Νικόλαος Δρυμουσης AM:1115201500037

# Project 1 – Search

**Question 1:** Για να λυσουμε αυτο το προβλημα απλως γραφουμε τον αλγοριθμο για αναζητηση σε γραφο και το υλοποιουμε με ενα stack. Καθε state ειναι ενα tuple που περιεχει τη τωρινη θεση και το μονοπατη που πηραμε για να φτασουμε σε αυτο το σημειο ( μονοπατη ειναι μια λιστα απο actions)

**Question 2:** Για τον BFS εργαζομαστε αρκιβως οπως και με τον DFS με τη διαφορα οτι τον υλοποιουμε με τη βοηθεια μια ουρας (queue) και οχι στοιβας.

**Question 3:** Ο UCS μοιαζει με τον BFS με τη διαφορα να ειναι οτι επιλεγει να επεκτεινει τον κομβο με το μικροτερο κοστος για επεκταση.Επομενως θα χρησιμοποιησουμε μια priority queue για την υλοποιηση του η οποια θα δεχεται ενα tuple(α) το οποιο θα περιεχει ενα tuple (β) και εναν αριθμο.
Το tuple(β) θα ειναι περιεχει οτι περιειχε και το tuple για BFS και DFS ενω στο tuple(β) ο αριθμος θα ειναι το συνολικο κοστος του μονοπατιου μεχρι να φτασουμε σε αυτη τη καταστση.
Επειδη ο αλγοριθμος υλοποιειται με priority queue ο κομβος που θα επιλεγεται θα ειναι παντα αυτος με το χαμηλοτερο κοστος.

**Question 4:** Ο A* υλοποιειται οπως και ο UCS (δηλαδη με priority queue) μονο που αυτη τη φορα το κοστος συναθροιζεται απο το κοστος της μεταβασης απο εναν κομβο στο επομενο καθως και απο την αποσταση του επομενου κομβου προς επεκταση απο τη καταστση λυσης.

**Question 5:** Για το ερωτημα 5 εχουμε να συμπληροσουμε τις εξης συναρτησεις:

1. getStartState
2. isGoalState
3. getSuccessors

Τη getStartState τη φτιαχνουμε ετσι ωστε απλα να γυρναει ενα tuple οπου περιεχει την αρχικη θεση του προβληματος και μια αδεια λιστα με τη μορφη tuple.Αυτη η μετατροπη απο λιστα σε tuple γινεται ωστε να μπορουμε να προσθεσουμε τον κομβο στο explored set.Η λιστα αυτη στην ουσια χρησιμευει ωστε να αποθηκευουμε τις γωνιες που εχουμε επισκεφθει και ετσι ωστε με το που φτασει σε ενα απο τα goalstate να μη κολλησει σε αυτη τη γωνια.

Η isGoalState οπως οριστικε το προβλημα θα πρεπει να ειναι να εχουν επισκεφθει ολες οι γωνιες. Αυτο το καταφερνουμε αν το μηκος της λιστας που περιεχεται στο tuple state ειναι ισο με τον αριθμο των γωνιων του προβληματος. Οπως ειπαμε αυτη η λιστα αποθηκευει τις γωνιες που εχουν εξερευνηθει επομενως στη τελευταια γωνια που θα επισκεφθουμε το μηκος της λιστας θα ειναι το ιδιο με τον αριθμο των γωνιων.

Στη getSuccessors βρισκουμε τις επομενες νομιμες κινησεις απο τη καταστση στη οποια βρισκομαστε. Αν μια απο αυτες ειναι γωνια τοτε κανουμε append τη λιστα με τις εξερευνημενες γωνιες στο tuple state αφου πρωτα τη μετατρεψουμε σε tuple.Αλλιως βρισκουμε απλα τους successors και δεν επεκτεινουμε τη λιστα.

**Question 6:** Στο ερωτημα 6 χρησιμοποιω μια ευρετικη συναρτιση η οποια αρχικα δημιουργει ενα set με ολες τις γωνιες που δεν εχουν εξερευνιθει ακομα.Επισης δημιουργει μια κενη λιστα στην οποια θα αποθηκευουμε tuple το οποιο θα περιεχει την αποσταστη και τη θεση της γωνια που ειναι πιο κοντα στη τωρινη καταστση. Στη συνεχεια κανει εναν αριθμο επαναληψεων ισο με τον αριθμο

των γωνιων που δεν εχουν ακομα εξερευνιθει και για καθε επαναλαψη θα εισαγει στη λιστα τη πιο κοντινη γωνια στη τωρινη κατασταση και την αποσταση της. Στη συνεχεια μετατρεπει τη τωρινη κατασταση ωστε να ειναι η γωνια που μολις εξερευνηθηκε, προσθετει σε μια μεταβλητη total την αποσταση της απο τη προηγουμενη κατασταση και επαναλαμβανει τη διαδικασια μεχρι να εχουν εξερευνιθει ολες οι γωνιες. Στο τελος γυρναει τη μεταβλητη total η οποια ειναι το αθροισμα των μικροτερων αποστασεων απο την εκαστοτε θεση σε γωνια.

**Question 7:** Η ευρετικη στο ερωτημα 7 γυρναει την αποσταση του πιο απομακρισμενου φαγητου απο τη τωρινη κατασταση χρησιμοποιωντας τη συναρτηση mazeDistance.

**Question 8:** Το goalState του προβληματος επιτυγχανετε οταν ο pacman τρωει μια κουκιδα. Επισης αφου θελουμε εναν agent που να τρωει greedily τη πιο κοντινη κουκιδα απλα θα γυριζουμε το αποτελεσμα της uniformCostSearch που υλοποιησαμε στο ερωτημα 3.