Berkley AI Materials – Pacman Project – Part I

Search In Pacman

Ερώτημα 1ο – Υλοποίηση DFS

Η συνάρτηση depthFirstSearch υλοποιήθηκε με τη βοήθεια της δομής Stack του util.py, ως το σύνορο – fringe – όπως έχει παρουσιαστεί στο μάθημα. Έπειτα ακολουθείται η εξής διαδικασία: προστίθενται κόμβοι στη Stack, ελέγχονται τα στοιχεία του μονοπατιού ως προς το αν είναι τελικός κόμβος ή όχι, και προστίθενται στην τελική απάντηση.

Ερώτημα 2ο – Υλοποίηση BFS

Η συνάρτηση breadthFirstSearch έγινε με ακριβώς τον ίδιο τρόπο με την DFS αλλά αντί για Stack, χρησιμοποιήθηκε η δομή Queue του util.py όπως ορίζεται από τη θεωρία.

Ερώτημα 3ο – Υλοποίηση UCS

Η συνάρτηση uniformCostSearch υλοποιήθηκε με αντίστοιχο τρόπο όπως οι ανωτέρω, αλλά με χρήση της δομής Priority Queue του αρχείου util.py, και με την διαφορά ότι υπάρχει προσθήκη της διαδικασίας ενημέρωσης κόστους της εκάστοτε κίνησης σε άλλο κόμβο, καθώς η συνάρτηση βασίζεται στην μετακίνηση – επιλογή node με το μικρότερο δυνατό κόστος.

Ερώτημα 4ο – Υλοποίηση A*

Η συνάρτηση A* βασίστηκε εν πολλοίς στην υλοποίηση της συνάρτησης UCS, με τη διαφορά ότι, όπως απαιτείται από τη θεωρία, η συνάρτηση κόστους προκύπτει από την συνάρτηση κόστους κόμβου (UCS) και την συνάρτηση απόστασης (Best First/Greedy Search), όπως φαίνεται από την γραμμή

```
fringe.push(node, node["cost"] + node["eval"])
```

Ερώτημα 5ο – Εύρεση Γωνιών

Το ερώτημα αναπτύχθηκε στο αρχείο searchAgents.py (βλέπε σειρά 270 και έπειτα). Πρόκειται για την ανάπτυξη του προβλήματος εύρεσης γωνιών με τη χρήση της συνάρτησης breadthFirstSearch.
Η κλάση Corners Problem έχει τον constructor της, όπου αρχικά ορίζονται οι τοίχοι – τα περιθώρια του κόσμου – οι γωνίες, και το αν αυτές έχουν “φαγητό” για το Pacman. Οι επισκευθέντες κόμβοι αρχικοποιούνται ως κενή λίστα και αρχικοποιείται και η αρχική θέση του Pacman.
Λαμβάνεται η Start State με φυσιολογικό τρόπο, καθώς και το αν η θέση είναι Goal State ή όχι στις αντίστοιχες συναρτήσεις με τη βοήθεια λιστών corners ως boolean μεταβλητές.

Για την εύρεση successors της εκάστοτε state, στην αντίστοιχη συνάρτηση, το ζήτημα αντιμετωπίστηκε ξανά με τη λογική λιστών boolean μεταβλητών για τις ποικίλες θέσεις όπου μπορεί να πάει ο Pacman, καθώς και των δυνατών actions (δεν μπορούμε να περάσουμε μέσα από τοίχους). Αξιοποιήθηκε το hint που βρίσκεται στον σχολιασμένο κώδικα από τους δημιουργούς του project ως οδηγός για την υλοποίηση που ακολουθεί.

Ερώτημα 6ο – Εύρεση Γωνιών - Heuristic

Το ερώτημα αναπτύχθηκε στο ίδιο αρχείο (σειρά 373 και έπειτα) με τη βοήθεια της ανωτέρω υλοποιημένης Manhattan απόστασης, και έλεγχο με τη βοήθεια λιστών boolean μεταβλητών για τις states και τις actions του pacman, ώστε να βρίσκεται κάθε φορά η ελάχιστη απόσταση καθώς και το ποιες γωνίες ανήκουν ή όχι στη λίστα και ποιες αποτελούν λύση του προβλήματος.
Σε δοκιμές στον υπολογιστή μου (Linux – Ubuntu 16.04 64-bit) δημιουργείται το λάθος

```
File "/…/search.py", line 208, in aStarSearch
    if not visited.has_key(child[0]):
TypeError: unhashable type: 'list'
```

χωρίς να έχει βρεθεί λύση για αυτό.

Ερώτημα 7ο – Βρώση κάθε Τελείας

Το ερώτημα αναπτύχθηκε στο ίδιο αρχείο (σειρά 495 και έπειτα) με εύρεση τοποθεσιών που περιέχουν τροφή pacman – τελεία – καθώς και υπολογισμό του εκάστοτε κόστους των κινήσεών του.
Με την εκτέλεση στον υπολογιστή μου προκύπτει το εξής:

```
$ python pacman.py -l trickySearch -p AStarFoodSearchAgent
Path found with total cost of 60 in 3.5 seconds
Search nodes expanded: 8470
Pacman emerges victorious! Score: 570
Average Score: 570.0
Scores:        570.0
Win Rate:      1/1 (1.00)
Record:        Win
```

Δηλαδή εξερευνούνται κάτω από 9.000 κόμβοι.

Ερώτημα 8ο – Μη βέλτιστη Αναζήτηση

Για την συνάρτηση findPathToClosestDot, απλώς κλήθηκε η υλοποιημένη BFS για το δοθέν πρόβλημα.

Ο έλεγχος Goal State γίνεται με τον απλό τρόπο βάσει ύπαρξης ή όχι τροφής για τον pacman.