Χειμερινό εξάμηνο 2020-2021

Προγραμματισμός Ταυτοχρονισμού & Ασφάλεια λογισμικού

Αναφορά Εργασίας:

«Παράλληλος αλγόριθμος για την επίλυση του προβλήματος των Ν-Βασιλισσών»

Στέφανος Καραμπέρας
ΑΕΜ 2910

Εισαγωγή

Σκοπός της παρούσας αναφοράς είναι η περιγραφή της υλοποίησης του αλγορίθμου των «N-Βασιλισσών» (N-Queens), σε μία έκδοση που αξιοποιεί τον παραλληλισμό για την επιτάχυνση της εύρεσης των λύσεων του προβλήματος.

Στο πρόβλημα των «N-Βασιλισσών» το ζητούμενο είναι η τοποθέτηση N πλήθους βασιλισσών σε μία σκακιέρα διαστάσεων N x N, έτσι ώστε καμία βασίλισσα να μην είναι σε θέση να απειλήσει (ή αντίστοιχα, να απειληθεί) από κάποια άλλη στο ταμπλό.

Το πρόβλημα παρουσιάζει ενδιαφέρον λόγω της αυξανόμενης πολυπλοκότητας του αλγορίθμου, που παρουσιάζεται με την αύξηση του αριθμού N. Μία παράλληλη υλοποίηση, σαν αυτή που προτείνεται στην παρούσα αναφορά, δύναται να μειώσει σημαντικά τον απαιτούμενο χρόνο εκτέλεσης του αλγορίθμου επίλυσης για τον υπολογισμό όλων των λύσεων που προκύπτουν για δεδομένο N.

Τέλος, αξίζει να σημειωθεί ότι το πρόβλημα των «N-Βασιλισσών» δεν διαθέτει εκ φύσεως λύση για τις τιμές N = 2 και N = 3, συνεπώς οι παραπάνω τιμές αποκλείονται μέσω ελέγχου εγκυρότητας της εισόδου του χρήστη.

Με δεδομένο τον επαρκή σχολιασμό του κώδικα ως προς το ρόλο των μεθόδων που δημιουργήθηκαν αλλά και τον τρόπο λειτουργίας τους, η παρούσα αναφορά θα εστιάσει κυρίως στην αφαιρετική παρουσίαση της λογικής του κώδικα.

Σύντομη περιγραφή του τρόπου υπολογισμού λύσεων για το πρόβλημα

Οι δύο μέθοδοι που χρησιμοποιούνται για την εύρεση λύσεων στο πρόβλημα των «N-Βασιλισσών» είναι η **solveQueens** και η **checkValidity**.

Μέθοδος solveQueens()

Η «solveQueens» είναι μία αναδρομική μέθοδος, η οποία δέχεται ως παραμέτρους:

1. **int col**: Ο αριθμός στήλης (0 ≤ col ≤ **N**) που καθορίζει τη στήλη στην οποία πρέπει να αναζητηθεί πιθανή τοποθέτηση βασίλισσας.

2. **Integer[] cols**: Ένας πίνακας **N** θέσεων που χρησιμοποιείται για την αποθήκευση των αποδεκτών θέσεων βασιλισσών που εντοπίζονται κατά την εκτέλεση της solveQueens(). Κάθε θέση του πίνακα αντιπροσωπεύει μία στήλη **c**, ενώ το περιεχόμενο κάθε θέσης είναι ένας αριθμός **r** (0 ≤ r ≤ N - 1) που αντιπροσωπεύει τη γραμμή στην οποία τοποθετείται μία βασίλισσα για τη συγκεκριμένη στήλη. Στην πρώτη κλήση της solveQueens(), δίνεται ως όρισμα ένας πίνακας **cols** ο οποίος περιέχει αρχικά μόνο μία τιμή **r** (0 ≤ r ≤ N - 1) στην πρώτη θέση του, η οποία αντιπροσωπεύει τη θέση της πρώτης βασίλισσας (δηλαδή, τη γραμμή τοποθέτησης βασίλισσας για την πρώτη στήλη).

