
Matematikos ir informatikos fakultetas

**VILNIAUS UNIVERSITETAS**

**MATEMATIKOS IR INFORMATIKOS FAKULTETAS**

MANTAS DONĖLA

3 kursas, informatika

**Ataskaita**

# N Valdovių išdėstymo NxN lentoje uždavinys.

**Darbo vadovas:** Rimantas Vaicekauskas

# 1. Problemos Formulavimas

N karalienių problema yra problema, kurios tikslas yra išdėstyti N šachmatų karalienes ant N×N dydžio šachmatų lentos taip, kad nei viena karalienė negalėtų pulti kitos. Tai reiškia, kad nei vienoje eilutėje, stulpelyje ar įstrižainėje negali būti daugiau vienos karalienės. Problema yra žinoma dėl savo sudėtingumo ir didelio sprendimų skaičiaus, ypač didinant lentos dydį. Šio uždavinio sprendimas naudojant lygiagretųjį algoritmą leidžia geriau suprasti ir išnaudoti kompiuterių skaičiavimo galimybe, bei spręsti kompleksines problemas efektyviau, ypač turint omenyje modernių daugiabranduolių procesorių architektūrą.

# 2. Lygiagretusis algoritmas

**Darbo paskirstymas**: Kiekviena gija pradeda darbą su iš anksto nustatyta karalienės pozicija pirmoje eilutėje, tokiu būdu priskiriant pradinį sprendimo lygį. Toliau kiekviena gija, nepriklausomai nuo kitų, bando išdėstyti likusias karalienes, laikydamasi taisyklės, kad jos negali pulti viena kitos. Ši užduotis apima rekursinį metodą, kurio metu gija, nustačiusi saugią karalienės poziciją, kviečia save su kitu eilės numeriu. Jei pasiekiama lentos pabaiga ir visos karalienės saugiai išdėstytos, sprendimas pridedamas prie bendro sprendimų sąrašo.

**Gijų bendravimas:** gijos „bendrauja“ per bendrą saugomą struktūrą – sinchronizuotą sąrašą, kuris leidžia saugoti visus rastus sprendimus be duomenų vientisumo pažeidimo. Sinchronizacijos mechanizmai užtikrina, kad tuo pačiu metu duomenis modifikuojantys procesai nesukeltų duomenų konfliktų ar klaidų.

# 3. Vykdymo aplinka

Šiam eksperimentui buvo pasinaudota Vu MIF clusteriai “Nuotoliniai kompiuteriai”. Buvo prisijungta prie jų iš asmeninio kompiuterio ir paleisti su skirtingu skaičiu gijų.

Testavimo programa buvo pasirinkta IntelliJ IDEA aplinka, kurioje yra naudojama JAVA programavimo kalba.

# 4. Eksperimento tyrimo rezultatai

Eksperimentinio tyrimo rezultatai parodė, kad N-Queens Problem Solver algoritmo veiksmingumas priklauso tiek nuo lentos dydžio (N), tiek nuo naudojamų gijų skaičiaus. Tyrimas atliktas su įvairiais lentos dydžiais nuo 2 iki 14, o panaudotų gijų skaičius buvo – 1, 2, 4, 8, 16.

Didėjant lentos dydžiui, vykdymo laikas didėjo, tačiau lygiagretinimas leido žymiai sumažinti šį laiką. Pavyzdžiui, su lenta dydžio 14, vienos gijos sprendimo laikas buvo 5661 ms, o naudojant 16 gijų, laikas sumažėjo iki 832 ms, rodydamas akivaizdų spartinimą. Įdomu pastebėti, kad kai kuriais atvejais, pavyzdžiui, lenta dydžio 8 ir 9, naudojant didesnį gijų skaičių, sprendimo laikas nežymiai padidėjo, galimai dėl sinchronizacijos sąnaudų.

