TRABALHO PRÁTICO

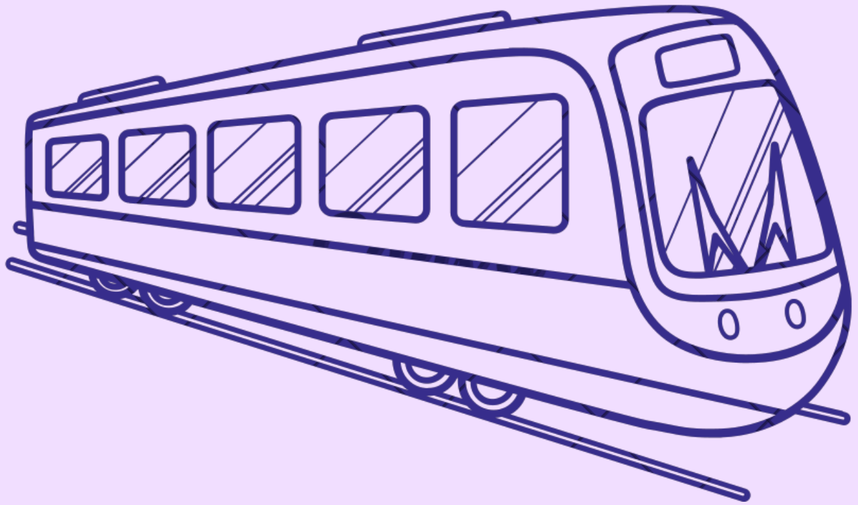
Projeto e Análise de Algoritmos

Tópicos:

Victor Hugo Braz

Vinicius Dutra Goddard

ENTENDENDO O PROBLEMA



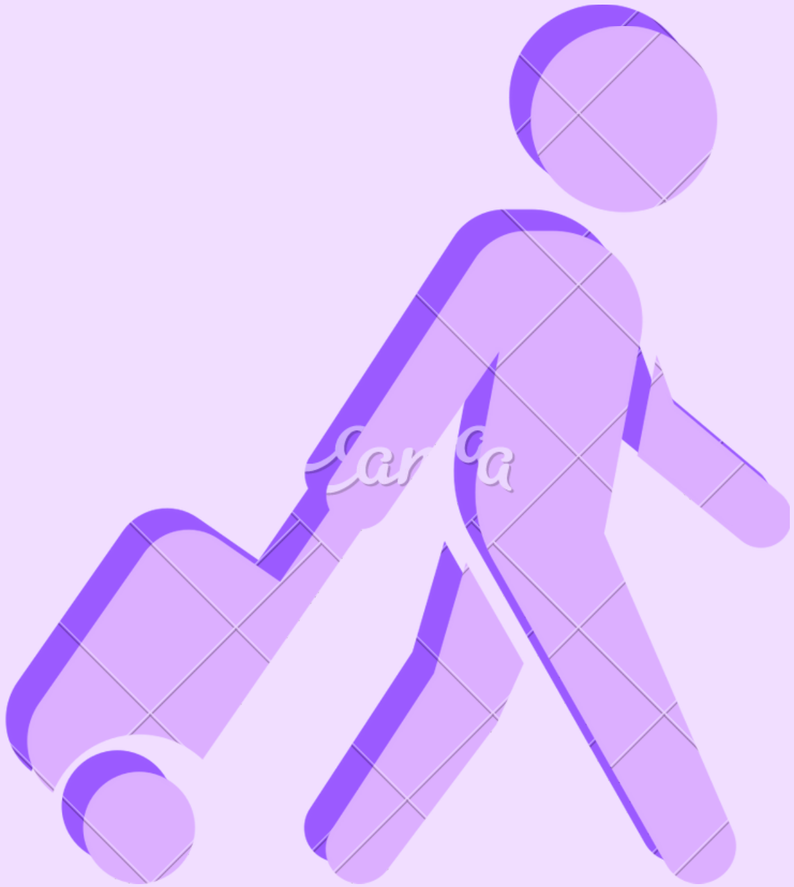
ENTENDENDO O PROBLEMA

Problema A)

Encontrar o número máximo de estações que um turista poderá visitar com um só passe

Variante do problema do Caixeiro Viajante / Travelling Salesman

Problema de Otimização
NP-hard
Complexidade Fatorial O(N!)



