Τεχνητή Νοημοσύνη: Εργασία Πρώτη

Δημήτρης Φωτεινός

AM:1115201700181

Επεξήγηση Κώδικα:

- Q1(**DFS**): Γενικότερα,χρησιμοποιήσα τις οδηγίες που υπήρχαν στις διαφάνειες του μαθήματος για κάθε αλγόριθμο αναζήτησης.Η δομή Fringe στην ερώτηση αυτή,όπως και στα επόμενα ερωτήματα,περιέχει **κυρίως** το σύνολο των κόμβων που πρόκειται να επισκεφθούμε,καθώς και τα **"actions"**,δηλαδή τις κινήσεις που πρέπει να κάνουμε απ'την αρχική κατάσταση για να φθάσουμε στον κόμβο αυτόν.

  Στο συγκεκριμένο ερώτημα, η δομή **Fringe** ήταν μια στοίβα η οποία κάθε φορά "γέμιζε" με τα παιδιά του τωρινού κόμβου που επισκεπτόμασαν.

  Ο κόμβος ο οποίος εξετάζαμε,ήταν και ο τελευταίος ο οποίος εισήλθε στην στοίβα,δηλαδή **LIFO**(για τον λόγο ότι εδώ κάνουμε αναζήτηση σε βάθος).

- Q2(**BFS**): Στο συγκεκριμένο ερώτημα,εργάστηκα ακριβώς με την ίδια λογική όπως πριν,αλλά τώρα η δομή **Fringe** ήταν μια ουρά.

  Ο κόμβος που εξετάζαμε κάθε φορά δηλαδή,ήταν και ο πρώτος ο οποίος εισήλθε,δηλαδή **FIFO** (για τον λόγο ότι εδώ κάνουμε αναζήτηση σε πλάτος).

- Q3(**UCS**): Εδώ,εργαστήκαμε ελαφρώς διαφορετικά,απ'την άποψη των πληροφοριών που συγκρατεί η δομή Fringe.

Απλώς,η ***Fringe*** αποθηκεύει,πέρα απ'την θέση που βρίσκεται ο κάθε κόμβος και τις κινήσεις που χρειάζεται να κάνουμε για να φθάσουμε απ'την αρχική κατάσταση σε αυτόν,το **κόστος** των κινήσεων που χρειάζεται για να φθάσουμε απο έναν κόμβο σε έναν επόμενο.

Εν συνεχεία,η δομή ***Fringe*** είναι υλοποιημένη ως μια Priority Queue, η οποία κάνει ***pop*** τον κόμβο κάθε φορά όπου έχει την μικρότερη προτεραιότητα (δηλαδή το μικρότερο κάθε φορά κόστος).

- Q4(**A***):

  Στο ερώτημα αυτό,εργαστήκαμε όπως και στην UCS, με την διαφορά ότι στο συνολικό κόστος,προσθέσαμε το κόστος της ευριστικής συνάρτησης για τον expaned κόμβο-παιδί του τρέχοντος κόμβου που εξετάζαμε κάθε φορά.

- Q5(**CornersProblem**):

  Στην συγκεκριμένη άσκηση καλούμασταν να "δημιουργήσουμε" εμείς το δικό μας "state" για το Pacman.Σκέφτηκα 2 υλοποιήσεις:

  Καταρχάς σκέφτηκα πως ο στόχος μας (goal) ήταν να φάμε και τα 4 φαγητά που υπήρχαν σε κάθε μια γωνία.Την πληροφορία αυτή θα μπορούσαμε να την βάλουμε σε μια λίστα,όπου κάθε φορά θα κάνουμε push την θέση της γωνίας που έχουμε επισκεφθεί.

  Μια άλλη λογική ήταν να κάνουμε push εξαρχής σε αυτή την λίστα την θέση κάθε γωνίας,και κάθε φορά που επισκεπτόμασταν κάποια,να την κάναμε remove απ'την λίστα.Εγώ επέλεξα να υλοποιήσω την 1η μορφή.

  Άρα,το state μου είναι ένα tuple:

  $$\boldsymbol{state(Node, GoalStates)}$$

  Στην συνάρτηση με τους Successors,δεν κάναμε τίποτα άλλο πέρα απ'το να εισχωρήσουμε στο state του successor κάθε φορά,την θέση του κόμβου,το ποιες (καινούριες και προυπάρχον)γωνίες έχει επισκεφθεί,τις κινήσεις που πρέπει να κάνουμε για να φτάσουμε εκεί,και το κόστος που ήταν σταθερά 1.

- Q6(**CornerHeuristic**):

  Η στρατηγική μου στο ερώτημα αυτό,ήταν να υπολογίσω την απόσταση που είχε ο κόμβος που εξέταζα κάθε φορά,με την απόσταση κάθε φορά που είχε με τις γωνίες που δεν είχε επισκεφθεί.Την μέγιστη από αυτήν την απόσταση(καθώς εμείς μελετάμε την χείριστη περίπτωση), την επέστρεφα ως αποτέλεσμα. (2/3 στο autograder,για κάποιον λόγο).

- Q7(**FoodHeuristic**):

  Στην ουσία,στο συγκεκριμένο ερώτημα εργάστηκα όπως ακριβώς και στο προηγούμενο ερώτημα,μόνο που εδώ υπολόγιζα την κάθε φορά απόσταση του κόμβου απ'το φαγητό που υπάρχει στον χάρτη,και επέστρεφα την μεγαλύτερη μεταξύ τους απόσταση κάθε φορά,όπως ακριβώς και στο Q6.

- Q8(**ClosestDotSearchAgent**):

  Τέλος,εδώ,απλά υλοποιήσαμε το isGoalState στην κλάση AnyFoodSearchProblem,όπου goal για εμάς κάθε φορά ήταν οποιαδήποτε κουκίδα φαγητού και όχι το να φάμε πλήρως όλα τα φαγητά.
  Στην συνάρτηση findPathToClosestDot,το μόνο που χρειάστηκε να κάνουμε ήταν να σκεφτούμε πως κάθε φορά θα πηγαίνουμε στην, "πλησιέστερη" επιλογή.Εύκολα διαπιστώνουμε πως μια "άπληστη" αλλά κατάλληλη αναζήτηση για αυτό το πρόβλημα είναι η BFS,την οποία την είχαμε υλοποιήσει στο Q2.