Board size: 6

Threads: 1 | Time taken: 2 ms | Solutions found: 4

Threads: 2 | Time taken: 1 ms | Solutions found: 4

Threads: 4 | Time taken: 1 ms | Solutions found: 4

Threads: 8 | Time taken: 1 ms | Solutions found: 4

Threads: 16 | Time taken: 1 ms | Solutions found: 4

Board size: 7

Threads: 1 | Time taken: 2 ms | Solutions found: 40

Threads: 2 | Time taken: 1 ms | Solutions found: 40

Threads: 4 | Time taken: 1 ms | Solutions found: 40

Threads: 8 | Time taken: 1 ms | Solutions found: 40

Threads: 16 | Time taken: 1 ms | Solutions found: 40

Board size: 8

Threads: 1 | Time taken: 4 ms | Solutions found: 92

Threads: 2 | Time taken: 2 ms | Solutions found: 92

Threads: 4 | Time taken: 3 ms | Solutions found: 92

Threads: 8 | Time taken: 1 ms | Solutions found: 92

Threads: 16 | Time taken: 1 ms | Solutions found: 92

Board size: 9

Threads: 1 | Time taken: 5 ms | Solutions found: 352

Threads: 2 | Time taken: 3 ms | Solutions found: 352

Threads: 4 | Time taken: 2 ms | Solutions found: 352

Threads: 8 | Time taken: 2 ms | Solutions found: 352

Threads: 16 | Time taken: 2 ms | Solutions found: 352

Board size: 10

Threads: 1 | Time taken: 9 ms | Solutions found: 724

Threads: 2 | Time taken: 9 ms | Solutions found: 724

