# Τεχνητή Νοημοσύνη
# Εργασία 2
# *Pacman PDF*

Κωνσταντίνος Χαϊδεμένος
*sdi*2200262

Να σημειωθεί οτί για κάθε ερώτημα υπάρχουν σχόλια στον κώδικα όπου εξηγείτε οριακά κάθε γραμμή. Σε αυτό το *pdf* θα πούμε την γενική ιδέα υλοποίησης κάθε ερωτήματος. Τις σχεδιαστικές μας επιλογές αν θέλετε...

## $Q1$ - $Reflex\ Agent$

Στη συνάρτηση $evaluationfunction(..)$ κάναμε μία αλλαγή μόνο στη σειρά 73 του έτοιμου κώδικα, όπου προσθέσαμε την λειτουργία $asList()$ στην συνάρτηση $getFood()$ της $successorGameState$. Η αλλαγή αυτή έγινε για να έχουμε τις θέσεις των διαθέσιμων φαγητών του *pacman* σε μορφή λίστας.

Ο κώδικας που γράψαμε περιέχει 2 ελέγχους:

- Αρχικά βρίσκει την κοντινότερη απόσταση φαγητού από τον *pacman* και έπειτα ελέγχει αν κοντά στη συγκεκριμένη θέση φαγητού υπάρχει φάντασμα. Αν υπάρχει, ελέγχει την αμέσως επόμενη κοντινότερη απόσταση μέχρι να βρεθεί θέση φαγητού που δεν "άπειλείται".
- Έπειτα ελέγχει την απόσταση του *pacman* από τα φαντάσματα και αν είναι πολύ κοντινή τότε απορρίπτει την συγκεκριμένη *successor state*.

Στο τέλος επιστρέφει το *evaluation score* ανάλογα με τις τιμές των παραπάνω δύο παραγόντων.

## *Q2 - MinimaxAgent*

Για την υλοποίηση της *MinimaxAgent* θα χρησιμοποιήσουμε 2 *helper* συναρτήσεις τις *max* και *min value*.

- **μαξ βαλιου**
  Η συνάρτηση αυτή υπολογίζει την ΜΑΞ τιμή του *agent* 0, ο οποίος είναι πάντα ο *pacman*.

  Επιστρέφει ένα *tuple* που περιέχει την μεγαλύτερη τιμή *evalScore* που μπορεί να πετύχει ο *pacman* από την τρέχουσα κατάσταση που βρίσκεται καθώς και την κίνηση που αντιστοιχεί σε αυτή τη τιμή.

  Η μαξ βαλιου λειτουργεί υπολογίζοντας την MIN τιμή του *agent* που παίζει αμέσως μετά για κάθε δυνατή κίνηση από την τρέχουσα κατάσταση του *pacman*. Την MIN τιμή την υπολογίζει καλώντας την μιν βαλιου...

- **μιν βαλιου**
  Αυτή η συνάρτηση υπολογίζει την MIN τιμή των φαντασμάτων ( δηλαδή *agents* 1,2,3... ).

  Επιστρέφει επίσης ένα *tuple* το οποίο όμως περιέχει την μικρότερη τιμή *evalScore* που μπορεί να πετύχει το συγκεκριμένο φάντασμα και την αντίστοιχη κίνηση.

  Λειτουργεί υπολογίζοντας την ΜΑΞ τιμή του *pacman*, αν είναι το τελευταίο φάντασμα που παίζει, αλλιώς υπολογίζοντας την MIN τιμή του αμέσως επόμενου φαντάσματος που παίζει. Τις τιμές αυτές τις υπολογίζει είτε καλώντας τον εαυτό της με όρισμα το επόμενο φάντασμα, είτε καλώντας την μαξ βαλιου.

Όπως είναι φανερό οι μιν βαλιου και μαξ βαλιου λειτουργούν *recursively*. Στην αρχή και των δύο συναρτήσεων γίνεται ένας έλεγχος για το αν αυτή η κλήση της συνάρτησης έχει γίνει με κατάσταση στόχο ή δεν υπάρχουν άλλες νόμιμες κινήσεις ή απλά ξεπεράσαμε το επιτρεπτό βάθος αναζήτησης. Αν ο έλεγχος αυτός επιστρέψει *True*, τότε λήγει ο βρόχος και μια μια οι τιμές επιστρέφουν στις αντίστοιχες αναδρομικές κλήσεις τους.

## $Q3$ - *Alpha Beta Pruning*

Στο συγκεκριμένο ερώτημα βολευτήκαμε με τις μαξ βαλιου και μιν βαλιου συναρτήσεις από την $Q2$.

Οι αλλαγές σε αυτές αποτελούν:

- τα ορίσματα στα οποία προστέθηκαν οι τιμές των *alpha* και *beta* οι οποίες αρχικοποιούνται σε $-\infty$ και $\infty$ αντίστοιχα
- έναν έλεγχο σε κάθε συνάρτηση για το αν θα κάνουμε *prune* το συγκεκριμένο *branch*.
  Δηλαδή αν το μέγιστο *evalScore* είναι μεγαλύτερο του *beta* ή αν το ελάχιστο είναι μικρότερο του *alpha*.
- Επίσης μια τελευταία αλλαγή είναι η αναβάθμιση (αν χρειάζεται) του *alpha* και του *beta* σε κάθε επανάληψη του *for* που ελέγχει κάθε δυνατή κίνηση.