ENTENDENDO O PROBLEMA

Problema A) é fundamentado na teoria de grafos.

Na verdade, estamos buscando o MAIOR ciclo hamiltoniano. [1]

Djikstra propós um algoritmo em tempo linear para resolver esse problema quando o grafo é uma árvore. [2]

Para grafos de intervalo, existe um algoritmo que resolve esse problema em $O(n^4)$ [3]

'Longest Simple Cycle'

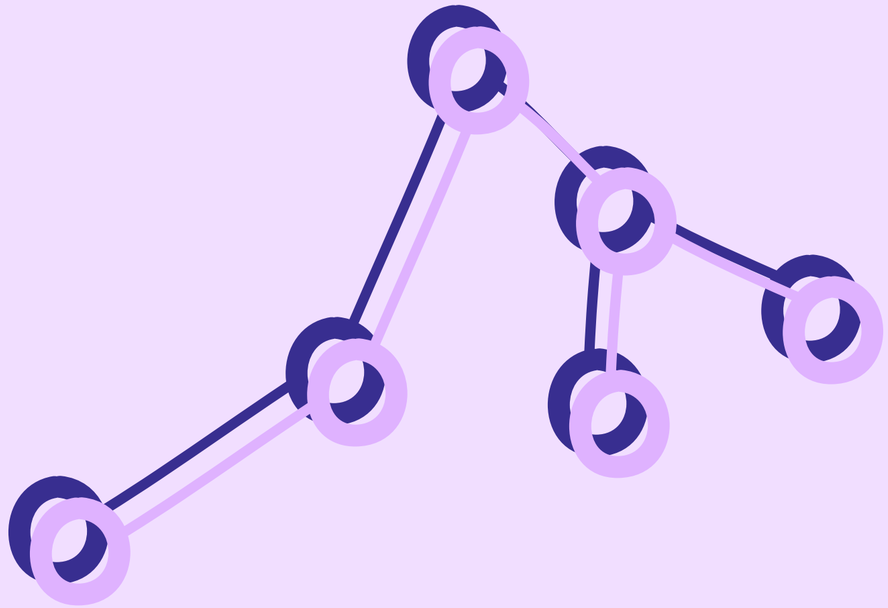
ENTENDENDO O PROBLEMA

Problema B)

Determinar as estações de guiche... Para que o turista não precise caminhar mais do que uma estação para encontrar um guiche.

Variante do problema do Conjunto Dominante Mínimo / Dominant Set

Problema de Otimização
NP-hard
Complexidade Exponencial $O(2^n)$



ENTENDENDO O PROBLEMA

Problema B) é, também fundamentado em grafos.

Queremos escolher o menor número de pontos para que todos os outros estejam ligados a pelo menos um deles.

‘Dominating Set Problem’

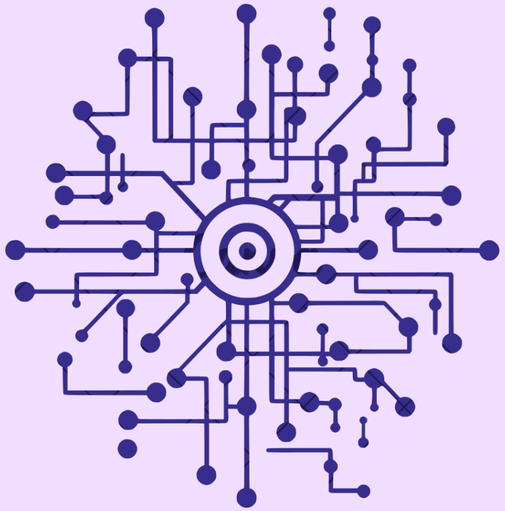
Fomin, Grandoni & Kratsch (2009) elaboraram um algoritmo que resolve esse problema em $O(1.5137^n)$ [4]

Com programação dinâmica, e se o grafo for uma arvore, pode ser resolvido em $O(n)$ [5]