Threads: 4 | Time taken: 4 ms | Solutions found: 724

Threads: 8 | Time taken: 4 ms | Solutions found: 724

Threads: 16 | Time taken: 4 ms | Solutions found: 724

Board size: 11

Threads: 1 | Time taken: 33 ms | Solutions found: 2680

Threads: 2 | Time taken: 29 ms | Solutions found: 2680

Threads: 4 | Time taken: 20 ms | Solutions found: 2680

Threads: 8 | Time taken: 10 ms | Solutions found: 2680

Threads: 16 | Time taken: 8 ms | Solutions found: 2680

Board size: 12

Threads: 1 | Time taken: 171 ms | Solutions found: 14200

Threads: 2 | Time taken: 98 ms | Solutions found: 14200

Threads: 4 | Time taken: 60 ms | Solutions found: 14200

Threads: 8 | Time taken: 36 ms | Solutions found: 14200

Threads: 16 | Time taken: 37 ms | Solutions found: 14200

Board size: 13

Threads: 1 | Time taken: 927 ms | Solutions found: 73712

Threads: 2 | Time taken: 487 ms | Solutions found: 73712

Threads: 4 | Time taken: 286 ms | Solutions found: 73712

Threads: 8 | Time taken: 154 ms | Solutions found: 73712

Threads: 16 | Time taken: 183 ms | Solutions found: 73712

Board size: 14

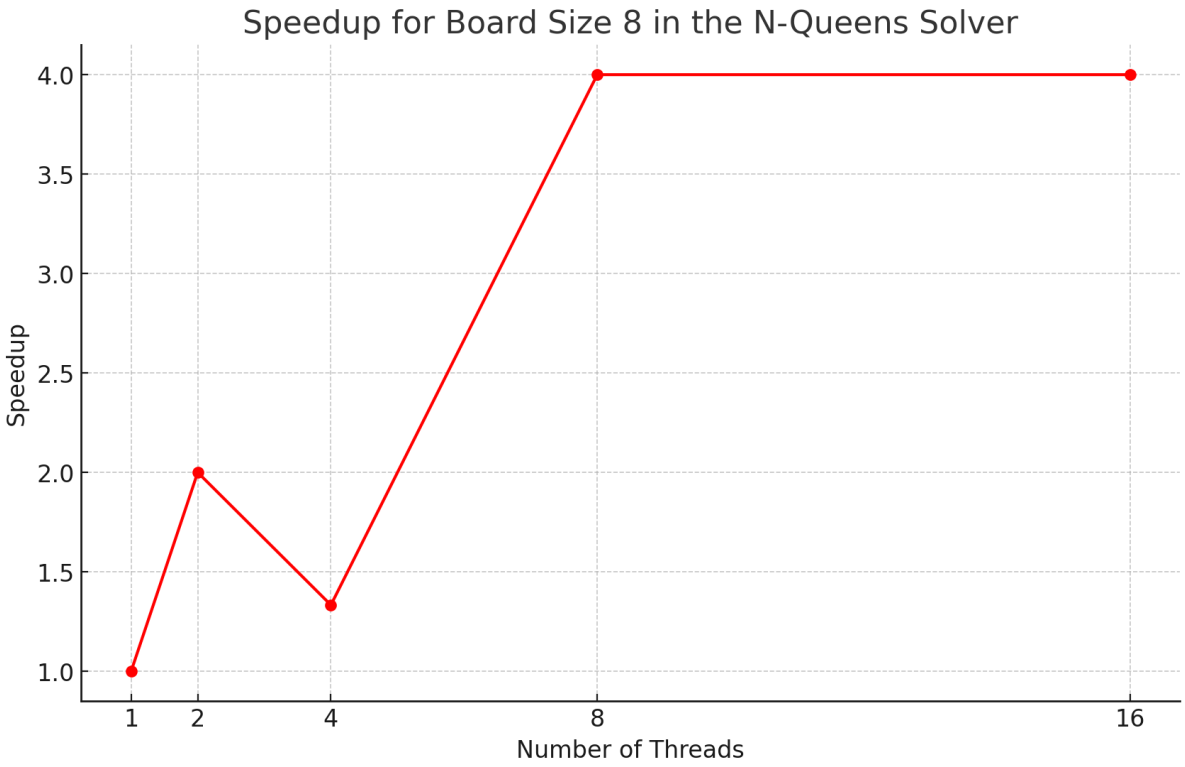
Threads: 1 | Time taken: 5661 ms | Solutions found: 365596

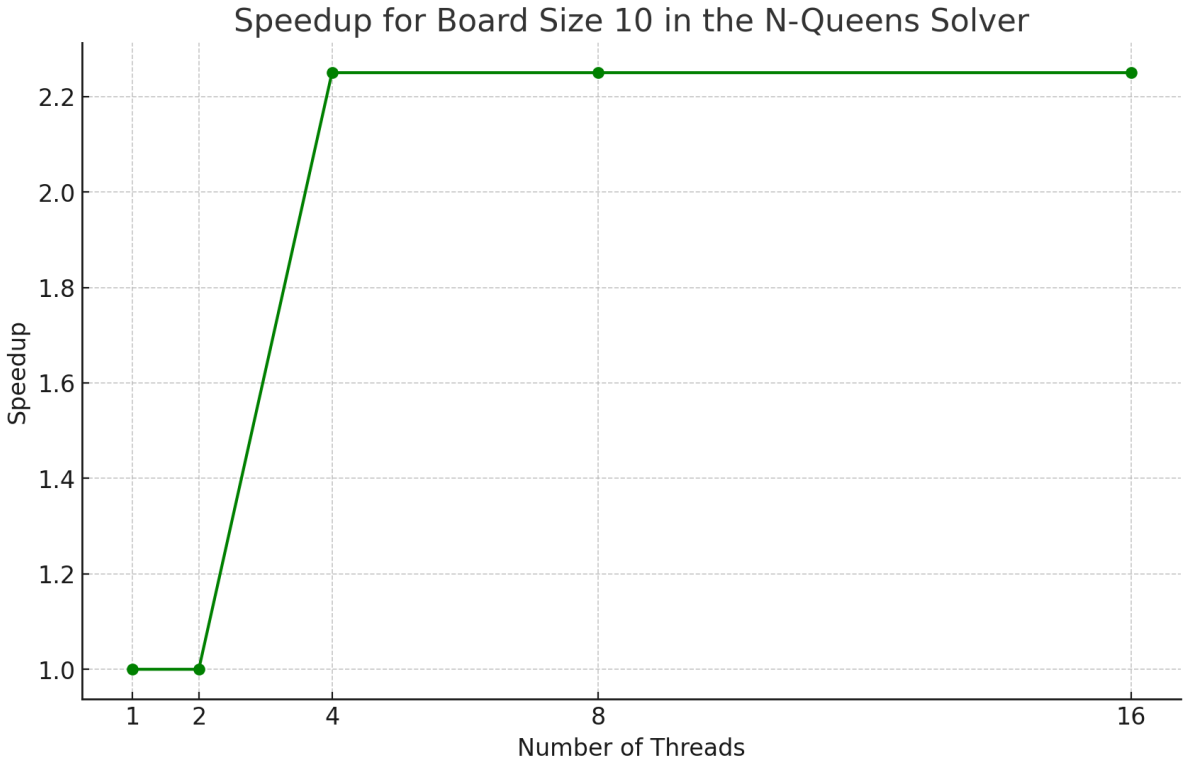
Threads: 2 | Time taken: 2814 ms | Solutions found: 365596

