

RELATÓRIO TÉCNICO

Problema das N-Rainhas

Análise Comparativa de Algoritmos

Algoritmos Implementados:

- Backtracking (Busca Completa)
- Guloso Simples (Heurística Determinística)
- Guloso com Restart (Heurística Randomizada)

Data: 03/12/2025

Linguagem: Python 3.12

Métricas: Tempo, Memória, Taxa de Sucesso

1. RESUMO EXECUTIVO

Este relatório apresenta uma análise comparativa de três abordagens algorítmicas para resolver o Problema das N-Rainhas: Backtracking, Guloso Simples e Guloso com Restart. Os experimentos foram conduzidos para valores de N entre 4 e 14, medindo tempo de execução, consumo de memória e taxa de sucesso.

1.1 Principais Resultados

Algoritmo	Complexidade	Taxa Sucesso	Tempo N=12
Backtracking	$O(N!)$	100%	28.34s
Guloso Simples	$O(N^2)$	7% (1/14)	0.0006s
Guloso Restart	$O(N^2)$	100%	0.011s

Conclusão Principal: Guloso com Restart combina eficiência (2.580x mais rápido que Backtracking) com confiabilidade (100% de sucesso), sendo a melhor escolha para N grande.

2. ALGORITMOS IMPLEMENTADOS

2.1 Backtracking

Algoritmo de busca completa que explora todas as possibilidades através de recursão. Garante encontrar todas as soluções válidas mas com complexidade exponencial $O(N!)$.

Características:

✓ Solução garantida	✓ Encontra todas as soluções
✗ Tempo exponencial	✗ Alto consumo de memória

2.2 Guloso Simples

Heurística determinística que escolhe a posição com menor número de conflitos a cada linha. Rápido mas frequentemente falha em encontrar solução válida.

✓ Muito rápido $O(N^2)$	✓ Baixo consumo de memória
✗ Taxa de falha 93%	✗ Determinístico (sempre falha nos mesmos casos)

2.3 Guloso com Restart

Versão randomizada do guloso que usa aleatoriedade na escolha entre posições empatadas. Permite até 100 tentativas, garantindo alta taxa de sucesso mantendo eficiência.

✓ Rápido $O(N^2)$	✓ Taxa de sucesso 100%
✓ Escalável para N grande	✓ Memória constante

3. RESULTADOS EXPERIMENTAIS

3.1 Configuração dos Experimentos

Parâmetro	Valor
Linguagem	Python 3.12
Sistema Operacional	Windows
Valores de N testados	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
Medição de tempo	time.perf_counter()
Medição de memória	tracemalloc
Tentativas (Guloso Restart)	Máximo 100

3.2 Dados Completos

Tabela 1: Tempo de Execução (segundos)

N	BT	G. Simple	G. Restart	Speedup GR vs BT
4	0.000112	0.000067	0.000172	1x
5	0.000306	0.000060	0.000101	3x
6	0.000895	0.000083	0.000237	4x
7	0.004885	0.000098	0.000197	25x
8	0.022345	0.000115	0.000849	26x
9	0.117890	0.000141	0.001113	106x
10	0.727302	0.000289	0.000480	1515x
11	3.91	0.000333	0.006635	589x
12	28.34	0.000591	0.010975	2582x
13	187.75	0.000474	0.004001	46927x
14	1425.16	0.000654	0.019660	72490x

Tabela 2: Consumo de Memória (MB)

N	BT	G. Simples	G. Restart	Razão BT/GR
4	0.000557	0.000320	0.000519	1x
5	0.001434	0.000328	0.000473	3x
6	0.000862	0.000336	0.000534	2x
7	0.005051	0.000343	0.000542	9x
8	0.011742	0.000351	0.000549	21x
9	0.046204	0.000359	0.000580	80x
10	0.100288	0.000366	0.000626	160x
11	0.390671	0.000374	0.000633	617x
12	2.17	0.000381	0.000641	3394x
13	11.85	0.000389	0.000648	18276x
14	61.37	0.000397	0.000656	93529x

