# Algoritmos de Busca para o Problema da 8 Rainhas: Estudo Comparativo

Bianca Carvalho de Oliveira[1], Ana Paula Bernardes da Silva[1]

[1] Acadêmico do curso de Ciência da Computação – Universidade Estatual do Oeste do Paraná (UNIOESTE) – Cascavel, PR - Brasil

## 1. Introdução

Este estudo objetiva-se em avaliar, por meio de experimentação, os dados gerados com a utilização de algoritmos para a resolução do problema das 8 rainhas através das estratégias de busca por profundidade limitada e busca subindo o morro. Para tanto, são executados testes com 50 entradas para cada algoritmo e, os dados coletados são comparados.

Ao iniciar com um arquivo com um tabuleiro com uma rainha em uma posição aleatória (estado inicial), são avaliados os tempos de execução, bem como se o estado desejado é alcançado. Com os resultados espera-se apontar as vantagens e desvantagens ao se fazer uso de uma estratégia em detrimento de outra.

O presente estudo está organizado da seguinte forma. A Seção 2 apresenta a descrição do problema. A seção 3 apresenta a descrição da implementação. Seção 4 descreve a análise dos resultados. Por fim, a seção 5 conclui o trabalho.

## 2. Descrição do Problema

O problema das 8 Rainhas é um jogo de tabuleiro para ser jogado individualmente, sem adversários. O problema popularizou-se em 1848 quando o jornal alemão Schachzeitung o publicou, sendo proposto pelo enxadrista Max Bezzel. Ele questionou de quantas maneiras oito rainhas rivais – capazes de mover qualquer número de casas na horizontal, vertical e diagonal (Figura 1) – poderiam ser posicionadas em um tabuleiro padrão de 64 quadrados sem que nenhuma rainha atacasse outra.

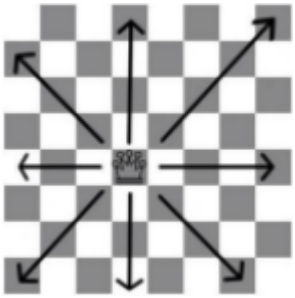


*Figura 1: Direção de ataque da rainha.*

Mesmo com vários matemáticos tentando resolver, a primeira solução foi proposta por Franz Nauck em 1850. A solução foi que havia 92 configurações que mantinham as oito rainhas afastadas umas das outras, com todas menos 12 das soluções sendo simples rotações e reflexões umas das outras. (TURNER, 2022).

O tabuleiro pode ter variação de tamanho de 4x4 até nxn, e o número de rainhas é

correspondente ao tamanho do tabuleiro. Neste estudo usaremos um tabuleiro de tamanho fixo 8x8 e 8 rainhas.

## 3. Descrição da implementação

Visando analisar o desempenho de ambas as abordagens, bem apontar suas vantagens e desvantagens, foi implementada uma solução utilizando a linguagem JAVA. A solução de um problema está em seu espaço de busca, formado por todas as suas possíveis soluções, este espaço de busca pode ser estruturado em forma de árvores ou grafos. Para encontrar soluções para esse desafio, são implementados algoritmos de busca, neste trabalho foi utilizado um algoritmo de busca cega: Profundidade Limitada, e um de busca heurística: subindo o Morro (ou Hill Climbing ou a Colina ou a Encosta).

Vale ressaltar que o problema foi implementado de modo que quando uma rainha é colocada no tabuleiro, todas as casas na qual ela pode atacar, são preenchidas, evitando que os métodos coloquem alguma rainha nessas posições.

### 3.1. Profundidade Limitada

O algoritmo de profundidade segue sua rota até sua máxima profundidade para depois seguir para a próxima rota. Emprega um procedimento chamado retrocesso cronológico, ele volta na árvore de busca quando um caminho sem saída é encontrado. No nosso caso, aplicamos o método de Profundidade Limitada, ele funciona da mesma maneira da profundidade citada acima, a única diferença é que temos um limite de profundidade no algoritmo, isso evita os caminhos infinitos (POSTAL, 2022).