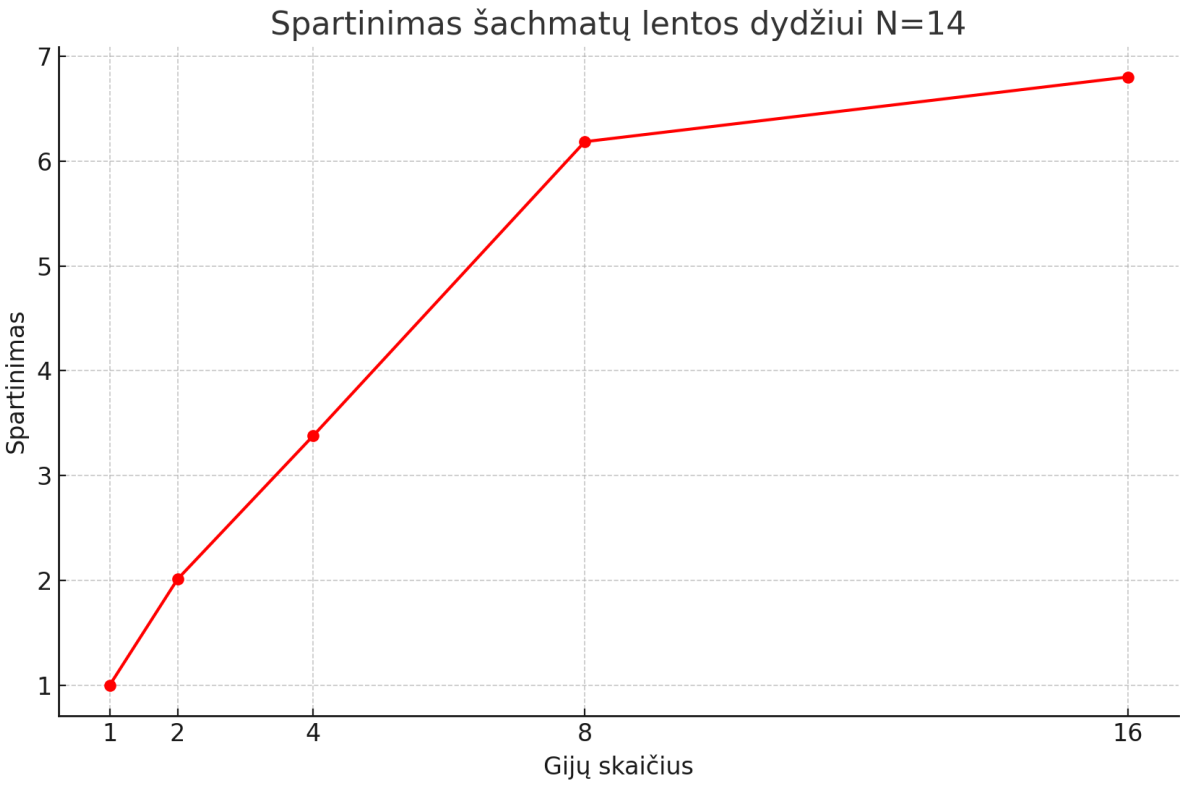
Threads: 4 | Time taken: 1674 ms | Solutions found: 365596