Σε κάθε της κλήση, η μέθοδος ελέγχει τη δυνατότητα τοποθέτησης βασίλισσας σε κάθε σειρά της σκακιέρας για τη δεδομένη στήλη που ορίζει το όρισμα **col** που της έχει δοθεί. Ο παραπάνω έλεγχος εγκυρότητας τοποθέτησης γίνεται με κλήση της μεθόδου checkValidity(), στην οποία δίνονται ως

ορίσματα ο πίνακας **cols**, η τιμή στήλης **col** και η τιμή σειράς **row** για τα οποία θέλουμε να κάνουμε έλεγχο εγκυρότητας τοποθέτησης βασίλισσας. Σε περίπτωση που η τοποθέτηση βασίλισσας στη δεδομένη γραμμή και στήλη της σκακιέρας κριθεί ορθή, η checkValidity() επιστρέφει **True** και η solveQueens() εκχωρεί την τιμή σειράς **row** για την οποία εκτελέσαμε τον έλεγχο στη θέση **col** του πίνακα **cols**. Στη συνέχεια, πραγματοποιείται αναδρομική κλήση της μεθόδου solveQueens(), δίνοντας ως ορίσματα την τιμή της υπό εξέταση στήλης col αυξημένη κατά 1 (**col + 1**) καθώς και τον ανανεωμένο πίνακα λύσεων **cols**.

Η συνθήκη τερματισμού αναδρομής της μεθόδου είναι η εύρεση θέσης τοποθέτησης βασίλισσας για κάθε στήλη της σκακιέρας του δεδομένου σεναρίου. Η συνθήκη τερματισμού ικανοποιείται όταν γίνει αναδρομική κλήση της solveQueens() με τιμή **col** που ξεπερνάει το αποδεκτό όριο τιμών της (col == this.gridSize, δηλαδή col == N).

Μέθοδος checkValidity()

Όπως έχει ήδη αναφερθεί, σκοπός της checkValidity() είναι ο έλεγχος της εγκυρότητας τοποθέτησης βασίλισσας στη θέση που υποδεικνύουν τα ορίσματα **col1**, **row1** που λαμβάνει (στήλη και γραμμή αντίστοιχα), λαμβάνοντας υπόψιν την κατάσταση της σκακιέρας που αποτυπώνεται στον πίνακα λύσεων **cols** που ομοίως της δίνεται ως όρισμα.

Για να κριθεί δυνατή η τοποθέτηση βασίλισσας στη θέση που ορίζουν τα **row1** και **col1**, πρέπει να ικανοποιούνται οι εξής προϋποθέσεις:

1. Δεν πρέπει να υπάρχει ήδη βασίλισσα κατά μήκος της δοθείσας γραμμής **row1**.

2. Δεν πρέπει να υπάρχει ήδη βασίλισσα σε οποιαδήποτε από τις διαγώνιους που διέρχονται από τη θέση της σκακιέρας που ορίζουν τα **row1** και **col1**.

Διευκρινίζεται ότι δεν είναι αναγκαίος ο έλεγχος ύπαρξης βασίλισσας στη δοθείσα στήλη **col1**, μιας και η μέθοδος solveQueens() καλείται για κάθε στήλη ξεχωριστά, τοποθετώντας **1 βασίλισσα** κάθε φορά στη σκακιέρα. Συνεπώς, γνωρίζουμε ότι η τρέχουσα στήλη **col1** είναι άδεια.

Σε περίπτωση που οι 2 προαναφερθείσες συνθήκες ικανοποιούνται, η checkValidity() επιστρέφει **True**. Σε διαφορετική περίπτωση, επιστρέφεται **False**.

Μεταφορά της επίλυσης του προβλήματος στη λογική του παραλληλισμού

Η λογική που χρησιμοποιήθηκε για την υλοποίηση του αλγορίθμου επίλυσης του προβλήματος των «N-Βασιλισσών» με χρήση παράλληλων νημάτων (threads) πάρθηκε από την εκφώνηση του προβλήματος[1]. Σύμφωνα με αυτή, για την αξιοποίηση ταυτοχρονισμού στην επίλυση του προβλήματος *«αρκεί μία ξεχωριστή διεργασία να αναζητά κάποια λύση ξεκινώντας από την καθεμία από τις n εκχωρήσεις μιας βασίλισσας στην πρώτη στήλη».* Έτσι, γίνεται ανάθεση σε νήματα καθηκόντων (tasks) εύρεσης λύσης στο πρόβλημα «N-Βασιλισσών», με κάθε ένα από αυτά να έχει ως αρχικό δεδομένο την τοποθέτηση της πρώτης βασίλισσας σε μία από τις σειρές της πρώτης στήλης της σκακιέρας. Συνεπώς,

[1] "Principles of Concurrent and Distributed Programming", Mordechai Ben-Ari, σελίδα 378

όπως ορίζει η εκφώνηση, η εύρεση του συνόλου των αποδεκτών λύσεων διαχωρίζεται σε **N** διακριτά καθήκοντα (tasks) εύρεσης λύσης.

