Grupo: Freddy Da Paz Ilha, Maurício Dorneles Caldeira Balboni

O documento a seguir irá apresentar o relatório de um trabalho da disciplina de Conceitos de Linguagens de Programação da UFPEL, o trabalho tem como objetivo resolver o problema das N-Rainhas o qual consiste em posicionar n rainhas em um tabuleiro nxn tal que nenhuma se cruze nem em linha, nem coluna e nem nas diagonais.
Nesse Relatório será apresentado o código comentado, e gráficos de tempo de execução das implementações conforme o número de threads usadas.

O Código mostrado a seguir foi o utilizado para a realização dos testes:

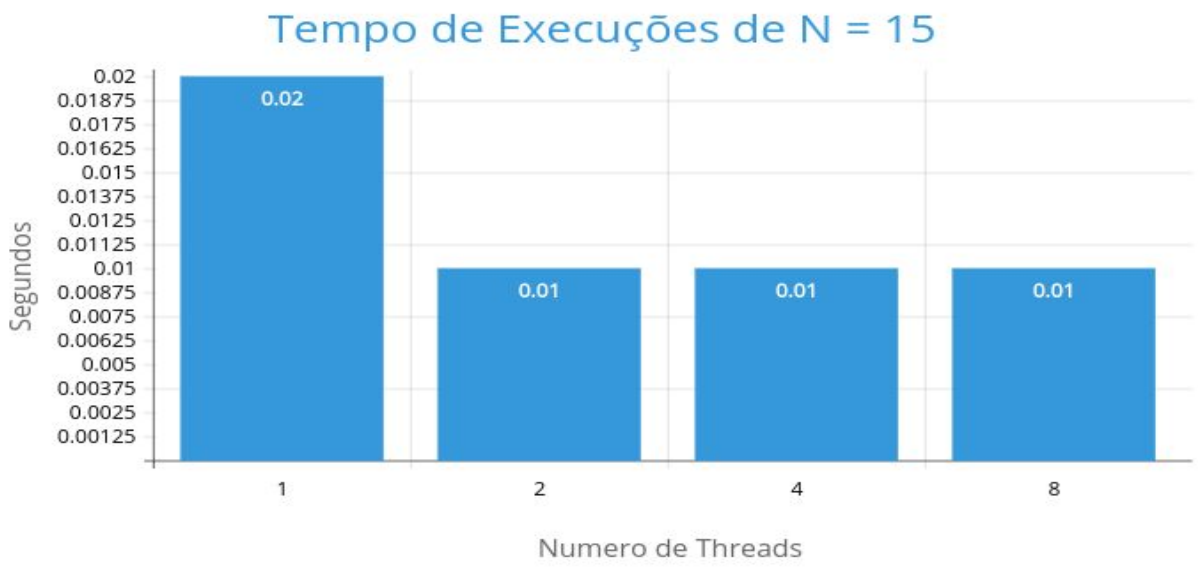
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

#define QUEEN  7
#define EMPTY  0

int menor = 999;
int main(int argc, char* argv[])
{
	omp_lock_t writelock;
	int
nthreads,tid,boardSize,i,j,k,g,w,linhaAcima,wrongMove,aux,aux1=0;
	if (argc == 1)
	{/*Se nenhum valor foi passado o valor default é 4*/
		boardSize = 4;
	}
	else
	{/*Se o valor passado for menor que 4 não existe solução para o
problema*/
		if (atoi(argv[1]) < 4)
		{
			printf("Não pode ser um numero menor que 4\n");
			return 0;
		}
		boardSize = atoi(argv[1]); //Tamanho recebe o valor passado
por parametro
	}

