Οικονομικό Πανεπιστήμιο Αθηνών
Σχολή Επιστημών και Τεχνολογίας της Πληροφορίας
Τμήμα Πληροφορικής

Τεχνητή Νοημοσύνη

1η Ομάδα Ασκήσεων

Διδάσκων: Ίων Ανδρουτσόπουλος

Μελάς – Φραγκίσκος Παναγιώτης
Α.Μ. : p3190118

Μουσικός Κωνσταντίνος
Α. Μ. : p3180111

Παπαδοπούλου Σταυρούλα
Α. Μ. : p3190160

Ημερομηνία Παράδοσης Εργασίας: 05/12/2021

Άσκηση 1

Τρόποι χρήσης, δυνατότητες και αρχιτεκτονική του προγράμματος:

Κατά την εκκίνηση του προγράμματος ζητάται από το χρήστη ο αριθμός των ιεραποστόλων και κανίβαλων - που είναι ο ίδιος - καθώς και η μέγιστη χωρητικότητα ατόμων στη βάρκα. Στη συνέχεια δημιουργούμε την αρχική κατάσταση τύπου stateNode[2] κατά την οποία τόσο όλοι οι ιεραπόστολοι και κανίβαλοι, όσο και η βάρκα βρίσκονται στην αριστερή όχθη του ποταμού και

ένα νέο αντικείμενο τύπου stateNode στο οποίο εκχωρείται η τιμή του αποτελέσματος του αλγορίθμου A* με κλειστό σύνολο[1] που έχει λάβει ως είσοδο την αρχική κατάσταση. Ελέγχουμε αν ο αλγόριθμος βρίσκει λύση. Αν δεν βρει εμφανίζουμε το αντίστοιχο μήνυμα, αλλιώς εμφανίζουμε ολόκληρο το μονοπάτι του δέντρου για την εύρεση της λύσης από την αρχική προς την τελική κατάσταση, καθώς και το χρόνο που έκανε ο αλγόριθμος να το βρει.

Αλγόριθμος A* με κλειστό σύνολο[1] : Ο αλγόριθμος κατασκευάζει μια λίστα η οποία αναπαριστά το μέτωπο και ένα hashSet που αναπαριστά το κλειστό σύνολο. Με τη συνάρτηση run ελέγχουμε αν η αρχική κατάσταση είναι τελική. Σε περίπτωση που είναι την επιστρέφει αλλιώς την προσθέτει στο μέτωπο. Έπειτα όσο υπάρχει κάποια κατάσταση στο μέτωπο, αφαιρούμε από το μέτωπο τον πρώτο κόμβο, ελέγχοντας αν είναι τελικός και αν είναι τον επιστρέφουμε. Στην αντίθετη περίπτωση ελέγχουμε αν ο κόμβος υπάρχει ήδη στο κλειστό σύνολο και αν δεν είναι προσθέτουμε αυτόν στο κλειστό σύνολο, δημιουργούμε τα παιδιά του με την getChildren και τα προσθέτουμε στο μέτωπο. Τέλος ταξινομούμε(scoreSorter[3]) την ανανεωμένη λίστα του μετώπου με βάση το score ώστε κάθε φορά στην πρώτη θέση της λίστας να αντιστοιχεί η καλύτερη δυνατή κατάσταση.

stateNode[2] : Για την αναπαράσταση ενός στιγμιότυπου του προβλήματος χρησιμοποιούμε ένα πίνακα 6 θέσεων, όπου στην πρώτη θέση βάζουμε τον αριθμό των ιεραποστόλων αριστερά, στη δεύτερη θέση τον αριθμό των κανιβάλων στα αριστερά, στη τρίτη θέση των μέγιστη χωρητικότητα της βάρκας, στην τέταρτη θέση τον αριθμό των ιεραποστόλων στα δεξιά, στην πέμπτη θέση τον αριθμό των κανιβάλων στα δεξιά και στην έκτη θέση το αν η βάρκα είναι αριστερά. Ως score, δηλαδή την συνάρτηση f, ορίζουμε το άθροισμα του αποτελέσματος της ευρετικής συνάρτησης h (το πηλίκο του αθροίσματος των ιεραποστόλων και κανίβαλων από τα αριστερά με τη μέγιστη χωρητικότητα στην βάρκα) και της συνάρτησης g (η απόσταση του εκάστοτε κόμβου από τη ρίζα).