Para análise desta estratégia utilizou-se o pseudocódigo abaixo (Pseudocódigo 1):

```
v <- raiz
BuscaProfundidade(l)
        marcar v;
        colocar v na pilha Q;
        enquanto(verdadeiro)
            se Q == []
                retorne falha;
            se eh_objetico(v)
                retorne Caminho(v);
            senão
                se profundidade de v < l
                    se v possui sucessores w
                        se existe w não marcado
                                buscaSucessor(v); //busca sucessor mais à esquerda de v que não foi marcado
                                coloca x na pilha;
                        senão
                                retira v da pilha;

                    senão
                        retira v da pilha;
                senão
                    retira v da pilha;
FIM
```

*Pseudocódigo 1: Busca por Profundidade Limitada.*

### 3.2. Subindo o morro

O algoritmo subindo o morro foi inspirado no método de profundidade, porém ele amplia o nó atual da busca e analisa os seus filhos, o melhor filho é escolhido para a próxima análise. O método não guarda o histórico do processo de subida, então ele não consegue se recuperar de possíveis falhas de sua técnica.

Ele possui duas variantes:

- Subindo o morro simples: examina os sucessores do estado atual e segue para o primeiro estado que for melhor (mais próximo da solução) que o atual.
- Subindo o morro pela trilha mais íngreme: examina todos os sucessores do estado atual e escolhe entre estes sucessores qual é o que está mais próximo da solução, para isso utiliza-se uma heurística.

Para este estudo utilizou-se a segunda variante deste algoritmo e a heurística escolhida foi: escolher o sucessor cujo tabuleiro possui mais casas vazias – e que não possua conflito com outra rainha – na qual a próxima rainha possa ser inserida.

Para análise desta estratégia utilizou-se o pseudocódigo abaixo (Pseudocódigo 2):

```
fila <-- [] // inicializa uma fila vazia
estado <-- no_raiz // inicializa o estado inicial
enquanto (verdadeiro)
        se eh_objetivo(estado)
                retorne SUCESSO
        senão
                ordenar(sucessores(estado))
                inserir_na_frente_da_fila(sucessores(estado))
         se fila == []
                retorne FALHA
        estado <-- fila[0] // o novo estado é o primeiro item da fila
        remover_primeiro_item_da(fila)
fim_enquanto
```

*Pseudocódigo 2: Busca Subindo o Morro.*

## 4. Resultados e Discussões

A Linguagem JAVA foi definida como meio de codificação para a realização das simulações e a *IntelliJ IDEA* como ambiente de desenvolvimento. Quanto ao *hardware*, a máquina em que os testes foram executados tem as seguintes configurações:

- Processador: Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz  1.80 GHz
- Memória RAM: 6,00 GB (utilizável: 5,88 GB)
- Sistema Operacional: Windows 10 Home

Foram realizados testes para analisarmos o tempo de execução e a completude dos algoritmos. Cada algoritmo foi executado 50 vezes, com a primeira rainha colocada de forma

aleatória em uma casa do tabuleiro. Nas tabelas 1 e 2 apresentadas, optamos por apresentar os primeiros 10 testes (Anexo I) para melhor visualização.

Nesta seção faremos a comparação entre os dois algoritmos.

### 4.1. Profundidade Limitada

A Tabela 1 mostra os valores médios do desempenho do algoritmo de busca por profundidade limitada em relação aos tempos de execução.

| Teste | Tempo Execução (ns) | Colocou 8 rainhas? (0-não 1-sim) |
|---|---|---|
| **1** | 284718800 | 1 |
| **2** | 14391300 | 1 |
| **3** | 9665200 | 1 |
| **4** | 75561800 | 1 |
| **5** | 1244100 | 1 |
| **6** | 369697200 | 1 |
| **7** | 15073600 | 1 |
| **8** | 12424800 | 1 |
| **9** | 11827500 | 1 |
| **10** | 268991800 | 1 |
| **Média** | **106359610** | - |