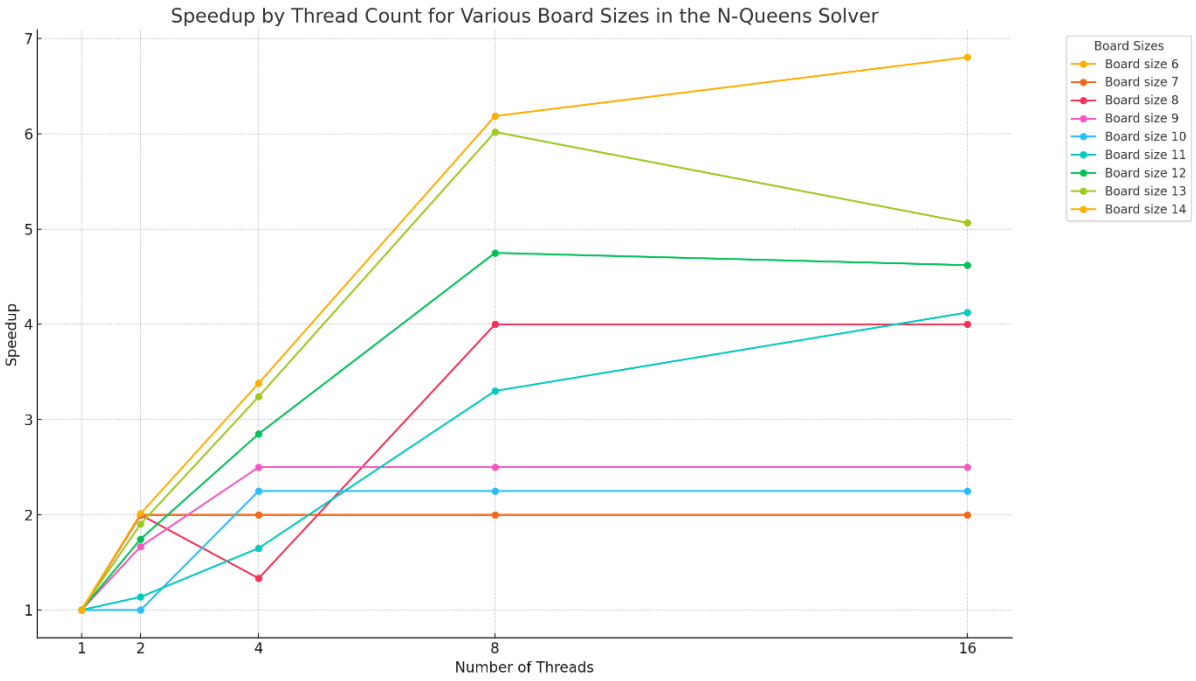
Threads: 8 | Time taken: 915 ms | Solutions found: 365596

Threads: 16 | Time taken: 832 ms | Solutions found: 365596


Speedup for Board Size 8 in the N-Queens Solver
Speedup
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
2
4
8
16
Number of Threads


Speedup for Board Size 10 in the N-Queens Solver
Speedup
2.2
2.0
1.8
1.6
1.4
1.2
1.0
1
2
4
8
16
Number of Threads


Spartinimas šachmatų lentos dydžiui N=14
Spartinimas
Gijų skaičius
1
2
3
4
5
6
7
1
2
4
8
16


Speedup by Thread Count for Various Board Sizes in the N-Queens Solver
Speedup
Number of Threads
Board Sizes
Board size 6
Board size 7
Board size 8
Board size 9
Board size 10
Board size 11
Board size 12
Board size 13
Board size 14
1
2
3
4
5
6
7
1
2
4
8
16

# 5. Išvados

Remiantis atliktais eksperimentais su įvairiais lentos dydžiais ir skirtingu gijų skaičiumi, N-Queens Problem Solver algoritmo analizė rodo, kad lygiagretusis programavimas yra efektyvus būdas spręsti šią problemą, ypač kai uždavinį skaičiuojame su didesniais N. Algoritmo spartinimas matomas aiškiai, didinant gijų skaičių, kas leidžia ženkliai sumažinti vykdymo laiką.