Η συνάρτηση getChildren δημιουργεί μία κενή λίστα για τα παιδιά του κόμβου που εξετάζουμε. Με τη βοήθεια δύο βρόγχων εξετάζουμε όλους τους πιθανούς αποδεκτούς συνδυασμούς κανίβαλων και ιεραποστόλων σε σχέση

με την χωρητικότητα της βάρκας. Αρχικά εξετάζουμε την περίπτωση της βάρκας να βρίσκεται αριστερά του ποταμιού. Απορρίπτουμε τις περιπτώσεις:

i) Ο αριθμός των ιεραποστόλων και κανίβαλων που επρόκειτο να περάσουν απέναντι να είναι μηδενικός (ισοδυναμεί με την κατασκευή του γονέα)

ii) Ο συνολικός αριθμός ιεραποστόλων και κανίβαλων που επρόκειτο να περάσουν απέναντι να ξεπερνάει τον μέγιστο αριθμό ατόμων που χωράει στην βάρκα.

iii) Ο αριθμός των ιεραποστόλων/ κανιβάλων που που επρόκειτο να περάσουν απέναντι να μην υπερβαίνει το πλήθος τους στα αριστερά.

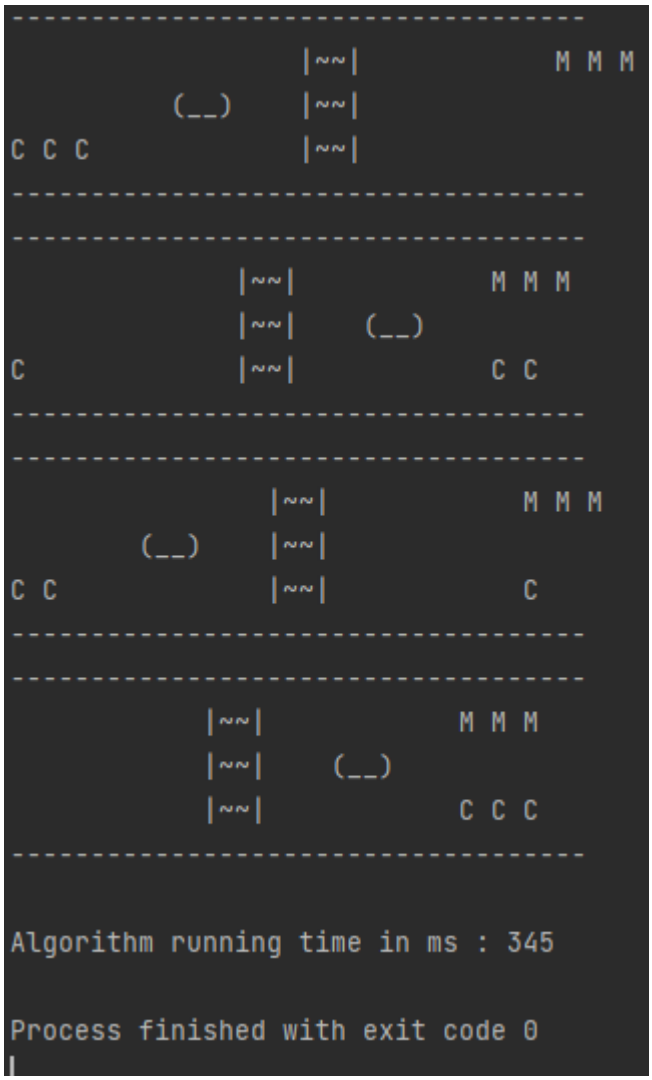
Έπειτα ελέγχουμε τους περιορισμούς, το πλήθος των κανίβαλων να μην ξεπερνάει το πλήθος των ιεραποστόλων της αριστερής και της δεξιάς όχθης αντίστοιχα μετά την μετακίνηση της βάρκας. Αν οι περιορισμοί τηρούνται τότε δημιουργούμε μια νέα κατάσταση-παιδί,θέτουμε την προηγούμενη ως πατέρα, βρίσκουμε το score της και την προσθέτουμε στην λίστα. Η αντίστοιχη διαδικασία γίνεται στην περίπτωση που η βάρκα βρίσκεται στην δεξιά πλευρά του ποταμού. Η συνάρτηση επιστρέφει τον πίνακα συμπληρωμένο με τα παιδιά του εξεταζόμενου κόμβου-πατέρα.