*Tabela 1: Valores do desempenho do algoritmo em relação aos tempos de execução.*

Com base na Tabela 1 observa-se que a média do tempo de execução do algoritmo foi de 106359610ns e em todas as execuções foram colocadas as 8 rainhas no tabuleiro, ou seja, ele consegue atingir a solução ótima. Isso deve-se ao fato que o limite de profundidade determinado foi igual a 7, caso ele seja menor que 7, o algoritmo não alcançara uma solução (Figura 2), pois a árvore de busca deve ter no mínimo 7 níveis (o primeiro nível da árvore é 0) para colocar as 8 rainhas no tabuleiro.

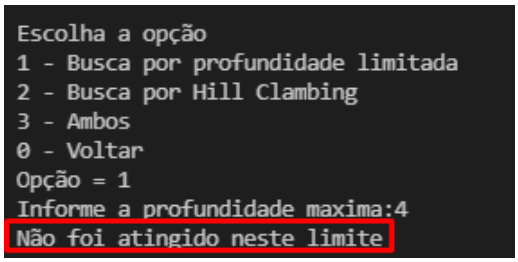



*Figura 2: Execução do algoritmo de busca por profundidade limitada com o limite menor que 7.*

### 4.2. Subindo do morro

A Tabela 2 mostra os valores médios do desempenho do algoritmo de busca subindo o morro em relação aos tempos de execução.

| Teste | Tempo Execução (ns) | Colocou 8 rainhas? (0-não 1-sim) |
|---|---|---|
| **1** | 453500 | 0 |
| **2** | 554000 | 0 |
| **3** | 330900 | 1 |
| **4** | 752700 | 1 |
| **5** | 646200 | 1 |
| **6** | 918900 | 0 |
| **7** | 624500 | 0 |
| **8** | 627300 | 0 |
| **9** | 717400 | 1 |
| **10** | 1059400 | 0 |
| **Média** | **668480** | **-** |

*Tabela 2: Valores do desempenho do algoritmo em relação aos tempos de execução.*

Observa-se que a média do tempo de execução do algoritmo foi de 668480 ns e em apenas 4 execuções foram colocadas as 8 rainhas no tabuleiro, isto é, o método só encontrou a solução em 40% dos casos.

### 4.3. Profundidade Limitada VS. Subida de Encosta

Observando as tabelas acima é possível levantar as seguintes vantagens e desvantagens das estratégias analisadas.

| | **Profundidade Limitada** | **Subida de Encosta** |
|---|---|---|
| **Tempo execução** | Maior | Menor |
| **Solução Ótima** | Maioria das vezes | Minoria das vezes |

*Tabela 3: Comparação dos algoritmos de busca por profundidade limitada Vs busca subindo o morro.*

## 5. Conclusão

Considerando esses resultados, e possível concluir que o problema proposto é mais bem solucionado utilizando a estratégia de busca por profundidade limitada, já que ela consegue alcançar o estado desejável todas as vezes, ao contrário da estratégia utilizando busca subindo o morro.

Conclui-se também que a estratégia utilizando busca subindo o morro, apesar de possuir um tempo execução mais baixo, não consegue atingir o estado desejado na maioria das vezes.

### Referências

POSTAL, Adriana. INTELIGÊNCIA ARTIFICIAL. Cascavel: Adriana Posta, 2022. 162 slides, color.

