

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΙΑ 5^{ου} ΕΞΑΜΗΝΟΥ 2024

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

ΜΕΛΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

Αθανάσιος-Ζώης Δημητρακόπουλος (Α.Μ. 3220039)

Ανδρέας Λάμπος (Α.Μ. 3220105)

ΔΙΔΑΣΚΩΝ

Γιων Ανδρουτσόπουλος

1^η Εργασία

Για την 1^η εργασία επιλέξαμε το πρόβλημα των κανιβάλων και ιεραποστόλων.

1. Εισαγωγή

Το project υλοποιεί τη λύση του προβλήματος "Missionaries and Cannibals" χρησιμοποιώντας τον αλγόριθμο A* (A-star). Περιλαμβάνει τρία βασικά αρχεία:

- AStarSolver.java: Υλοποιεί τον αλγόριθμο A* και τις απαραίτητες λειτουργίες.
 - Main.java: Το κεντρικό πρόγραμμα που διαχειρίζεται την είσοδο χρήστη και την εκτέλεση της επίλυσης.
 - State.java: Περιγράφει την κατάσταση του προβλήματος και παρέχει τις βασικές μεθόδους για την επαλήθευση της εγκυρότητας και του στόχου.
-

Κλάση AStarSolver

Αυτή η κλάση είναι ο πυρήνας του αλγορίθμου A*. Περιέχει τα δεδομένα του προβλήματος, τη λογική για την αναπαράσταση κόμβων και τις μεθόδους για την εκτέλεση του αλγορίθμου.

1. Ιδιότητες της Κλάσης

- **totalMissionaries**: Ο συνολικός αριθμός των ιεραποστόλων.
- **totalCannibals**: Ο συνολικός αριθμός των κανίβαλων.
- **boatCapacity**: Η χωρητικότητα της βάρκας.
- **maxCrossings**: Το μέγιστο επιτρεπτό πλήθος διασχίσεων του ποταμού.

Αυτά τα δεδομένα ορίζονται κατά την αρχικοποίηση μέσω του constructor.

2. Εσωτερική Κλάση Node

- Ο **Node** αναπαριστά έναν κόμβο του αλγορίθμου. Περιέχει:
 - **state**: Την κατάσταση που αντιστοιχεί στον κόμβο (από την κλάση State).
 - **parent**: Τον γονικό κόμβο από τον οποίο προήλθε.
 - **cost**: Το κόστος για να φτάσουμε σε αυτόν τον κόμβο.
 - **heuristic**: Την ευρετική τιμή που υπολογίζει πόσο "μακριά" είμαστε από τον στόχο.

Αυτή η δομή επιτρέπει τη σύνδεση των κόμβων και την ανακατασκευή της διαδρομής μετά την εύρεση της λύσης, άρα και την ανάκτηση της λύσης μας.

3. Μέθοδος AStarSolve()

Αυτή είναι η κύρια μέθοδος του αλγορίθμου A*.

Λεπτομέρειες λειτουργίας:

1. Αρχικοποίηση:

- Δημιουργείται μια ουρά προτεραιότητας (**PriorityQueue**) που ταξινομεί τους κόμβους με βάση το άθροισμα κόστους και ευρετικής τιμής.
- Η αρχική κατάσταση (**initialState**) και ο αρχικός κόμβος (**initialNode**) εισάγονται στην ουρά.

2. Επεξεργασία Κόμβων:

- Ανακτάται ο κόμβος με το χαμηλότερο κόστος από την ουρά.
- Ελέγχεται αν η κατάσταση του κόμβου είναι ο στόχος. Αν ναι, επιστρέφεται η διαδρομή προς αυτόν τον κόμβο (μέσω της **buildPath**).

3. Δημιουργία Παιδιών:

- Καλείται η **getChildren** για να δημιουργήσει όλες τις πιθανές επόμενες καταστάσεις.
- Ελέγχεται η εγκυρότητα και το κόστος για κάθε παιδί. Αν είναι αποδεκτό, προστίθεται στην ουρά.

4. Τερματισμός:

- Αν η ουρά εξαντληθεί χωρίς να βρεθεί λύση, επιστρέφεται **null**.

4. Μέθοδος `getChildren()`

Παράγει όλες τις έγκυρες καταστάσεις που μπορούν να προκύψουν από μια δεδομένη κατάσταση.

