Γιαννακοπούλου Σταυρούλα - 3220027

Σταμαδιάνου Μαρία - 3220194

Αναφορά για το Μέρος Α της 1ης εργασίας στην Τεχνητή Νοημοσύνη

Το πρόβλημα των Ιεραποστόλων και Κανίβαλων είναι ένα πρόβλημα τεχνητής νοημοσύνης, όπου ο στόχος είναι να μεταφερθούν N ιεραπόστολοι και N κανίβαλοι από τη μία πλευρά του ποταμού στην άλλη, χρησιμοποιώντας μια βάρκα χωρητικότητας M. Είναι σημαντικό να τηρηθούν κάποια συγκεκριμένα κριτήρια ώστε να καταφέρουν να περάσουν με ασφάλεια στην απέναντι όχθη και οι δυο ομάδες:

1. Σε καμία όχθη οι κανίβαλοι δεν είναι περισσότεροι από τους ιεραποστόλους
2. Εντός της βάρκας το πλήθος των ιεραποστόλων πρέπει να είναι μεγαλύτερο ή ίσο με αυτό των κανιβάλων
3. Η βάρκα πρέπει να έχει τουλάχιστον 1 και το πολύ M άτομα σε κάθε ταξίδι.
4. Ο συνολικός αριθμός των διασχίσεων δεν πρέπει να υπερβαίνει το K, έναν προκαθορισμένο μέγιστο αριθμό (στην περίπτωσή μας αυτό τον αριθμό τον δίνει ο χρήστης).

Ο κώδικας προσπαθεί να βρει λύση στο πρόβλημα χρησιμοποιώντας τον αλγόριθμο **A*** με κλειστό σύνολο, έναν ευρετικό αλγόριθμο αναζήτησης που βρίσκει την πιο αποδοτική λύση, εφόσον αυτή υπάρχει.

Ο κώδικας στο αρχείο MC.java είναι το κύριο πρόγραμμα που χειρίζεται τις εισόδους (ελέγχει αν οι παράμετροι εισόδου N, M και K που δίνονται από τον χρήστη είναι έγκυροι), την υλοποίηση του αλγορίθμου A* και τελικά επιστρέφει τη λύση αν υπάρχει ή ότι δεν υπάρχει λύση εντός του περιθωρίου K ή ενημερώνει τον χρήστη σε περίπτωση που για τα δεδομένα που έδωσε δεν υπάρχει ποτέ λύση. Το M μπορεί να είναι οποιοσδήποτε αριθμός μεγαλύτερος ή ίσος του 1. Το N είναι ο αριθμός των κανιβάλων και ισούται με τον αριθμό των ιεραποστόλων δηλαδή στο σύνολο είναι 2N. Ισχύουν οι εξής περιορισμοί: α)αν η βάρκα μπορεί να χωρέσει 2 άτομα, τότε δυο ζευγάρια (που το κάθε ζευγάρι αποτελείται από έναν ιεραπόστολο και έναν κανίβαλο) απαιτούν 5 ταξίδια ενώ με 4 ή περισσότερα ζευγάρια, το πρόβλημα δεν έχει λύση. Αν η βάρκα χωράει 3 άτομα, τότε μέχρι και 5 ζευγάρια δηλαδή 5 κανίβαλοι και 5 ιεραπόστολοι μπορούν να διασχίσουν το ποτάμι.

Για την υλοποίηση του A* χρησιμοποιείται μια ουρά προτεραιότητας για το open set και ένα HashSet για το closed set. Χρησιμοποιεί κλειστό σύνολο για να αποθηκεύει τις καταστάσεις που έχει εξετάσει για να μην τις επανεξετάσει ενώ η ουρά προτεραιότητας διατηρεί τις υποψήφιες προς εξέταση καταστάσεις με βάση την τιμή της f . Κάθε μια από τις καταστάσεις αξιολογείται με βάση τη συνάρτηση κόστους: f = g + h, όπου g το πραγματικό κόστος μέχρι την παρούσα κατάσταση που μετριέται με τον αριθμό διασχίσεων και h ευρετική εκτίμηση του κόστους για να φτάσει στην τελική κατάσταση. Δημιουργεί παιδιά για κάθε κατάσταση και απορρίπτει τις μη έγκυρες ή επαναλαμβανόμενες καταστάσεις ενώ ελέγχει αν η παρούσα κατάσταση είναι τελική (όλοι στην άλλη όχθη) και τερματίζει αν βρεθεί λύση. Αν η λύση που βρεθεί ξεπερνά το εκάστοτε K, τυπώνει ανάλογο μήνυμα.