## $Q4$ - *ExpectimaxAgent*

Βολευτήκαμε ΞΑΝΆ με τις μαξ βαλιου και μιν βαλιου του ερωτήματος $Q2$....

Συγκεκριμένα η μαξ βάλιου είναι **ακριβώς ίδια κόπι πάστε!!!**.
Ωστόσο η μιν βαλιου χρειαζόταν κάποιες αλλαγές:

Η ιδέα του ερωτήματος είναι πως ο *pacman* θα παίζει ενάντια στα φαντασματάκια τα οποία αυτά αυτή τη φορά παίζουν με τυχαία επιλογή των κινήσεων τους. Η αλλιώς με *suboptimal* αντιπάλους φαντάσματα.

Επομένως η αλλαγή στην μιν βαλιου θα είναι ο υπολογισμός του *evalScore*.
Πλέον δε χρειαζόμαστε την κίνηση που θα μας φέρει το βέλτιστο *evalScore* για το κάθε φάντασμα. Χρειαζόμαστε την αναμενόμενη τιμή του *evalScore* αν λάβουμε υπόψην μας κάθε πιθανή κίνησή τους (μιας και αυτή θα επιλεγεί τυχαία...).

Δηλαδή απλώς υπολογίζουμε το κλάσμα με αριθμιτή το άθροισμα των τιμών *evalScore* κάθε πιθανής κίνησης του φαντάσματος και με παρονομαστή το πλήθος των τιμών αυτών.

Αυτός ο αριθμός θα είναι το πρώτο στοιχείο του *tuple* που επιστρέφει η μιν βαλιου και θα τον χρησιμοποιήσει η μαξ βαλιου κανονικά στις συγκρίσεις της για την εύρεση της βέλτιστης κίνησης του *pacman*.

Το δεύτερο στοιχείο είναι κενό *None*. Υπάρχει μόνο γιατί η μαξ βαλιου συγκρίνει *tuples* μεταξύ τους και βαριόμασταν να κάνουμε τη μετατροπή στο να συγκρίνει $float$ αφού μπορούσαμε να κάνουμε ένα πανέμορφο κόπι πάστε...<3

Θα διαφωνούσε κανείς πως η προσπάθεια που θα καταβάλει κάποιος για να αλλάξει δύο μεταβλητές μιας σύγκρισης σε μια μέθοδο από *tuple* $(float, action)$ σε απλά $float$, είναι σημαντικά μικρότερη από το να γράψει όλη αυτή την επεξήγηση σε αυτό το **πανέμορφο** *pdf*...

Όμως στη ζωή υπάρχουν πολλοί τέτοιοι άνθρωποι και εμείς δείχνοντας ανωτερότητα δεν θα τους δώσουμε σημασία.

## $Q5$ - $betterEvaluationFunction$

Σε αυτό το ερώτημα καλούμαστε να βρούμε μια ακόμα καλύτερη *evaluation* $function$ από την **ήδη πάρα πολύ καλή** συνάρτηση που είχαμε βρεί στο πρώτο ερώτημα $Q1$.

Έχοντας αλλάξει μόνο την $successorGameState$ μεταβλητή σε $currentGameState$ έτσι ώστε οι έλεγχοι να γίνονται για την τρέχουσα κατάσταση, αν τρέξουμε την *evaluation* $function$ που φτίαξαμε στο ερώτημα $Q1$ συγκεντρώνουμε τους 5 από τους 6 πόντους του *autograder*!!!

Επομένως είμαστε πάρα πολύ καλοί προγραμματιστές και το αποδείξαμε;;;;

# ΟΧΙ!

πρέπει να συλλέξουμε **όλες** τις μονάδες.

Χρειάζεται μόνο μια μικρή βελτίσωη....

Στην ακόμα-καλύτερη-από-την-ήδη-πάρα-πολύ-καλή συνάρτηση (η ονομασία θέλει δουλειά) θα προσθέσουμε ένα απλό σύστημα επιβράβευσης ή τιμώρησης του *pacman* για το αν βρίσκεται κοντά σε φαντάσματα ή όχι. Πιο συγκεκριμένα:

- Αν ο *pacman* είναι κοντά σε φαντάσματα και είναι φοβισμένα, τότε κερδίζει πόντους.
- Αν ο *pacman* είναι κοντά σε φαντάσματα και δεν είναι φοβισμένα τότε χάνει πόντους - περισσότερους σε απόλυτη τιμή από όσους θα κέρδιζε αν ήταν φοβισμένα.

  Αυτό το κάνουμε γιατί τα φαντάσματα είναι φοβισμένα μόνο αν ο *pacman* φάει το λουκούμι με τις υπερδυνάμεις και μόνο για ένα περιορισμένο χρονικό διάστημα. Συνεπώς οι καταστάσεις αυτές είναι λίγες, άρα σε γενικές γραμμές καλό θα ήταν ο *pacman* και γενικά ο οποιοσδήποτε, να αποφεύγει τα θανατηφόρα φαντάσματα.
- Κατά συνέπεια όσο ο *pacman* είναι αρκετά μακριά από τα φαντάσματα τότε κερδίζει λίγους πόντους. Πολύ λίγους σε σχέση με το αν φάει ένα, αλλά όπως λένε οι σοφοί:
  *better safe, than sorry*

Τέλος - *The End - La Fin*