Σπανάκης Δημήτριος – 1115201800183

Καλησπέρα, σας παραθέτω τα σχόλια μου σχετικά με το project_1 του Pacman:

- *Question 1*
  Η υλοποίηση της DFS έγινε με την χρήση στοίβας όπου ισχύει η προτεραιότητα LIFO. Με την βοήθεια της Stack() εξασφαλίζουμε ότι το τελευταίο node που έγινε expand , θα είναι και το πρώτο που θα ξανακάνει εξασφαλίζοντας την αναζήτηση κατά βάθος. Η γενικότερη υλοποίηση είναι ότι ξεκινάμε από το startNode, το βάζουμε στην Stack(), κάνουμε pop(), και μετά επεκτείνουμε με βάση τα successors του και τα προσθέτουμε πάλι στην Stack(), ενώ συγχρόνως κάνουμε έλεγχο , αν τα έχουμε ήδη επισκεφτεί η και αν αποτελούν κόμβοι λύσης.

- *Question 2*
  Η υλοποίηση του BFS ακολουθεί το ίδιο σκεπτικό (σαν δόμηση κώδικα) με τον DFS, αλλά χρησιμοποιεί Queue (FIFO), ώστε να κάνουμε expand πάντα κατά πλάτος.

- *Question 3*
  Ακολουθεί την ίδια νοοτροπία στην υλοποίηση όπως στα Q1/Q2, αλλά τώρα χρησιμοποιείται ουρά προτεραιότητας με priority το συνολικό κόστος από το startNode στο curNode. Έτσι η priorityQueue κάνει pop() με βάση τα node με το μικρότερο κόστος.

- *Question 4*
  Για το Q4, ενώ είχα κατανοήσει πλήρως τον aStar αλγόριθμο, είχα θέμα με τον autograder οπότε και πειραματίστηκα με το να δομήσω αλλιώς τον κώδικα σε σχέση με τους Q1,Q2,Q3. Το σκεπτικό είναι ότι έχουμε priority queue και το priority προκύπτει από την σχέση f(n) = g(n) + h(n) όπου g(n) το αληθινό κόστος μέχρι τον κόμβο n, και h το χειριστικό. Σε κάθε pop() που γίνεται, ελέγχουμε αν φτάσαμε στο goal state και μετά αν έχουμε ήδη επισκεφτεί τον κόμβο n. Αν τον έχουμε επισκεφτεί, τότε ελέγχουμε αν το κόστος που υπολογίστηκε, είναι μικρότερο από αυτό που έχει αποθηκευτεί στο already visited list. Αν δεν τον έχουμε επισκεφτεί, η ισχύει το παραπάνω, τότε κάνουμε append στο priorityQueue και ύστερα κάνουμε pop() για να προχωρήσουμε στο επόμενο loop.

- *Question 5*
  Για το CornersProblem, πιστεύω ότι είναι αρκετά ευανάγνωστος ο κώδικας. Τώρα για το κομμάτι του getSuccessors, χρησιμοποίησα κομμάτια του βοηθητικού κώδικα που έχει ο Berkeley στα comments του κώδικα, και μετά το σκεπτικό ήταν απλό, αν η επόμενη κίνηση που θέλουμε να κάνουμε δεν βρίσκει σε wall, αλλά αγγίζει κάποια γωνία, την προσθέτουμε στα succesors.

- *Question 6*
  Για την χειριστική συνάρτηση, βρίσκουμε τις μικρότερες Manhattan αποστάσεις από εκεί που βρισκόμαστε μέχρι κάποια από τις γωνίες και το επαναλαμβάνουμε για όλες τις γωνίες

- *Question 7*

  Η food heuristic συνάρτηση που υλοποίησα υπολογίζει το πιο μακρινό foodDot που υπάρχει και χειριστικά μας καθοδηγεί εκεί, γιατί αν το Pacman πάει να φάει το πιο απόμακρο dot , σχεδόν σίγουρα θα φάει και άλλα στο πέρασμα του. Το μόνο θέμα που αντιμετώπισα, είναι ότι ενώ η άσκηση παίρνει 5/4 στον autograder.py, θέλει να περιμένουμε περίπου 20 δεύτερα πριν τελειώσει η εκτέλεση. Δεν μπόρεσα να βρω τον λόγω αυτής της καθυστέρησης.

- *Question 8*

  Για το question 8, το μόνο που έκανα ήταν ότι κάλεσα την uniformCostSearch() του search.py που είχα υλοποιήσει σε προηγούμενο ερώτημα.

Σας ευχαριστώ πολύ για τον χρόνο σας, καλή διόρθωση 😊