησιάσουν τα φαντασματα προκειμένου να εμφανιστεί μια πιο ασφαλέστερη ενέργεια.

Ερώτηση 7:

Όταν ο Pacman βλέπει ότι θα χάσει, προσπαθεί να χάσει το συντομότερο δυνατό. python pacman.py -p MinimaxAgent -l trappedClassic -a depth=3. Εξηγήστε γιατί συμβαίνει αυτό στην περίπτωση του MinimaxAgent . Πότε είναι λάθος η συγκεκριμένη στρατηγική και γιατί;

Στο συγκεκριμένο παράδειγμα φαίνεται πως η ήττα του pacman είναι αναπόφευκτη.Επομένως προκειμένου να ελαχιστοποιήσει την ήττα του θα χάσει το συντομότερο δυνατό για να μην καταλήξει με χαμηλότερο σκορ.Η συγκεκριμένη στρατηγική θα ήταν λάθος αν δεν βρισκόμασταν στον trappedClassic και ο pacman μπορούσε να ξεφύγει πάνω από τις 3 κινήσεις που βλέπει.

Ερώτηση 8:

Εξηγήστε πως λειτουργεί η συνάρτηση αξιολόγησής σας. Επίσης, δοκιμάστε πως λειτουργεί ο πράκτοράς σας σε συνδυασμό με alpha-beta pruning εκτελώντας την εντολή python pacman.py -p AlphaBetaAgent -a evalFn=better,depth=2 -l smallClassic -k 2.Πως διαφέρει η συμπεριφορά του πράκτορα σε σχέση με αυτόν του ερωτήματος 6;

Η συνάρτηση αξιολόγησης μας χρησιμοποιεί την απόσταση manhattan για να υπολογίσει την απόσταση του pacman απο τις κουκίδες,τα φαντάσματα,τις κάψουλες λαμβάνοντας υπόψη και τον χρόνο που τα φαντάσματα θα είναι φοβισμένα επιστρέφοντας το αξιολογούμενο σκορ.Χρησιμοποιώντας τον AlphaBetaAgent σε σχέση με τον MiniMaxAgent του ερωτήματος 6 παρατηρούμε ότι ο AlphaBetaAgent είναι πιο αποδοτικός,καθώς θα αποφύγει καταστάσεις που είναι χειρότερες.