ALGORITMOS E METODOLOGIA

Três tipos de algoritmos utilizados:

- Força-Bruta (Com Backtracking)
- Branch-and-Bound
- Algoritmo Guloso Aproximado



Força-Bruta (Com Backtracking)

- Solução exata
- Itera sobre quase todas as soluções possiveis.
- Complexidade exponencial
- Salva o melhor caminho. Poda se o melhor caso for pior que o melhor caminho salvo

Branch-and-Bound

- Solução exata
- Possui mais overhead. Não é tão eficiente para problemas com baixa possibiliade de poda
- Complexidade exponencial
- Desiste de caminhos ruins antes mesmo de tentar. Heuristica melhor do que força bruta

Algoritmo Guloso Aproximado

- Solução Aproximada
- Extremamente rapido. Para ambos os problemas, terminou em menos de 1 segundo.
- Complexidade Polinomial
- Segue apenas um caminho, que é o melhor localmente, mas não garante melhor solução global.

Metodologia de Testes: Problema A

- Todos os algoritmos rodaram até o final. (Após incluir Backtracking no Força Bruta)
- Foi tomada a média entre 10 testes.
- Dados salvos: Chamadas de recursão, Nós visitados, tempo gasto, caminho final.

Metodologia de Testes: Problema A

- No pior caso (Um grafo conexo), resolver esse problema por força bruta precisaria de cerca de $1.27*10^{32}$ operações.
- Como o grafo utilizado para testes era esparço (48 vértices, 68 arestas), seriam necessárias ~10^{24} operações.
- Esse tipo de problema apresenta baixo potencial para poda. Branch and Bound não foi eficiente.

Metodologia de Testes: Problema B

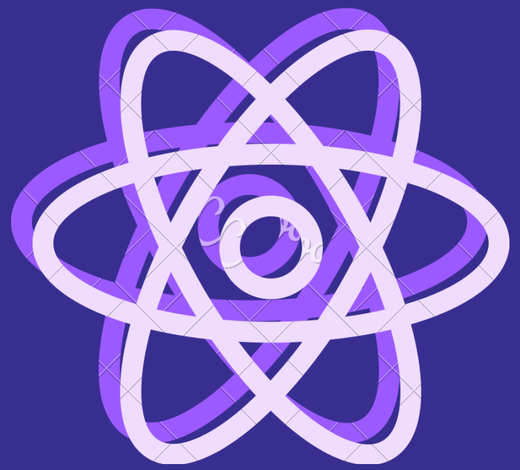
- Algoritmo de força bruta NÃO rodou até o final. (Foi necessário reduzir o tamanho do grafo).
- Também foi feita a média entre 10 testes.
- Dados salvos: Chamadas de recursão, tempo gasto, conjunto dominante (Aonde por os guiches)

Metodologia de Testes: Problema B

- Apesar de ser exponêncial e não fatorial, demorou mais que o problema A com o algoritmo de Força Bruta com Bactkracking
- Um grafo com 46 vérticas gera 2^{46} combinações. Demorária dias ou até meses para resolver com força bruta pura.
- Possui ótimo critério de poda para branch and bound. Grande ganho de performance em relação ao problema A.