Βασικά Σημεία:

- Η θέση της βάρκας καθορίζει τη "κατεύθυνση" της κίνησης.
- Δημιουργούνται όλοι οι δυνατοί συνδυασμοί ιεραποστόλων και κανίβαλων που μπορούν να επιβιβαστούν στη βάρκα.
- Ελέγχεται η εγκυρότητα της νέας κατάστασης μέσω της `isValid()` της κλάσης `State`.

5. Μέθοδος `heuristic()`

Υπολογίζει την ευρετική τιμή για μια κατάσταση.

- **Ευρετική συνάρτηση:**
 - Εκτιμά τον ελάχιστο αριθμό διασχίσεων που απαιτούνται για να μεταφερθούν όλοι οι χαρακτήρες στη δεξιά πλευρά.
 - Υπολογίζεται ως:

$$(\text{state.missionariesLeft} + \text{state.cannibalsLeft} + \text{boatCapacity} - 1) / \text{boatCapacity}$$

Αυτή η ευρετική αποτελεσματική, καθώς διασφαλίζει την καλή απόδοση του A^* .

6. Μέθοδος `buildPath()`

Ανακατασκευάζει τη διαδρομή προς τον αρχικό κόμβο.

- Ξεκινά από τον τελικό κόμβο.
 - Περνά από κάθε γονικό κόμβο (parent) μέχρι να φτάσει στον αρχικό κόμβο.
 - Η διαδρομή αντιστρέφεται για να εμφανιστεί με τη σωστή σειρά.
-
-

Κλάση `State`

Η κλάση `State` αναπαριστά την τρέχουσα κατάσταση του προβλήματος, περιλαμβάνοντας πληροφορίες για τους χαρακτήρες (ιεραποστόλους, κανίβαλους) σε κάθε πλευρά του ποταμού, καθώς και τη θέση της βάρκας. Παρέχει επίσης εργαλεία για την επικύρωση και τη σύγκριση καταστάσεων.

1. Ιδιότητες της Κλάσης

- **missionariesLeft**: Ο αριθμός των ιεραποστόλων στην αριστερή πλευρά.
- **cannibalsLeft**: Ο αριθμός των κανίβαλων στην αριστερή πλευρά.
- **missionariesRight**: Ο αριθμός των ιεραποστόλων στη δεξιά πλευρά.
- **cannibalsRight**: Ο αριθμός των κανίβαλων στη δεξιά πλευρά.
- **boatOnLeft**: Λογική τιμή που δηλώνει αν η βάρκα βρίσκεται στην αριστερή πλευρά.

Αυτά τα πεδία περιγράφουν πλήρως μια συγκεκριμένη κατάσταση του προβλήματος.

2. Κατασκευαστής

Ο κατασκευαστής (State) δέχεται τις παραπάνω τιμές ως παραμέτρους και τις αποθηκεύει στα αντίστοιχα πεδία.

3. Μέθοδος isValid()

Ελέγχει αν η κατάσταση είναι έγκυρη, βασισμένη στους εξής κανόνες:

1. Σε κάθε πλευρά του ποταμού, ο αριθμός των κανίβαλων δεν μπορεί να υπερβαίνει τον αριθμό των ιεραποστόλων, εκτός αν δεν υπάρχει κανένας ιεραπόστολος.

2. Ο αριθμός των χαρακτήρων σε κάθε πλευρά δεν μπορεί να είναι αρνητικός.

Η μέθοδος εξασφαλίζει ότι η κατάσταση είναι ασφαλής, αποτρέποντας το να "φαγωθούν" οι ιεραπόστολοι.

4. Μέθοδος isGoalState()

Ελέγχει αν η τρέχουσα κατάσταση είναι η τελική (στόχος).

Κριτήρια:

- Όλοι οι ιεραπόστολοι και κανίβαλοι βρίσκονται στη δεξιά πλευρά.
- Η αριστερή πλευρά είναι άδεια.

Αυτή η μέθοδος χρησιμοποιείται από τον αλγόριθμο A^* για να ελέγξει αν έχει βρεθεί λύση.

5. Μέθοδος equals()

Υπερκαλύπτει (Override) τη μέθοδο equals της κλάσης Object για τη σύγκριση δύο καταστάσεων.