Ωστόσο, το πλήθος των «tasks» δεν είναι ανάγκη να ταυτίζεται απαραίτητα με το πλήθος ενεργών νημάτων. Για παράδειγμα, για N = 20 ορίζονται 20 διακριτά **«tasks»** εύρεσης λύσης στο πρόβλημα των «N-Βασιλισσών», όμως μπορεί να επιθυμούμε την κατανομή τους σε ένα πλήθος P = 4 νημάτων. Σε αυτή την περίπτωση, θέλουμε να κατανέμονται διαδοχικά «tasks» στα 4 διαθέσιμα νήματα, μέχρι την ολοκλήρωση της επεξεργασίας όλων των (N) καθηκόντων. Για να επιτευχθεί το παραπάνω, αξιοποιήθηκε η υπηρεσία εκτελεστή **Executors.newFixedThreadPool()** της «Java», με σκοπό την δημιουργία ενός «thread pool» μεγέθους P (που προσδιορίζεται από τον χρήστη), την τροφοδοσία του με N «tasks» και την αυτόματη κατανομή τους στα διαθέσιμα νήματα από αυτό. Δημιουργήθηκε επίσης μία κλάση **QueenTask** που υλοποιεί τη διεπαφή «Runnable». Η τροφοδοσία του εκτελεστή **taskExecutor** με «tasks» γίνεται με την υποβολή αντικειμένων της συγκεκριμένης κλάσης σε αυτόν.

Τέλος, εντός της κύριας κλάσης «Solver», έχει δημιουργηθεί μία λίστα **ArrayList<Integer[]> results**. Ο ρόλος της **results** είναι η αποθήκευση των λύσεων (των έγκυρων δηλαδή συνδυασμών στήλης-σειράς για την τοποθέτηση βασιλισσών στη σκακιέρα) που εντοπίζονται από τον αλγόριθμο. Με δεδομένο ότι στην παράλληλη υλοποίηση του αλγορίθμου είναι πιθανό διαφορετικά νήματα να αποπειραθούν ταυτόχρονη εγγραφή δεδομένων στη λίστα **results**, γίνεται χρήση «**synchronized block**» για το «**monitor**» της κλάσης **Solver** στο συγκεκριμένο τμήμα κώδικα της **solveQueens()** από το οποίο αποκτάται πρόσβαση εγγραφής στην **results**. Κατά αυτό τον τρόπο, διασφαλίζεται η αποκλειστική πρόσβαση μόνο ενός νήματος κάθε φορά στη λίστα **results**, για τον σκοπό της αποθήκευσης του αποτελέσματος του σε αυτή.

Μέθοδος main()

Εντός της μεθόδου main() γίνεται η εμφάνιση των αρχικών μηνυμάτων της εφαρμογής στον χρήστη και ο έλεγχος εγκυρότητας των εισόδων του. Οι είσοδοι που καλείται να δώσει ο χρήστης αφορούν το μέγεθος **N** της σκακιέρας του προβλήματος και το πλήθος νημάτων **P** που θα αξιοποιήσει ο αλγόριθμος για την επεξεργασία των **N** καθηκόντων εύρεσης λύσεων.

Οι είσοδοι του χρήστη ελέγχονται ως προς την εγκυρότητά τους σύμφωνα με τα παραπάνω κριτήρια:

1. Η είσοδος που αφορά το πλήθος νημάτων **P** της εφαρμογής πρέπει να είναι ακέραιος αριθμός μεγαλύτερος του 0.

2. Η είσοδος που αφορά το μέγεθος **N** πρέπει να είναι ακέραιος αριθμός μεγαλύτερος του 0, με **N ≠ 2 και N ≠ 3**.

Εφόσον οι είσοδοι του χρήστη ικανοποιούν τις παραπάνω συνθήκες, γίνεται αρχικοποίηση ενός αντικειμένου **solver** της κύριας κλάσης Solver με τα **P** και **N** ως ορίσματα. Ακολούθως, γίνεται κλήση της μεθόδου **startSolving()** του **solver** για την έναρξη της διαδικασίας εύρεσης λύσεων, ενώ στη συνέχεια καλείται η μέθοδος **printSolutionGrid()** του **solver** η οποία τυπώνει τα αποτελέσματα του αλγορίθμου, μόλις η **startSolving()** επιστρέψει στην «main» έχοντας επεξεργαστεί όλα τα «tasks».

Μέθοδος printSolutionGrid()

Η μέθοδος **printSolutionGrid()** εμφανίζει τις λύσεις που έχουν βρεθεί για το πρόβλημα των «N-Βασιλισσών». Οι λύσεις αποτυπώνονται πάνω σε μία σκακιέρα που σχηματίζεται με χρήση χαρακτήρων, όπου κάθε σύμβολο «**.**» αντιπροσωπεύει κενή θέση της σκακιέρας, ενώ κάθε σύμβολο «**X**» αντιπροσωπεύει θέση στην οποία τοποθετείται βασίλισσα.