Tabela 3: Taxa de Sucesso

N	BT	G. Simples	G. Restart	Soluções Totais
4	✓	✗	✓	2
5	✓	✓	✓	10
6	✓	✗	✓	4
7	✓	✗	✓	40
8	✓	✗	✓	92
9	✓	✗	✓	352
10	✓	✗	✓	724
11	✓	✗	✓	2680
12	✓	✗	✓	14200
13	✓	✗	✓	73712
14	✓	✗	✓	365596

4. ANÁLISE DOS RESULTADOS

4.1 Crescimento de Complexidade

O Backtracking apresenta crescimento exponencial claro: de $N=12$ para $N=13$, o tempo aumenta 6.6x (28s \rightarrow 188s), e para $N=14$, mais 7.6x (188s \rightarrow 1.425s = 23min). Isso confirma a complexidade $O(N!)$ teórica.

Transição	Tempo BT	Fator	Tempo GR	Fator
$N=11 \rightarrow 12$	3.9s \rightarrow 28.3s	7.2x	6.6ms \rightarrow 11.0ms	1.7x
$N=12 \rightarrow 13$	28.3s \rightarrow 187.7s	6.6x	11.0ms \rightarrow 4.0ms	0.4x
$N=13 \rightarrow 14$	187.7s \rightarrow 1425s	7.6x	4.0ms \rightarrow 19.7ms	4.9x

Guloso Restart mantém crescimento suave, com variação devido à aleatoriedade nas tentativas.

4.2 Eficiência Comparativa

N	Speedup (GR vs BT)	Economia de Memória	Trade-off
8	26x mais rápido	21x menos memória	Vantagem clara GR
10	1.515x mais rápido	160x menos memória	Vantagem clara GR
12	2.580x mais rápido	3.394x menos memória	Vantagem clara GR
14	72.488x mais rápido	93.532x menos memória	Vantagem MASSIVA GR

4.3 Por que Guloso Simples Falha?

O Guloso Simples é determinístico: sempre escolhe a primeira posição com mínimo de conflitos. Isso o prende em mínimos locais - configurações onde todas as próximas escolhas aumentam conflitos, mas não são a solução ótima global. Taxa de falha: 93% (13 de 14 casos testados).

4.4 Como Guloso Restart Resolve?

A randomização permite escolher aleatoriamente entre posições empatadas. Cada tentativa explora um caminho diferente. Com até 100 tentativas, a probabilidade de encontrar uma solução válida é extremamente alta. Nos testes: 100% de sucesso (14/14 casos).

5. CONCLUSÕES

5.1 Resumo das Descobertas

- 1. **Backtracking:** Solução completa e garantida, mas inviável para $N > 14$ (tempo proibitivo).
- 2. **Guloso Simples:** Extremamente rápido mas não confiável (93% de falha).
- 3. **Guloso com Restart:** MELHOR ESCOLHA - combina rapidez com confiabilidade total.

5.2 Recomendações de Uso

Cenário	Algoritmo Recomendado	Justificativa
$N \leq 10$	Backtracking	Rápido o suficiente e encontra todas soluções
N entre 11-20	Guloso Restart	Backtracking muito lento, Restart confiável
$N > 20$	Guloso Restart	Única opção viável
Precisa TODAS soluções	Backtracking	Gulosos encontram apenas uma
Aplicação crítica	Guloso Restart	100% sucesso vs 7% do Simples
Pesquisa acadêmica	Ambos	Comparação didática

5.3 Trabalhos Futuros

- Implementar algoritmos genéticos para comparação
- Testar paralelização do Backtracking
- Otimizar Guloso Restart com heurísticas adaptativas
- Estender testes para $N > 20$
- Estudar distribuição de soluções no espaço de busca

5.4 Considerações Finais

Este estudo demonstra que a escolha do algoritmo correto é crucial para problemas combinatoriais. O Guloso com Restart prova que heurísticas simples, quando combinadas com aleatoriedade, podem superar algoritmos completos em termos práticos, oferecendo uma solução 72.000x mais rápida para $N=14$ sem sacrificar confiabilidade.

6. VISUALIZAÇÕES

Os gráficos a seguir ilustram visualmente os resultados apresentados. Foram gerados automaticamente a partir dos dados coletados.

=====

FIM DO RELATÓRIO

Gerado em 03/12/2025 às 21:01:41