Τεχνητή Νοημοσύνη:Εργασία Δεύτερη

Δημήτρης Φωτεινός

AM:1115201700181

Επεξήγηση Κώδικα:

Q1:

Στο συγκεκριμένο πρόβλημα σκέφτηκα σχετικά απλά. Υπέθεσα ότι μια καλή συνάρτηση αξιολόγησης θα υπολογίζει κατάλληλα τις παραμέτρους που έχουμε για το συγκεκριμένο πρόβλημα,δηλαδή τα **GhostAgents(φαντάσματα)** και τους στόχους φαγητού του δικού μας Agent (pacman).

Υπάρχουν 2 Trivial περιπτώσεις:

- Η πρώτη όπου δεν υπάρχει άλλη κουκίδα φαγητού προς κατανάλωση,οπότε είμαστε στην περίπτωση που έχουμε νικήσει. Έτσι,επιστρέφουμε ∞.
- Η δεύτερη όπου το φάντασμα είναι αρκετά κοντά μας και κινδυνεύουμε να χάσουμε(απόσταση 1 τετραγώνου μακριά). Έτσι,επιστρέφουμε $-\infty$.

Έπειτα,αν δεν βρισκόμαστε σε καμία απ'τις 2 παραπάνω καταστάσεις,υπολογίζουμε μέσω της ευριστικής συνάρτησης **Manhattan Distance** την απόσταση μεταξύ του πράκτορα μας και του πλησιέστερου φαγητού προς κατανάλωση.

Όσο πιο κοντά είναι το φαγητό,τόσο μεγαλύτερη είναι η τιμή που επιστρέφει η **Evaluation Function**, και όσο πιο μακριά αντίστοιχα,τόσο πιο μικρή είναι η επιστρεφόμενη τιμή της συνάρτησης.

Τέλος,ο δεύτερος παράγοντας που καθορίζει την τιμή που θα επιστρέψει η συνάρτηση είναι η απόσταση των πρακτόρων φαντάσματος απ'την θέση του πράκτορά μας (pacman).Πάλι τώρα,όσο πιο κοντά βρίσκεται ένα φάντασμα στον πράκτορα μας,τόσο μικρότερη θα είναι και η επιστρεφόμενη τιμή. Όσο πιο

μακριά βρίσκεται ένα φάντασμα,τόσο μεγαλύτερη είναι και η τιμή που επιστρέφει η συνάρτηση.

Q2:

Το συγκεκριμένο ερώτημα επιστρέφει τις κινήσεις που πρέπει να κάνει ο agent pacman έτσι ώστε να φτάσει σε κατάσταση νίκης,υλοποιόντας τον αλγόριθμο Minimax

Στο συγκεκριμένο ερώτημα, υλοποιήσαμε τον **Minimax** αλγόριθμο ως εξής:

Δημιουργήσαμε μια νέα συνάρτηση **def Minimax**, η οποία μέσα στο σώμα της περιέχει 2 "υπορουτινές":

- Η μια είναι η **MinValue** η οποία καλέι αναδρομικά την συνάρτηση Minimax. Όταν η Minimax φτάσει σε συνθήκη τερματισμού,δηλαδή όταν έχουμε φτάσει στα **Φύλλα** ,τότε επιστρέφεται η trivial τιμή του EvaluationFunction που έχει το κάθε φύλλο.

- Και η επόμενη είναι η **MaxValue**, η οποία καλέι αναδρομικά και αυτή με την σειρά της την Minimax. Όταν η Minimax φτάσει σε συνθήκη τερματισμού,δηλαδή όταν έχουμε φτάσει στα Φύλλα, τότε επιστρέφεται η trivial τιμή του EvaluationFunction που έχει το κάθε φύλλο.