scoreSorter[3]: Η κλάση scoreSorter υλοποιεί μια αύξουσα ταξινόμηση για αντικείμενα τύπου stateNode με κριτήριο το πεδίο τους score(@override).

<u>Μέθοδος Τεχνητής Νοημοσύνης</u>

Το πρόγραμμα χρησιμοποιεί τον αλγόριθμο αναζήτησης A* με κλειστό σύνολο ώστε να βρει το βέλτιστο μονοπάτι προς μια τελική κατάσταση (λύση) για το γνωστό πρόβλημα των κανιβάλων και ιεραποστόλων.

```
Give me the number of missionaries and cannibals :
3
Give me the number of maximum people in boat :
2
Give me the maximum steps of the algorithm :
10000000
-----------------------------------
M M M             |~~|
          (__)    |~~|
C C C             |~~|
-----------------------------------
-----------------------------------
M M M             |~~|
                  |~~|    (__)
C                 |~~|           C C
-----------------------------------
-----------------------------------
M M M             |~~|
          (__)    |~~|
C C               |~~|           C
-----------------------------------
-----------------------------------
M M M             |~~|
                  |~~|    (__)
                  |~~|           C C C
-----------------------------------
-----------------------------------
M M M             |~~|
          (__)    |~~|
C                 |~~|           C C
-----------------------------------
-----------------------------------
M             |~~|             M M
              |~~|    (__)
C             |~~|             C C
-----------------------------------
-----------------------------------
M M             |~~|             M
        (__)    |~~|
C C             |~~|             C
-----------------------------------
-----------------------------------
                |~~|             M M M
                |~~|    (__)
C C             |~~|             C
-----------------------------------
```



Άσκηση 2

Τρόποι χρήσης, δυνατότητες και αρχιτεκτονική του προγράμματος:

Κατά την εκκίνηση του προγράμματος, η κλάση main ζητάει από το χρήστη ως είσοδο έναν ακέραιο αριθμό - έστω n - που αντιπροσωπεύει τον αριθμό των βασιλισσών, που πρέπει να τοποθετηθούν σε μια n*n σκακιέρα (είσοδος προγράμματος). Το πρόγραμμα τρέχει τον γενετικό αλγόριθμο για είσοδο n, φτίαχνοντας ένα αντικείμενο τύπου GeneticAlgorithm και τυπώνει μια λύση, η οποία αντιστοιχεί στο χρωμόσωμα - κατάσταση “solution” και τον χρόνο που χρειάστηκε στον αλγόριθμο να οδηγηθεί σε αυτήν.