Κριτήρια Ισοδυναμίας:

- Οι αριθμοί των ιεραποστόλων και κανίβαλων σε κάθε πλευρά είναι ίδιοι.
- Η θέση της βάρκας είναι η ίδια.

Αυτή η μέθοδος είναι κρίσιμη για τη σωστή λειτουργία δομών όπως το HashMap και το Set.

6. Μέθοδος hashCode()





Υπερκαλύπτει (Override) τη μέθοδο hashCode για τη δημιουργία ενός μοναδικού κωδικού για κάθε κατάσταση.

Αυτή η μέθοδος επιτρέπει την αποτελεσματική αποθήκευση και αναζήτηση καταστάσεων σε δομές δεδομένων.

7. Μέθοδος toString()

Επιστρέφει μια συμβολοσειρά που περιγράφει την τρέχουσα κατάσταση με κατανοητό τρόπο.

Παραγωγή Παραδείγματος:

"Left side - Missionaries: 3 , Cannibals: 2  | Right side - Missionaries: 0 , Cannibals: 0  | Boat is on the left side"

Αυτή η μέθοδος χρησιμοποιείται για την παρουσίαση των καταστάσεων κατά την εκτύπωση της λύσης για ευκολότερη κατανόηση από τον χρήστη.

Κλάση Main

Η κλάση Main λειτουργεί ως το σημείο εκκίνησης του προγράμματος. Αναλαμβάνει να λάβει τις παραμέτρους από τον χρήστη, να καλέσει τον αλγόριθμο A* και να παρουσιάσει τα αποτελέσματα.

1. Ιδιότητες

- **N**: Ο αριθμός των ιεραποστόλων και κανίβαλων (ίδιος αριθμός για τις δύο ομάδες).
- **M**: Η χωρητικότητα της βάρκας.
- **K**: Το μέγιστο επιτρεπτό πλήθος διασχίσεων.

Αυτές οι μεταβλητές καθορίζονται από την είσοδο του χρήστη.

2. Μέθοδος readInput()

Υπεύθυνη για τη λήψη και την επικύρωση της εισόδου του χρήστη.

Βήματα:

1. Ζητά τον αριθμό των ιεραποστόλων και κανίβαλων (**N**).
 2. Ζητά τη χωρητικότητα της βάρκας (**M**).
 3. Ζητά το μέγιστο επιτρεπτό πλήθος διασχίσεων (**K**).
 4. Επικυρώνει ότι οι τιμές είναι θετικές.
-

3. Μέθοδος `main()`

Η κύρια μέθοδος του προγράμματος.

Βήματα Εκτέλεσης:

1. Κλήση του `readInput`:

Διασφαλίζει ότι οι παράμετροι εισόδου έχουν καθοριστεί σωστά.

2. Δημιουργία Αλγορίθμου `A*`:

Δημιουργείται ένα αντικείμενο `AStarSolver` με τις παραμέτρους που έδωσε ο χρήστης.

3. Εκτέλεση της Επίλυσης:

Η μέθοδος `AStarSolve` εκτελεί τον αλγόριθμο και επιστρέφει τη λύση (ή `null` αν δεν υπάρχει λύση).

4. Υπολογισμός Χρόνου:

Μετράται ο χρόνος εκτέλεσης με

5. Εμφάνιση Αποτελεσμάτων:

- Αν δεν υπάρχει λύση:

"No solution found with the given variables."

- Αν βρεθεί λύση, εκτυπώνεται η διαδρομή, δηλαδή η ακολουθία των καταστάσεων που περιέχει η λύση μας.

6. Εκτύπωση Χρόνου Εκτέλεσης:

Ο χρόνος εκτέλεσης παρουσιάζεται σε δευτερόλεπτα.

Συμπέρασμα

Το project προσφέρει μια πλήρη και αποδοτική υλοποίηση του προβλήματος "Missionaries and Cannibals", αξιοποιώντας τον αλγόριθμο A* για την εύρεση της βέλτιστης λύσης. Η κλάση AStarSolver υλοποιεί τον αλγόριθμο, η State περιγράφει την κατάσταση του προβλήματος, και η Main ενσωματώνει τη διαχείριση εισόδου/εξόδου.

Το σύστημα είναι επεκτάσιμο και μπορεί να προσαρμοστεί για πιο σύνθετες εκδοχές ή παραλλαγές του προβλήματος.