Γενικότερα,εμπνευστήκαμε απ'τον εξής ψευδοκώδικα και τον οποίον υλοποιήσαμε σε Python με μικρές τροποποιήσεις συμβατότητας σχετικά με τις παραμέτρους του προβλήματος:

```
function minimax(position, depth, maximizingPlayer)
   if depth == 0 or game over in position
      return static evaluation of position

   if maximizingPlayer
      maxEval = -infinity
      for each child of position
         eval = minimax(child, depth - 1, false)
         maxEval = max(maxEval, eval)
      return maxEval

   else
      minEval = +infinity
      for each child of position
         eval = minimax(child, depth - 1, true)
         minEval = min(minEval, eval)
      return minEval
```

Με την διαφορά ότι τα ορίσματα που χρησιμοποιήσαμε στην Minimax συνάρτηση ήταν: **def Minimax(gameState,Depth,agentIndex**όπου:

- position=gameState: Το Node στο οποίο βρισκόμαστε κάθε φορά,δηλαδή ένα απ'τα παιδιά ενός συγκεκριμένου agentIndex (το ποιο απ'τα παιδιά του πράκτορα που επιλέγουμε κάθε φορά εξαρτάται απο το ποια θα είναι τα actions(κινήσεις) που θα του δώσουμε).

- depth=Depth: Το βάθος στο οποίο βρισκόμαστε κάθε φορά. Η συγκεκριμένη παράμετρος είναι ύψιστης σημασίας καθώς αυτή θα είναι και η συνθήκη τερματισμού μας.

- maximizingPlayer=agentIndex: Στην ουσία το συγκεκριμένο όρισμα είναι αυτό το οποίο θα μας κάνει να αποφανθούμε για το ποια υπορουτίνα θα καλέσουμε μετά. Στο συγκεκριμένο παράδειγμα, αν agentIndex==0 τότε έχουμε να κάνουμε με τον Pacman, και αν agentIndex!= τότε έχουμε να κάνουμε με φαντάσματα.

Q3:

Το συγκεκριμένο ερώτημα επιστρέφει τις κινήσεις που πρέπει να κάνει ο agent pacman έτσι ώστε να φτάσει σε κατάσταση νίκης,υλοποιόντας τον αλγόριθμο AB Pruning

Το συγκεκριμένο ερώτημα,όπως ξέρουμε, είναι μια επέκταση του Αλγορίθμου **Minimax**. Το **ΑΒ Κλάδεμα** είναι μια πολύ χρήσιμη επέκταση του προηγούμενο αλγορίθμου,καθώς μειώνει αρκετά το πλήθος των επαναλήψεων, επειδή με μια συγκεκριμένη συνθήκη "κόβει" αρκετούς κόμβους τους οποίους δεν χρειάζεται να τους επισκεφθούμε.

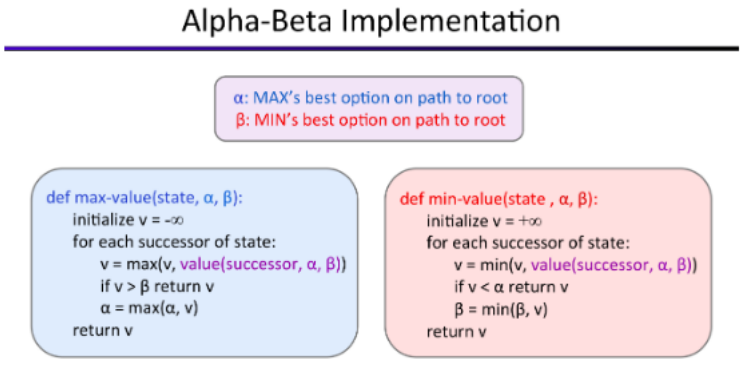
Πιο συγκεκριμένα,κρατήσαμε ίδια την "βάση" του προγράμματος,δηλαδή και πάλι έχουμε 1 συνάρτηση η οποία καλείται αναδρομικά, την **def Minimax**. Αυτό το οποίο αλλάζει απλά είναι η εισαγωγή 2 επιπλέον παραμέτρων στο πρόγραμμα, α και β.