	wrongMove = 0; // Essa variavel é utilizada pra ver, se é possivel
ter uma rainha ali ou não, 0 para sim, 1 para não.
```

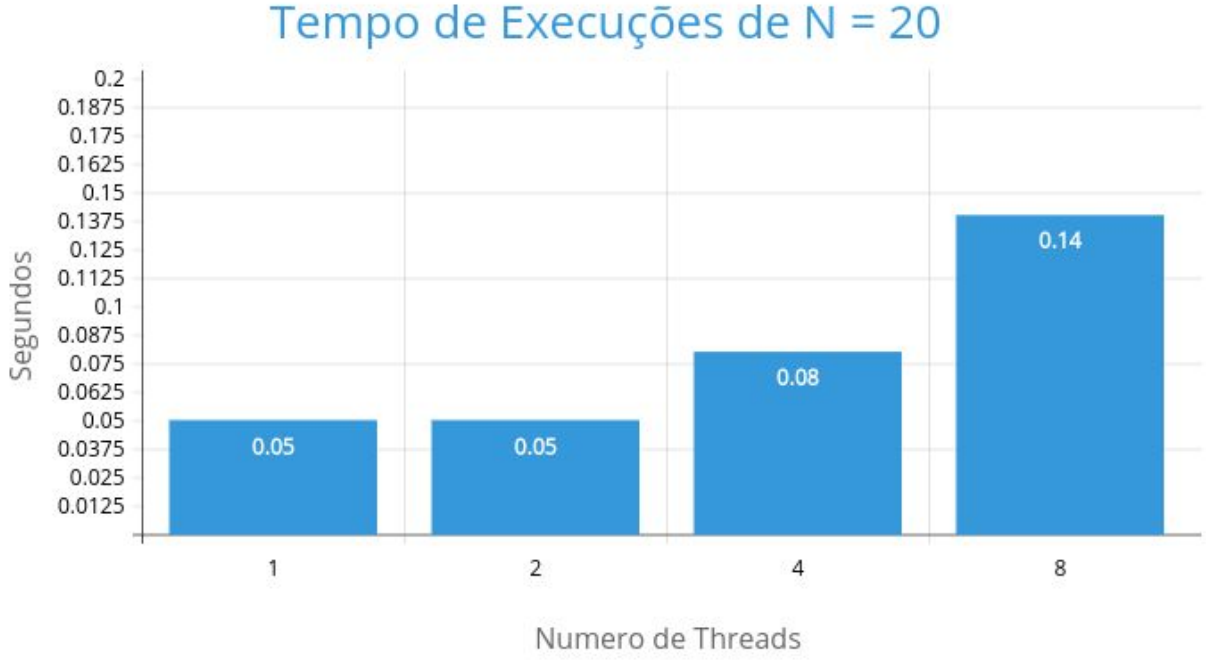
```
      aux = 0; // Essa variavel é utilizada depois para verificar as
rainhas da linha anterior
      int board [boardSize][boardSize];

      for (i = 0; i < boardSize; i++)
      {
            for (j = 0; j < boardSize; j++)
            {
                  board[i][j] = EMPTY;  // Preenche a matriz do tabuleiro
            }
      }
      j=0;
      omp_init_lock(&writelock); // A init no Lock
      {
            for (i = 0; i < boardSize; i++)
            {/*For percorrendo as linhas*/
                  #pragma omp parallel for
firstprivate(wrongMove,k,linhaAcima,i,nthreads, tid)
                  for (j=aux1; j < boardSize; j++)
                  {/*As variaveis 'wrongMove', 'k', 'linhaAcima' e 'i' são
firstprivate para cada thread ter a sua copia e não interferir com a
outra*/
                  /*For paralelo para cada thread pegar uma posição e
testar se a rainhas colidindo acima*/
                        linhaAcima = i-1; //Verifica a linha anterior da
matriz
                        k = 1;
                        wrongMove = 0;