[1]GeneticAlgorithm : Ο γενετικός αλγόριθμος δέχεται ως είσοδο τον αριθμό των βασιλισσών και κατασκευάζει 2 ArrayList τα οποία αντιστοιχούν στον πληθυσμό χρωμοσωμάτων (αντικείμενα τύπου Chromosome[2]) και τους αντίστοιχους ακεραίους fitness. Το μήκος της κάθε λίστας είναι ο τυχαία δοσμένος populationSize. Η μέθοδος run δέχεται ως ορίσματα το populationSize (σύνολο πληθυσμού) , mutationProbability (πιθανότητα μετάλλαξης ανά ζεύγος παιδιών), maxSteps και minFitness που λειτουργούν ως όρια/ φράγμα για την επιστροφή λύσης απο τον αλγόριθμο. Ο αλγόριθμος αρχικοποιεί τις λίστες πληθυσμού και καταλληλότητας με populationSize τυχαίες καταστάσεις. Κατασκευάζει μια λίστα με τον νέο πληθυσμό, στην οποία τοποθετεί σε κάθε επανάληψη τα 2 νέα παιδιά, που προκύπτουν ως εξής : Παίρνουμε δύο fitness και το αντίστοιχα χρωμοσώματά τους και βάσει ενός τυχαίου αριθμού διαχωρισμού, καλούμε τη συνάρτηση reproduce που επιστρέφει τον πίνακα με τα παιδιά (για τους γονείς που δόθηκαν ως παράμετροι). Εξετάζουμε το ενδεχόμενο μετάλλαξης και τα παιδιά που προκύπτουν τα προσθέτουμε στη λίστα του νέου πληθυσμού. Ανανεώνουμε τη λίστα πληθυσμού με τη νέα γενιά που προκύπτει, ταξινομούμε ώστε τα καταλληλότερα χρωμοσώματα να βρίσκονται στην αρχή της λίστας. Ανανεώνουμε τη λίστα καταλληλότητας και επιστρέφουμε την “πιθανότερη” λύση.

[2]Chromosome : Κατασκευάζει ένα πίνακα ακεραίων μεγέθους n. Η κάθε θέση του πίνακα αντιστοιχεί σε μία στήλη της σκακιέρας - μοναδική για κάθε βασίλισσα - και το στοιχείο της κάθε θέσης στην αντίστοιχη γραμμή. Η μέθοδος calculateFitness ελέγχει το ενδεχόμενο να υπάρξουν δύο βασίλισσες στην ίδια γραμμή ή ίδια διαγώνιο κρατώντας το ανάλογο άθροισμα των “μη-απειλών”. Η μέθοδος mutate υλοποιεί την μετάλλαξη ενός τυχαίου γονιδιόματος με τυχαία πιθανότητα.Τέλος τυπώνεται η κατάσταση “solution” υλοποιημένη σε μια εικονική σκακιέρα, το αντίστοιχο χρωμόσωμα και το αντίστοιχο fitness.

Μέθοδος Τεχνητής Νοημοσύνης

Το πρόγραμμα χρησιμοποιεί έναν γενετικό αλγόριθμο που μιμείται τον φυσικό μηχανισμό εξέλιξης.Δημιουργεί πληθυσμό καταστάσεων(στιγμιότυπο σκακιέρας) που παριστάνονται ως χρωμοσώματα.Με μια συνάρτηση καταλληλότητας αξιολογεί τις καταστάσεις του πληθυσμού ώστε να αποφασίσει τους γονείς που θα επιλέξει για την επόμενη γενιά πληθυσμού.
Το πρόγραμμα τρέχει σε εκθετική πολυπλοκότητα $O(n^n)$ καθώς έχει εμφωλευμένους βρόγχους.
Ο αλγόριθμος δεν είναι πλήρης ούτε βέλτιστος καθώς μπορεί να μην υπάρχει κατάσταση με κατάλληλο fitness για τελική κατάσταση ή να μην καταφέρνει να καταλήξει σε τελική κατάσταση λόγω μεταλλάξεων η λανθασμένων προσεγγίσεων καταλληλότητας ή να μην προλάβει να επιστρέψει λύση στο περιορισμένο πλήθος βημάτων.
Ο αλγόριθμος επιστρέφει προσεγγιστικά μία κατάσταση “κοντά” σε τελική.