Έχουμε δηλαδή:

def Minimax(gameState,Depth,alpha,beta,agentIndex):

Οι υπορουτίνες **MaxValue και MinValue** οι οποίες βρίσκονται μέσα στο σώμα της συνάρτησης Minimax καλούν και πάλι αναδρομικά την συνάρτηση μέσα στην οποία βρίσκονται,με την διαφορά ότι αυτή τη φορά χειρίζονται κατάλληλα τις παραμέτρους α και β,έτσι ώστε να "διακόπτουν" τις επαναλήψεις στο σημείο που χρειάζεται κάθε φορά.

Για άλλη 1 φορά, κάναμε χρήση του παρακάτω ψευδοκώδικα,τον οποίο και υλοποιήσαμε σε Python με τις κατάλληλες τροποποιήσεις για να είναι συμβατό με το πρόβλημα που είχαμε:



Q4:

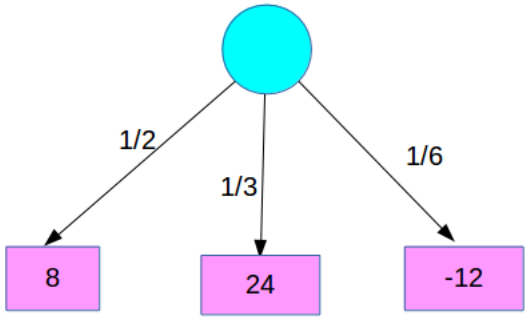
***Το συγκεκριμένο ερώτημα επιστρέφει τις κινήσεις που πρέπει**

να κάνει ο agent pacman έτσι ώστε να φτάσει σε κατάσταση νίκης,υλοποιόντας τον αλγόριθμο Expectimax*

Όπως γνωρίζουμε,το συγκεκριμένο ερώτημα είναι παρόμοιο με τον Minimax αλγόριθμο,με την διαφορά ότι στην συγκεκριμένη περίπτωση έχουμε και "τυχαίους" κόμβους.Αναλυτικότερα, εδώ μπορεί τα παιδιά ενός state, να είναι πιθανό να εμφανιστούν με διαφορετική **πιθανότητα** το καθένα.

Άρα, αυτό το οποίο θα επιστρέψει το EvaluationFunction στην προκειμένη περίπτωση,είναι η συνάρτηση αξιολόγησης κάθε (τυχαίου)παιδιού-κόμβου,πολλαπλασιασμένη επί την πιθανότητα να συμβεί το αντίστοιχο ενδεχόμενο.

Ας δούμε ένα σχετικό παράδειγμα για να καταλάβουμε καλύτερα την περίπτωση με την οποία έχουμε να ασχοληθούμε:



Εδώ,παρατηρούμε πως η τιμή που θα πάρει ο κόμβος αυτός είναι:

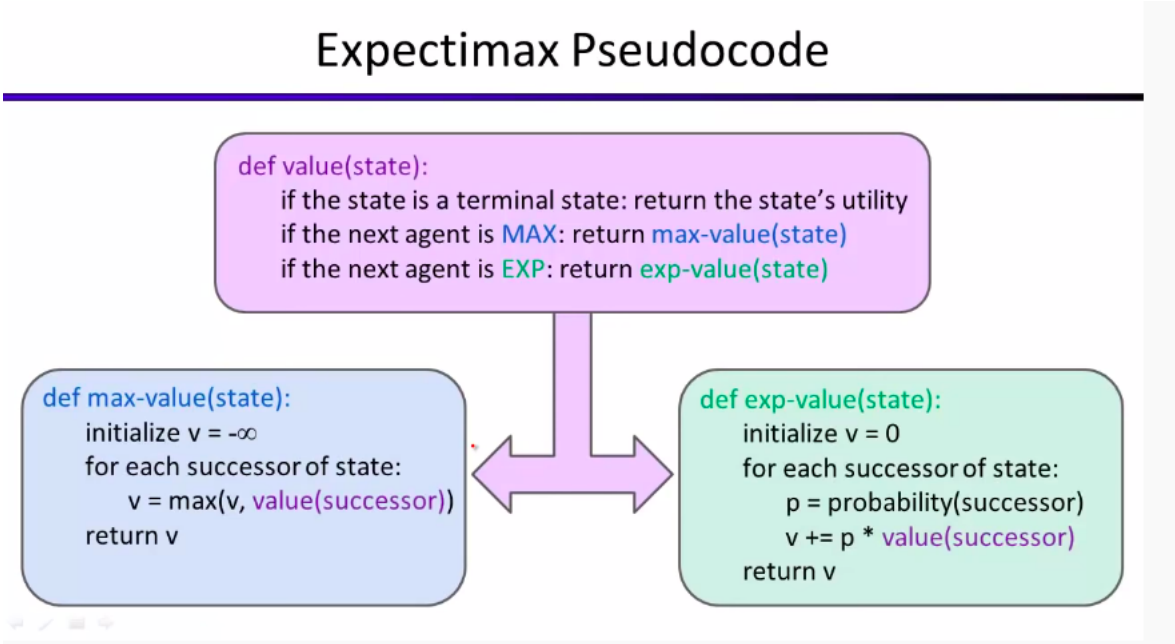
$$\sum_{i=1}^{3} Eval(i) * P(i)$$

Η συνάρτηση Eval,είναι η τιμή EvaluationFunction που θα έχει το κάθε παιδί i, και όπως καταλαβαίνουμε,η συνάρτηση P είναι η πιθανότητα του κάθε παιδιού i.

Άρα,σε μαθηματικούς όρους θα είναι:

$$\frac{1}{2} * 8 + \frac{1}{3} * 24 + \frac{1}{6} * (-12) = 10$$

Για να γυρίσουμε τώρα σε "προγραμματιστικούς" όρους, η μορφή ψευδοκώδικα που ακολουθήσαμε είναι η εξής:



Να σημειωθεί ότι επειδή έχουμε ισοπίθανες περιπτώσεις για την εμφάνιση των παιδιών-κόμβων, η τιμή probability είναι ίση με τον λόγο της μονάδας προς το πλήθος των επόμενων έγκυρων "κινήσεων" (κόμβων-παιδιών) του τρέχοντος κόμβου, δηλαδή αν **Childs=παιδιά** ,τότε θα είναι: Probability=$\frac{\mathbf{1}}{\mathbf{Childs}}$

Όπως και στις προηγούμενες 2 ερωτήσεις, έχουμε την ίδια **βάση** στον κώδικα: Την συνάρτηση **def Expectimax**, η οποία περιέχεις στο σώμα τις 2 υπορουτίνες, την "ψευδοσυνάρτηση" MaxValue, η οποία είναι ίδια όπως ακριβώς υλοποιήθηκε και στην Minimax περίπτωση (ερώτημα Q3) , και την νέα "ψευδοσυνάρτηση" ExpectValue, η οποία επιστρέφει την τιμή του κάθε κόμβου που εξετάζει,και εμπεριέχει την πιθανότητα των κόμβων "παιδιού" που εξετάζουμε(αν έχει).

Q5:

Στο συγκεκριμένο πρόβλημα,κληθήκαμε να δημιουργήσουμε μια "καλύτερη" συνάρτηση αξιολόγησης, με βάση τα:

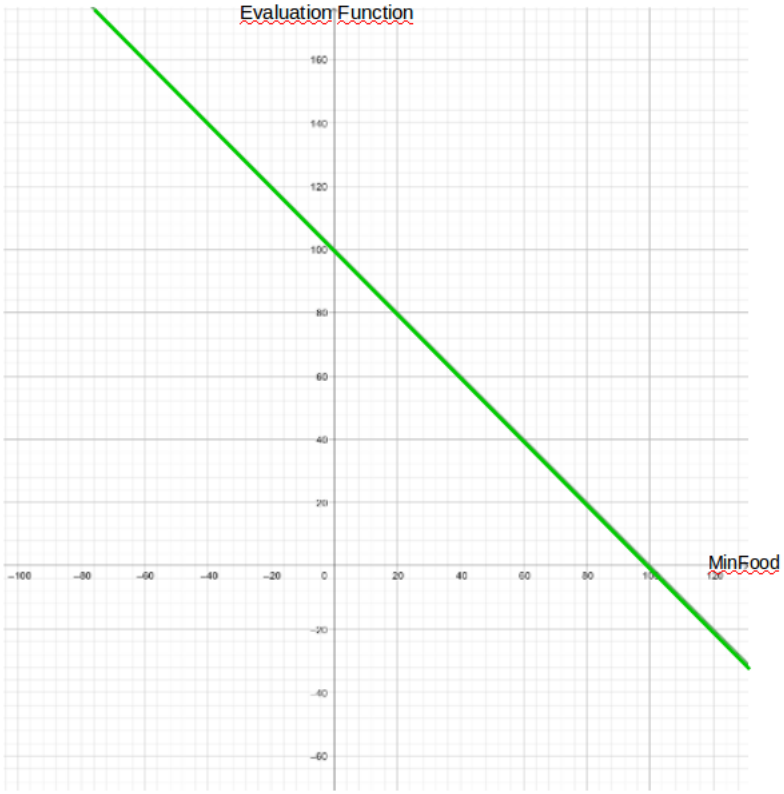
- Τις μεγάλες κουκίδες φαγητού, ή αλλιώς "κάψουλες"

- Την απόσταση μεταξύ των φαντασμάτων και τις περιπτώσεις όπου τα φαντάσματα αυτά δεν είναι άτρωτα αλλά τρώγοντας τις κάψουλες να μπορούν να είναι "φαγώσιμα" απ'τον πράκτορα Pacman

- Και φυσικά,την απόσταση μας απ'τις κανονικές κουκίδες φαγητού.

Στην προσπάθεια να δημιουργήσω την συγκεκριμένη συνάρτηση,αρχικά σκέφτηκα το εξής "κόλπο":

Να δοκιμάσω να επιστρέφω με **αρνητικό** πολλαπλασιαστή την απόσταση μεταξύ του πράκτορα μας και του κοντινότερου δυνατού φαγητού.



Η γραφική παράσταση που βλέπουμε στο παρακάτω σχήμα, είναι η συνάρτηση

$$\mathbf{f(x) = \lambda - x}$$

Όπου στο πρόγραμμά μας αυτό αναπαρίσταται ώς:

- λ = currentgameState.getScore()
- x = FoodCheck

Άρα,σκέφτηκα πως, όσο πιο "κοντά(μικρό x)" είναι το φαγητό στον πράκτορα μας,τόσο πιο "καλή" θα είναι η συνάρτηση αξιολόγησης,κάτι το οποίο βγάζει νόημα καταρχάς απ'την γραφική παράσταση και σκεπτόμενοι τα όσα έχουμε πει μέχρι τώρα για τις συναρτήσεις αξιολόγησης είναι απολύτως λογικό.

Έτρεξα το πρόγραμμα αρχικά για να δω αν λειτουργεί και αν επιστρέφει "καλές τιμές" η συνάρτηση αξιολόγησης, σκεπτόμενος έπειτα να "βάλω στο παιχνίδι" και τις υπόλοιπες παραμέτρους στις οποίες αναφερθήκαμε παραπάνω.

Προς έκπληξη μου η συνάρτηση επέστρεφε **6/6**(!),χωρίς να έχω βαλεί καν τις άλλες παραμέτρους. Αποφάσισα να το αφήσω ως έχει μιας και ο ελεγκτής μας "autograder" μας επέτρεψε να πάρουμε όλους τους δυνατούς πόντους.

*Επιπλέον σχόλιο: Με μια παραπάνω ματιά που έριξα και είδα απ'τον προσομοιωτή,όταν ο Pacman είναι αρκετά μακριά απ'το φαγητό επιστρέφει αρκετά μικρές τιμές,που σημαίνει ότι στην συγκεκριμένη περίπτωση η συνάρτηση αξιολόγησης δεν είναι "καλή" και πρέπει να δράσει αλλιώς. (Αυτό το οποίο το κάνει να κινείται εν τέλει είναι το φάντασμα το οποίο έρχεται αρκετά κοντά του (μόλις 1 τετράγωνο μακριά)).