**Spartinimas**: Lygiagretus vykdymas su keliomis gijomis parodė ženklų spartinimą, palyginti su viena gija. Pavyzdžiui, su N=14, vykdymo laikas su 16 gijomis buvo maždaug 7 kartus mažesnis nei su viena gija. Tai rodo, kad lygiagretus vykdymas leidžia efektyviai panaudoti daugiabranduolių procesorių galimybes.

**Optimalus gijų skaičius:** Nors didinant gijų skaičių sumažėjo vykdymo laikas, eksperimentai rodo, kad ne visada daugiau gijų reiškia greitesnį sprendimą. Ypač mažesniems N, didelis gijų skaičius nežymiai veikė ar net pablogino vykdymo laiką, galimai dėl didesnių sinchronizacijos ir gijų valdymo sąnaudų. Tai parodo, kad svarbu rasti optimalų gijų skaičių atsižvelgiant į konkrečios problemos dydį.

**Plečiamumas:** Algoritmas rodo gerą plečiamumą didinant lentos dydį, kas yra svarbu dideliems duomenų kiekiams ir sudėtingesniems skaičiavimams. Lygiagretusis algoritmas leidžia panaudoti skaičiavimo resursus proporcingai didinant uždavinio sudėtingumą, todėl yra ypač tinka sistemose, kur reikalingas efektyvus didelių problemų sprendimas. Tačiau idealiai plečiamumas nebuvo pasiektas nes jei yra daugiau gijų nei N tai gali gijos laukti darbo ir nedirbti.

# 6. Programos kodas

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
import java.util.Collections;
import java.util.List;
import java.util.ArrayList;

public class NQueensSolver {
    private int boardSize;
    private List<int[]> solutions;

    public NQueensSolver(int boardSize) {
        this.boardSize = boardSize;
        this.solutions = Collections.synchronizedList(new
ArrayList<>());
    }

    public long solve(int numberOfThreads) {
        long startTime = System.currentTimeMillis();
        ExecutorService executor =
Executors.newFixedThreadPool(numberOfThreads);
        for (int i = 0; i < boardSize; i++) {
            int[] initial = new int[boardSize];
            initial[0] = i;
            executor.execute(new WorkerThread(initial, 1));
        }
        executor.shutdown();
        try {
            executor.awaitTermination(1, TimeUnit.HOURS);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        long endTime = System.currentTimeMillis();
        return endTime - startTime;
    }

    private class WorkerThread implements Runnable {
        private int[] state;
        private int row;

        WorkerThread(int[] state, int row) {
            this.state = state.clone();
            this.row = row;
        }

        @Override
        public void run() {
            if (row == boardSize) {
                synchronized (solutions) {
                    solutions.add(state.clone());
                }
            } else {
                for (int i = 0; i < boardSize; i++) {
```

```
                    if (isValid(state, row, i)) {
                        state[row] = i;
                        new WorkerThread(state, row + 1).run();
                    }
                }
            }
        }

        private boolean isValid(int[] state, int row, int col) {
            for (int i = 0; i < row; i++) {
                if (state[i] == col || Math.abs(state[i] - col) == row -
i) {
                    return false;
                }
            }
            return true;
        }
    }

    public List<int[]> getSolutions() {
        return solutions;
    }

    public static void main(String[] args) {
        int n[] = {4, 8, 10}; // Default board size
        int[] threadCounts = {1, 2, 4, 8, 16}; // Different numbers of
threads to test



        System.out.println("N-Queens Problem Solver");

        for(int i : n){
            System.out.println("Board size: " + i);
            for (int threads : threadCounts) {
                NQueensSolver solver = new NQueensSolver(i);
                long timeTaken = solver.solve(threads);
                System.out.println("Threads: " + threads + " | Time
taken: " + timeTaken + " ms | Solutions found: " +
solver.getSolutions().size());
            }

        }

    }
}
```