Παραδείγματα για συγκεκριμένες εισόδους:

n = 4 :

```
Give the number of queens :
4
Chromosome : |2|0|3|1|, Fitness : 6
-----------------------------------
| |Q| | |
| | | |Q|
|Q| | | |
| | |Q| |
-----------------------------------
Run time : 15 ms

Process finished with exit code 0
```

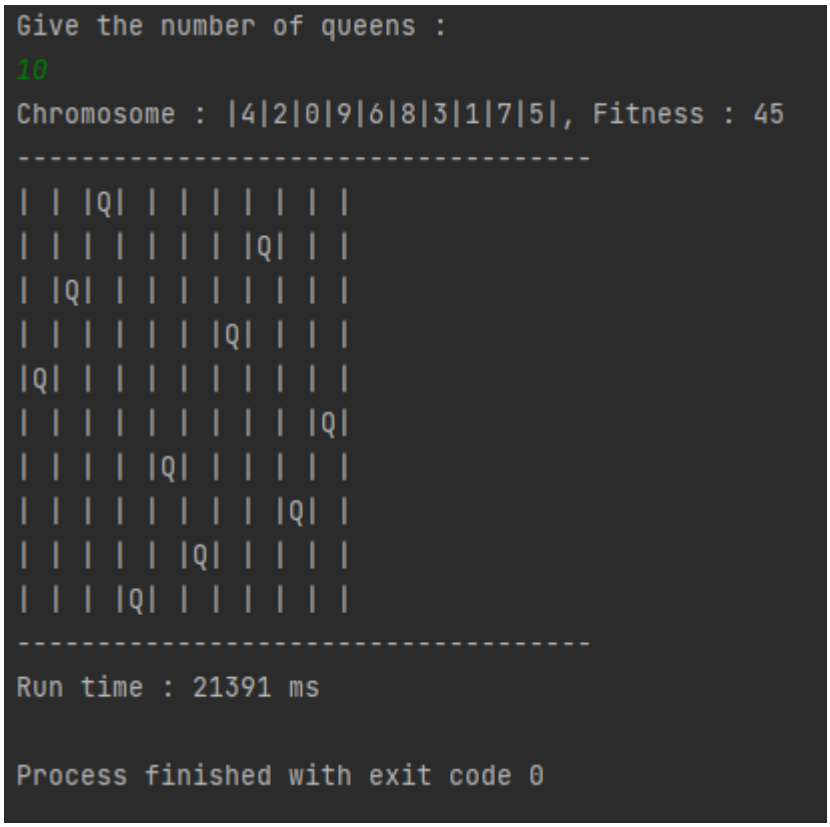
n=6:

```
Give the number of queens :
6
Chromosome : |3|0|4|1|5|2|, Fitness : 15
-----------------------------------
| |Q| | | | |
| | | |Q| | |
| | | | | |Q|
|Q| | | | | |
| | |Q| | | |
| | | | |Q| |
-----------------------------------
Run time : 51 ms

Process finished with exit code 0
```

n=8:

```
Give the number of queens :
8
Chromosome : |7|3|0|2|5|1|6|4|, Fitness : 28
-----------------------------------
| | |Q| | | | | |
| | | | | |Q| | |
| | | |Q| | | | |
| |Q| | | | | | |
| | | | | | | |Q|
| | | | |Q| | | |
| | | | | | |Q| |
|Q| | | | | | | |
-----------------------------------
Run time : 151 ms

Process finished with exit code 0
```

n=10:


Give the number of queens :
10
Chromosome : |4|2|0|9|6|8|3|1|7|5|, Fitness : 45
		Q						
						|Q|	
	Q							
					|Q|		
Q								
								|Q
			|Q|				
							|Q|
				|Q|			
		|Q|					
Run time : 21391 ms
Process finished with exit code 0