***ΥΣ02 Τεχνητή Νοημοσύνη – Χειμερινό Εξάμηνο 2019-2020***

***Εργασία Πρώτη***

***Berkeley Project 1: Search***

***Γιάννης Δαλιάνης***

***1115201700027***

Question 1: Η συνάρτηση depthFirstSearch καλεί την DFS\_BFS δίνοντάς της μία στοίβα ως δομή υλοποίησης του DFS. Στην αρχή στη στοίβα τοποθετώ τον πρώτο κόμβο του προβλήματος, μαζί με μία κενή λίστα η οποία στο τέλος θα έχει όλες τις κινήσεις και ένα τυχαίο κόστος, έστω 1. Για κάθε κόμβο που δεν έχει επεκταθεί, δηλαδή δε τον έχουμε ήδη επισκεφτεί, βάζουμε όλους τους κληρονόμους του στη στοίβα, εφόσον και αυτοί δεν έχουν ήδη επεκταθεί. Όταν βάζουμε έναν κληρονόμο στη στοίβα, του δίνουμε το σύνολο κινήσεων που πραγματοποιούνται μέχρι να φτάσουμε στον πατέρα του συν την κίνηση από τον πατέρα μέχρι τον συγκεκριμένο κληρονόμο. Αυτή η διαδικασία επαναλαμβάνεται για κάθε κληρονόμο κάθε κόμβου μέχρι να βρούμε τον κόμβο-στόχο και να γυρίσουμε το σύνολο των κινήσεων.

Question 2: Η συνάρτηση breadthFirstSearch καλεί την DFS\_BFS δίνοντάς της μία ουρά ως δομή υλοποίησης του BFS. Η λογική είναι ίδια με την υλοποίηση του DFS, απλά χρησιμοποιείται ουρά αντί για στοίβα.

Question 3: Η συνάρτηση unifrormCostSearh καλεί την Implementation\_withPQ, η οποία χρησιμοποιεί σα δομή υλοποίησης μια ουρά προτεραιότητας που επιστρέφει το αντικείμενο με την ελάχιστη κάθε φορά προτεραιότητα-κόστος (ουρά προτεραιότητας ελαχίστου). Το κόστος του αρχικού κόμβου είναι 0. Η λογική εισαγωγής των κληρονόμων στους επισκεπτόμενους είναι ίδια με την αντίστοιχη λογική της DFS και της BFS, με τη διαφορά ότι κάθε κόμβος-κληρονόμος που δεν έχουμε ήδη επισκεφτεί και εισάγεται στην ουρά, πέρα από ένα σύνολο κινήσεων κρατάει και ένα κόστος, το κόστος μέχρι να φτάσουμε στον πατέρα του συν το κόστος από τον πατέρα μέχρι αυτόν. Η μεταβλητή new\_h χρησιμοποιείται στο ερώτημα 4 και για τον UCS ισούται με 0.