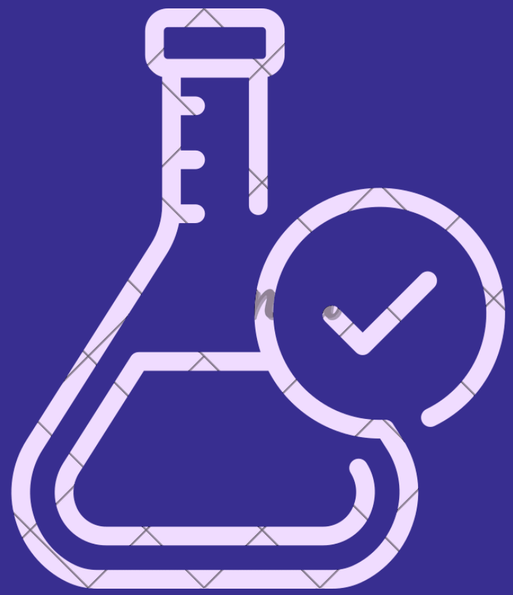
Técnologias Utilizadas

Python



React

TESTES E RESULTADOS



Qualidade dos Resultados

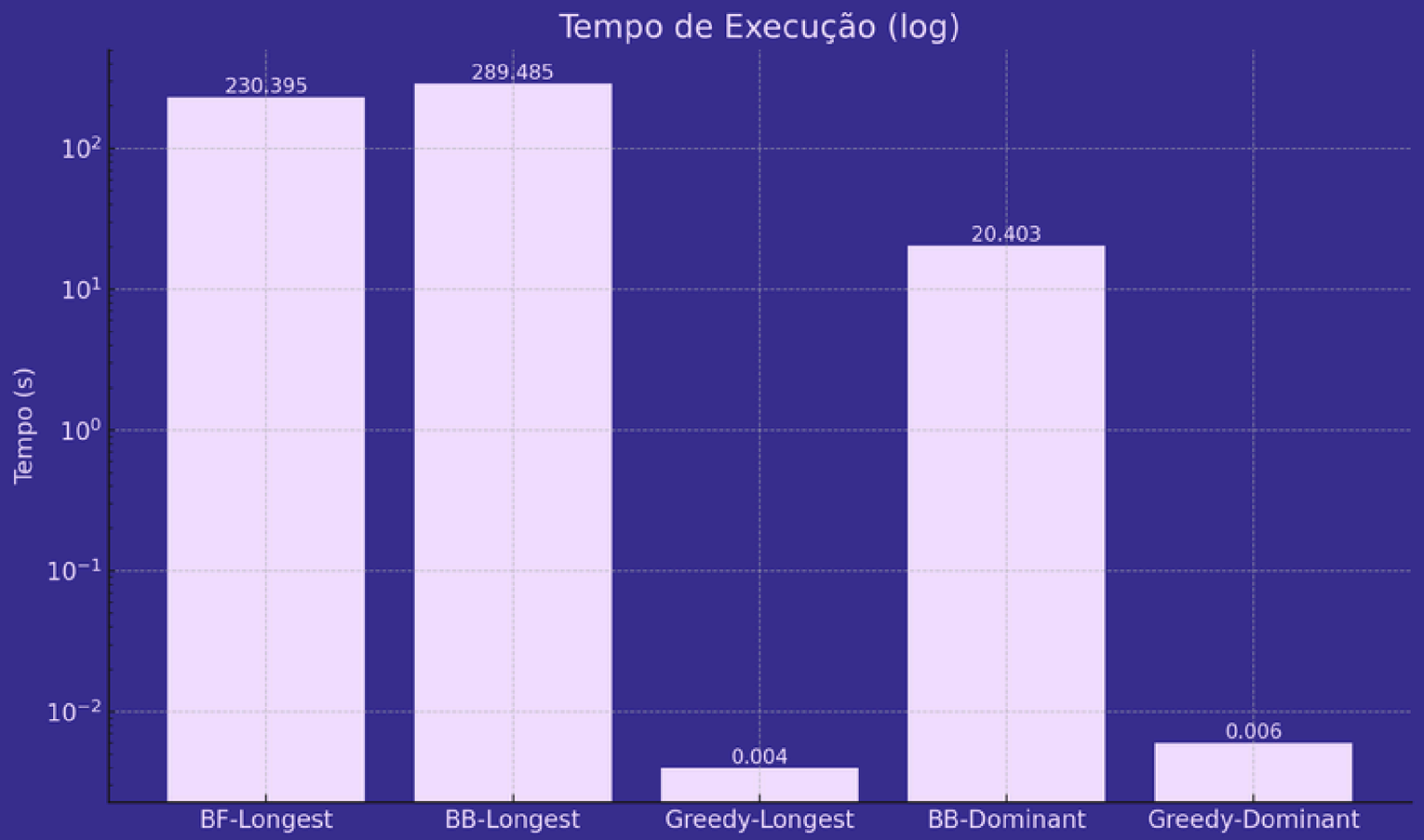
Problema A:

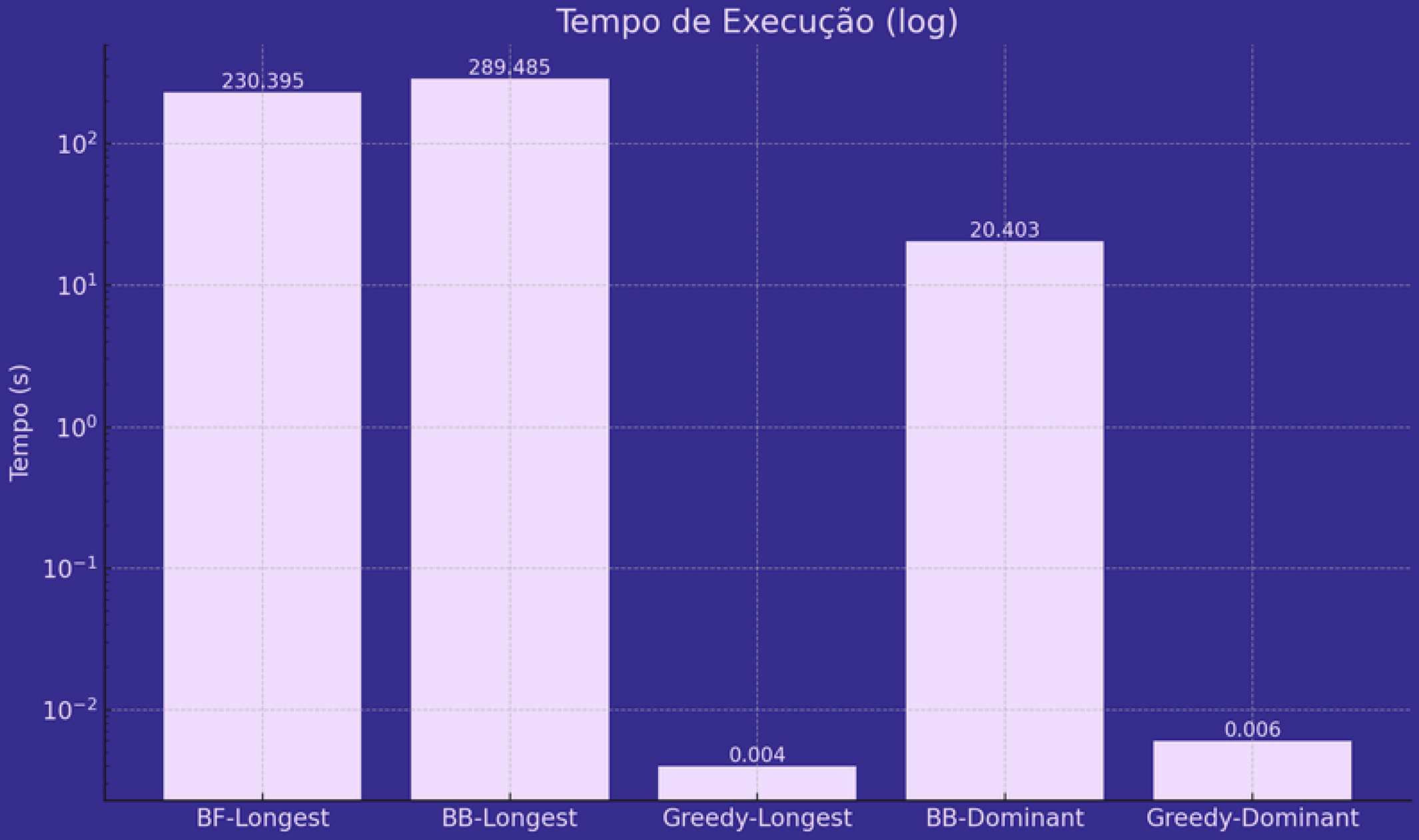
- Força bruta e Branch and Bound dão a resposta exata: Caminho de tamanho **33**
- Heuristica gulosa é extremamente rápida, mas a resposta é muito pior: **23**