                        while(linhaAcima != -1) /*Teste para conferir se há
outra rainha acima ou nas diagonais*/
                        {
                              if (j-k < 0)
                              {
                                    if(board[linhaAcima][j] == QUEEN ||
board[linhaAcima][j+k] == QUEEN)
                                    {
                                          wrongMove = 1;
                                    }
                              }
                              else if (j+k > boardSize-1)
                              {
                                    if(board[linhaAcima][j] == QUEEN ||
board[linhaAcima][j-k] == QUEEN)
                                    {
```

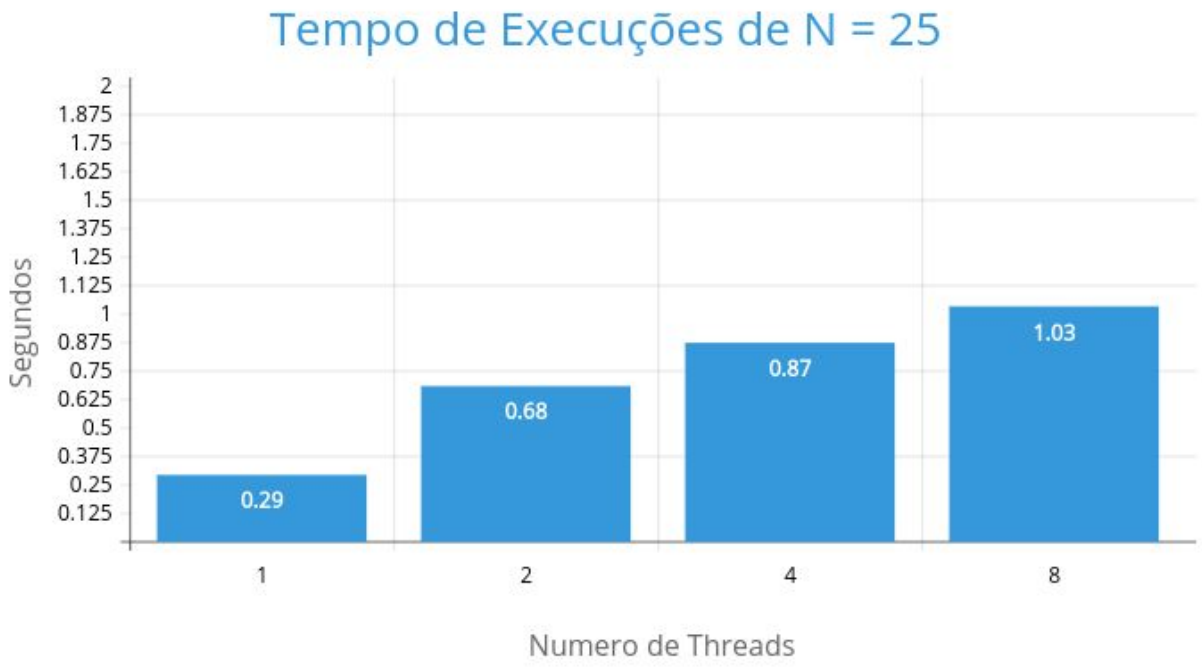
```
                                    wrongMove = 1;
                              }
                        }
                        else if (j+k > boardSize-1 || j-k < 0)
                        {
                              if(board[linhaAcima][j] == QUEEN ||
board[linhaAcima][j-k] == QUEEN)
                              {
                                    wrongMove = 1;
                              }
                        }
                        else
                        {
                              if(board[linhaAcima][j] == QUEEN ||
board[linhaAcima][j-k] == QUEEN || board[linhaAcima][j+k] == QUEEN)
                              {
                                    wrongMove = 1;
                              }
                        }

                        k++;
                        linhaAcima--;
                        if(wrongMove == 1)
                              linhaAcima = -1;
                  }
                  omp_set_lock(&writelock); // LOck para poder
escrever sem dar condição de corrida
                  if (wrongMove == 0)
                  {
                        if (j < menor)
                        {
                              menor = j; // Coloca a menor posição da
linha para se colocar uma rainha
                        }
                        else{

                        }
                  }
                  omp_unset_lock(&writelock);       //FIm do lock