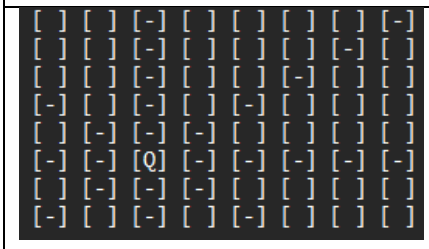
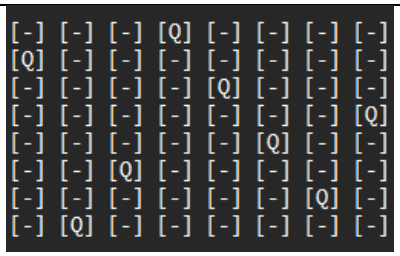
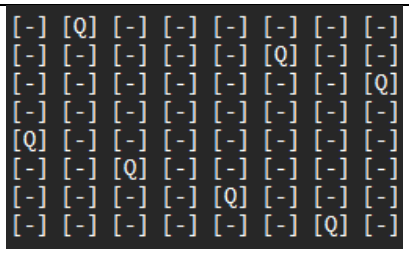
SOUZA, Handerson Muller Ferro de. Algoritmo eficiente para validação de soluções para o problema das n-rainhas. 2019. 43 f. TCC (Graduação) - Curso de Ciência da Computação, Ciências Exatas e da Terra., Universidade Federal de Alagoas, Arapiraca, 2019. Disponível em: https://ud10.arapiraca.ufal.br/repositorio/publicacoes/3165. Acesso em: 05 mar. 2022.

TURNER, Ben. **Mathematician cracks 150-year-old chess problem**. 2022. Disponível em: https://www.livescience.com/150-year-chess-problem-solved. Acesso em: 05 mar. 2022.

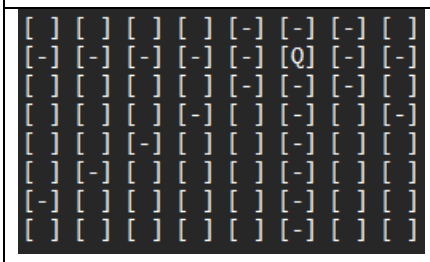
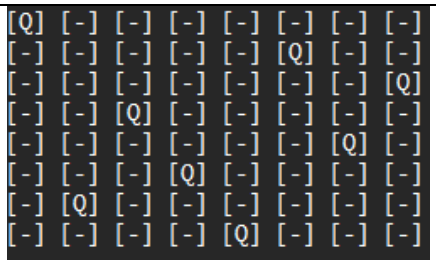
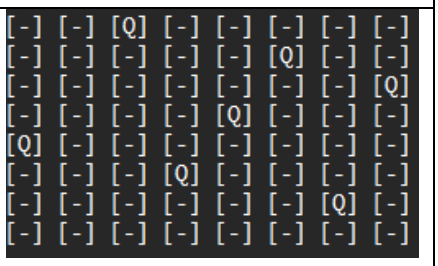
## Anexo I

Estados iniciais dos tabuleiros para cada teste

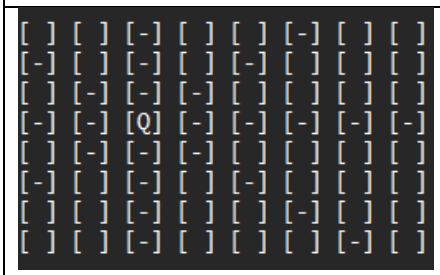
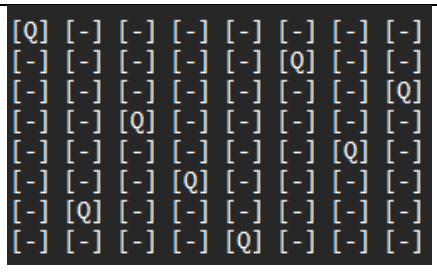
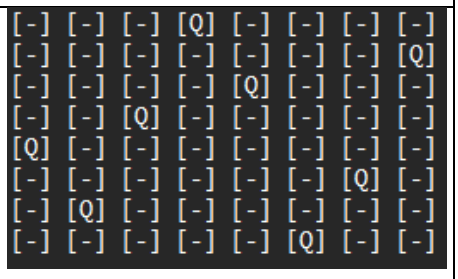
Teste 1:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