Επίσης με τη χρήση της getTotalTime υπολογίζει τον χρόνο που χρειάζεται το πρόγραμμα για να τρέξει σε νανοδευτερόλεπτα.

Ο κώδικας στο αρχείο State.java περιέχει την κλάση State που αναπαριστά μια μεμονωμένη κατάσταση του προβλήματος, παρέχοντας τις ιδιότητες:

- Αριθμός ιεραποστόλων και κανίβαλων σε κάθε όχθη.
- Θέση της βάρκας (αριστερά ή δεξιά).
- Χωρητικότητα της βάρκας.
- Ευρετικές τιμές:
  - g: Πραγματικό κόστος μέχρι την κατάσταση.
  - h: Ευρετική εκτίμηση για την επίτευξη του στόχου.
  - f=g+h: Συνολικό κόστος.

Υπάρχουν και οι εξής μεθόδοι:

evaluate(): Υπολογίζει τον αριθμό των ανθρώπων στην αριστερή όχθη και τη συνάρτηση h που εκφράζει το ελάχιστο πλήθος διαδρομών που χρειάζεται για να μεταφερθούν όλοι, με βάση τον αριθμό των ατόμων που απομένουν στην αριστερή όχθη και τη χωρητικότητα της βάρκας.

!!Το αποτέλεσμα στρογγυλοποιείται προς τα πάνω χρησιμοποιώντας τη Math.ceil, επειδή αν υπάρχουν άνθρωποι που δεν χωράνε σε μια πλήρη διαδρομή, θα χρειαστεί ακόμα μία.

print():Εκτυπώνει την τρέχουσα κατάσταση κάθε φορά, δηλαδή τον αριθμό ατόμων δεξιά και αριστερά του ποταμού και το που βρίσκεται τη συγκεκριμένη στιγμή η βάρκα.

getChildren():

1) Δημιουργεί και επιστρέφει μια λίστα με όλες τις έγκυρες καταστάσεις (children states) που μπορούν να προκύψουν από την τρέχουσα κατάσταση στο πρόβλημα των ιεραποστόλων και των κανιβάλων λαμβάνοντας υπόψη τους περιορισμούς που θέσαμε.

2)Η νέα κατάσταση συνδέεται με την τρέχουσα με την setFather, ενημερώνεται το g και το f και προστίθεται στη λίστα

isFinal(): Ελέγχει αν έχουν μεταφερθεί όλοι οι ιεραπόστολοι και οι κανίβαλοι στη δεξιά όχθη και η βάρκα βρίσκεται επίσης εκεί.

equals(): Εξασφαλίζει ότι οι καταστάσεις είναι μοναδικές για σωστή χρήση σε HashSet συγκρίνοντας τα πεδία δυο αντικειμένων της κλάσης state

hashCode(): εξασφαλίζει ότι θα ορίζεται μοναδικά το αντικείμενο

compareTo(): συγκρίνει δυο αντικείμενα βασισμένη στο heuristic score τους

Συνοπτικά, λοιπόν, ο αλγόριθμος A* χρησιμοποιεί ευρετική αναζήτηση για να ελαχιστοποίησει τον αριθμό των καταστάσεων που εξετάζονται και αν αυτή είναι συνεπής και αποδεκτή, θα βρει τη βέλτιστη λύση στο πρόβλημα κανιβάλων και ιεραποστόλων. Τελικά, επιστρέφει τη διαδρομή από την αρχική στην τελική κατάσταση αν αυτή δεν ξεπερνά τον μέγιστο αριθμό διασχίσεων που έχει οριστεί αρχικά και εφόσον τα αριθμητικά δεδομένα που αφορούν τους ιεραπόστολους (και τους κανίβαλους αντίστοιχα) και το μέγεθος της βάρκας είναι έγκυρα.