            }

            #pragma omp master
            {
                  if (menor < boardSize) //Verifica se tem alguma
```

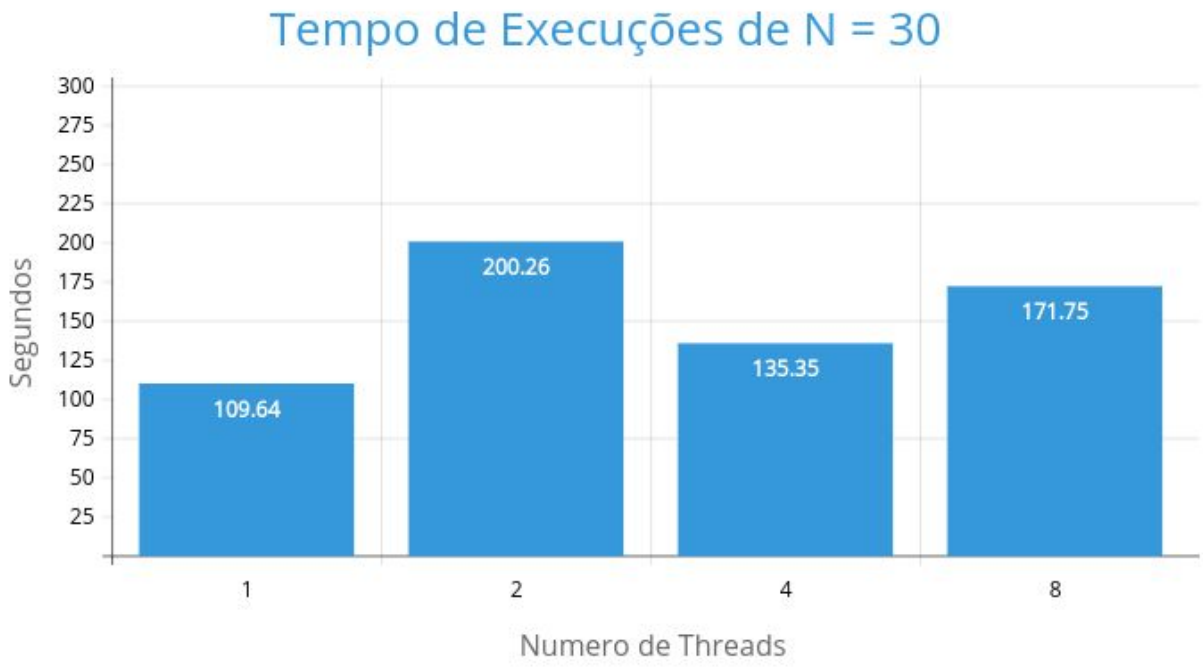
```
posição valida para por a rainha
                        {
                              board[i][menor] = QUEEN; // Coloca a Rainha
na menor posição valida
                              menor = 999; // seta o valor menor garnde,
para fazer a proxima iteração
                              aux1=0; // Novo valor do J
                        }
                        else
                        { // SE entrou aqui é pq nao tem nenhuma posição
valida para por a rainha
                              aux = i-1; // VAi para a linha de cima
                              for (g = 0; g < boardSize; g++) // Procura a
Posição que tem uma rainha
                              {
                                    if (board[aux][g] == QUEEN) // Se achou
a rainha, entao tira ela e seta como vazio
                                    {
                                          board[aux][g] = EMPTY;
                                          aux1 = g+1;  /* Posição que o j
deve continuar pra pegar a proxima rainha*/
                                          i -= 2; // Vai para a linha
anterior.
                                          g = boardSize+1;
                                    }
                              }
                        }
                  }
            }
      }
      /*Resultado*/
      // for (i = 0; i < boardSize; i++)
      // {
      //    for (j = 0; j < boardSize; j++)
      //    {
      //          printf(" %d ", board[i][j]);
      //    }
      //    printf("\n");
      // }
}
```

A seguir será apresentado quatro gráficos com a média de tempo de execução dos resultados de cada thread pelo número de entradas(N) de cada tabuleiro(NxN), foram obtidos 10 amostras para cada teste.


Tempo de Execuções de N = 15
Segundos
0.02
0.01875
0.0175
0.01625
0.015
0.01375
0.0125
0.01125
0.01
0.00875
0.0075
0.00625
0.005
0.00375
0.0025
0.00125
0.02
0.01
0.01
0.01
1
2
4
8
Numero de Threads


Tempo de Execuções de N = 20
Segundos
0.2
0.1875
0.175
0.1625
0.15
0.1375
0.125
0.1125
0.1
0.0875
0.075
0.0625
0.05
0.0375
0.025
0.0125
0.05
0.05
0.08
0.14
1
2
4
8
Numero de Threads





Comando usados para a realização dos testes
export OMP_NUM_THREADS='NThreds'
gcc -pg -o 'teste' -O0 -fopenmp 'nometrabalho.c'
./'teste' 'N'

gprof -b ‘teste’ gmon.out