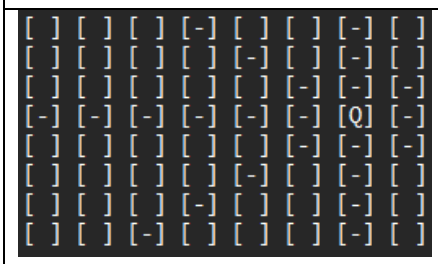
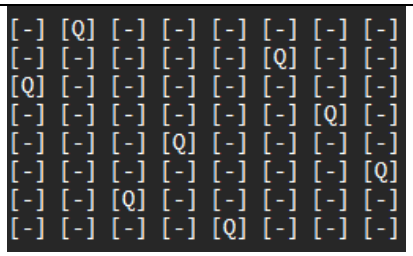
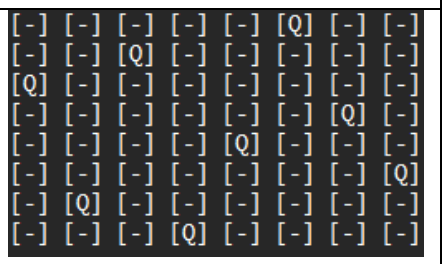
Teste 2:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

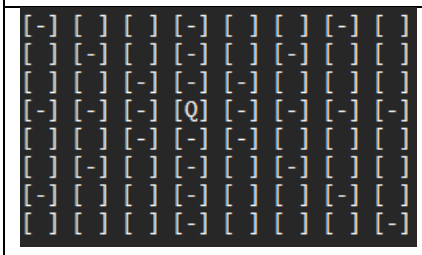
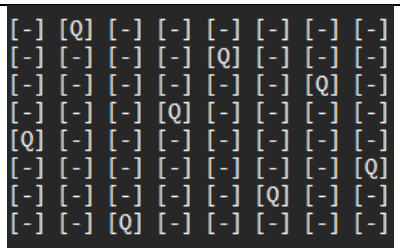
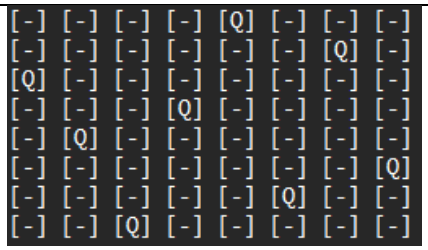
Teste 3:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

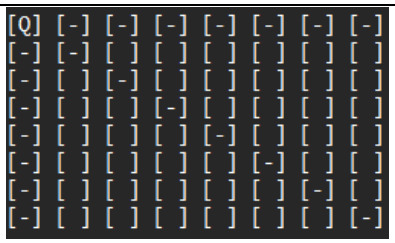
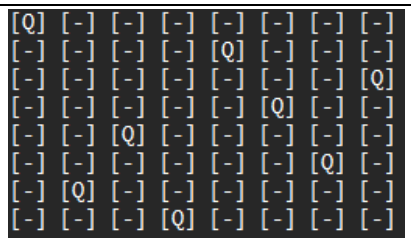
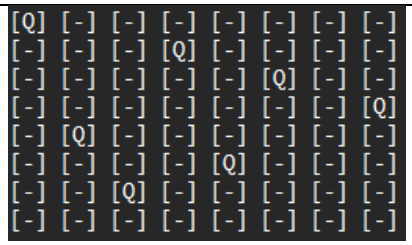
Teste 4:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

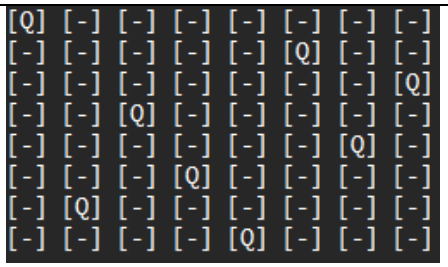
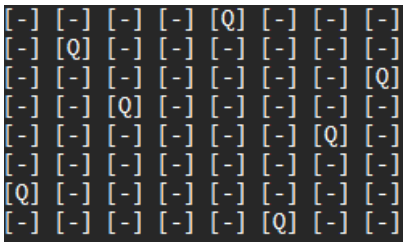
Teste 5:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

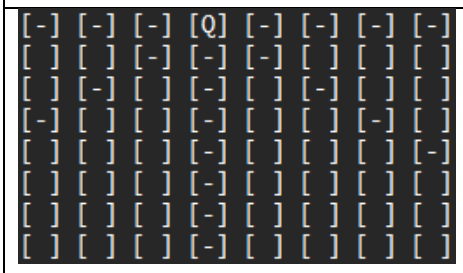
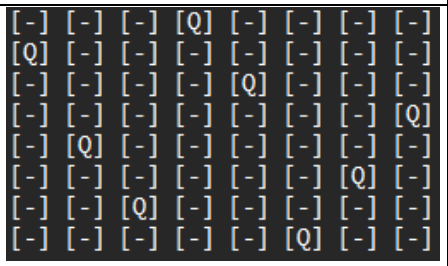
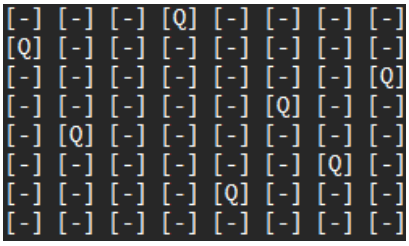
Teste 6:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

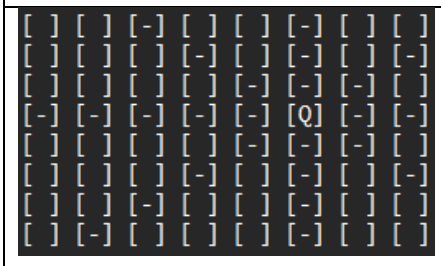
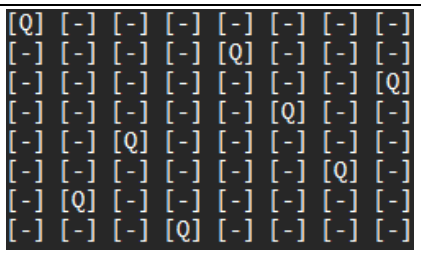
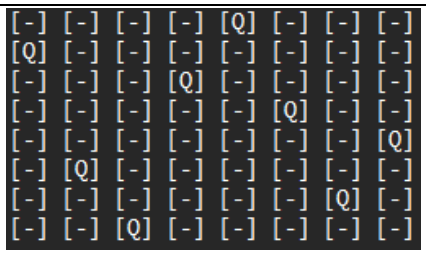
Teste 7:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

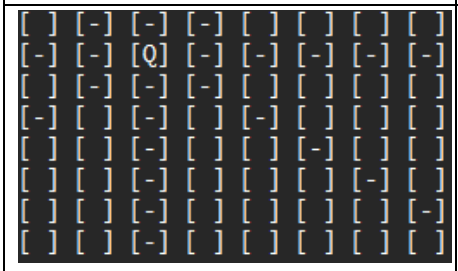
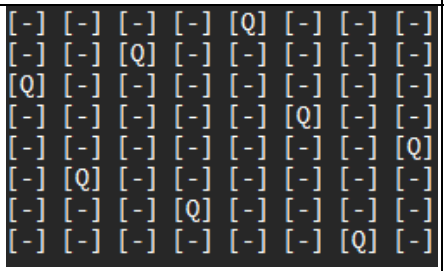
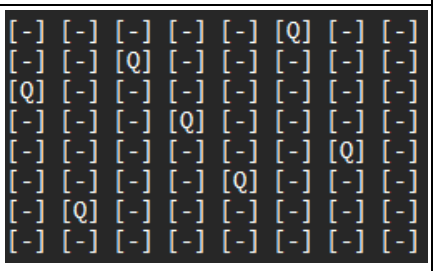
Teste 8:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

Teste 9:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |

Teste 10:

| Estado Inicial | Estado final utilizando busca por profundidade limitada | Estado final utilizando busca subindo o morro |
|---|---|---|
|  |  |  |