Problema B:

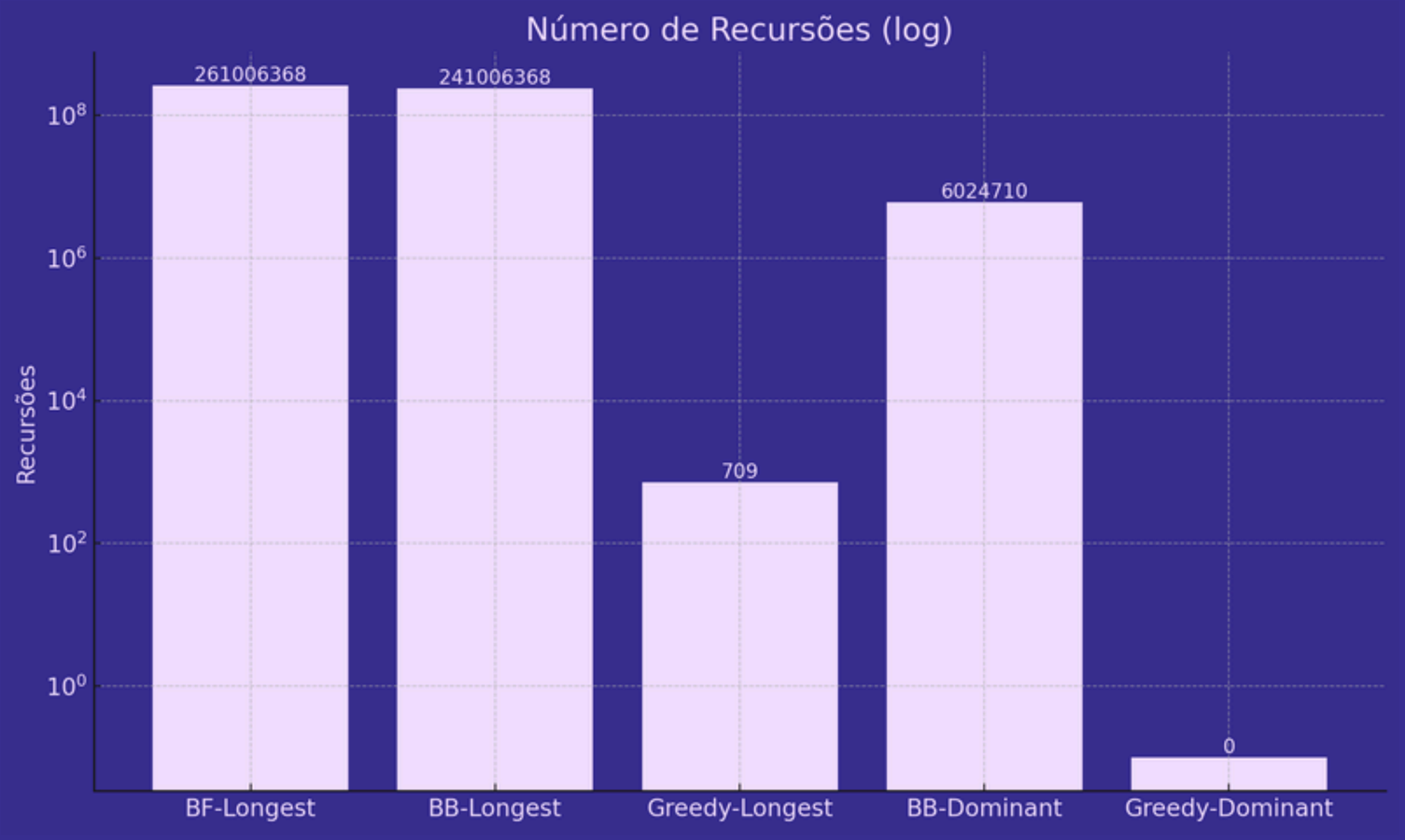
- Força bruta não rodou. Branch and bound deu um conjunto de estações de tamanho **14**
- Heuristica gulosa foi extremamente rápida. Gerou um conjunto diferente de estações, também de tamanho **14**

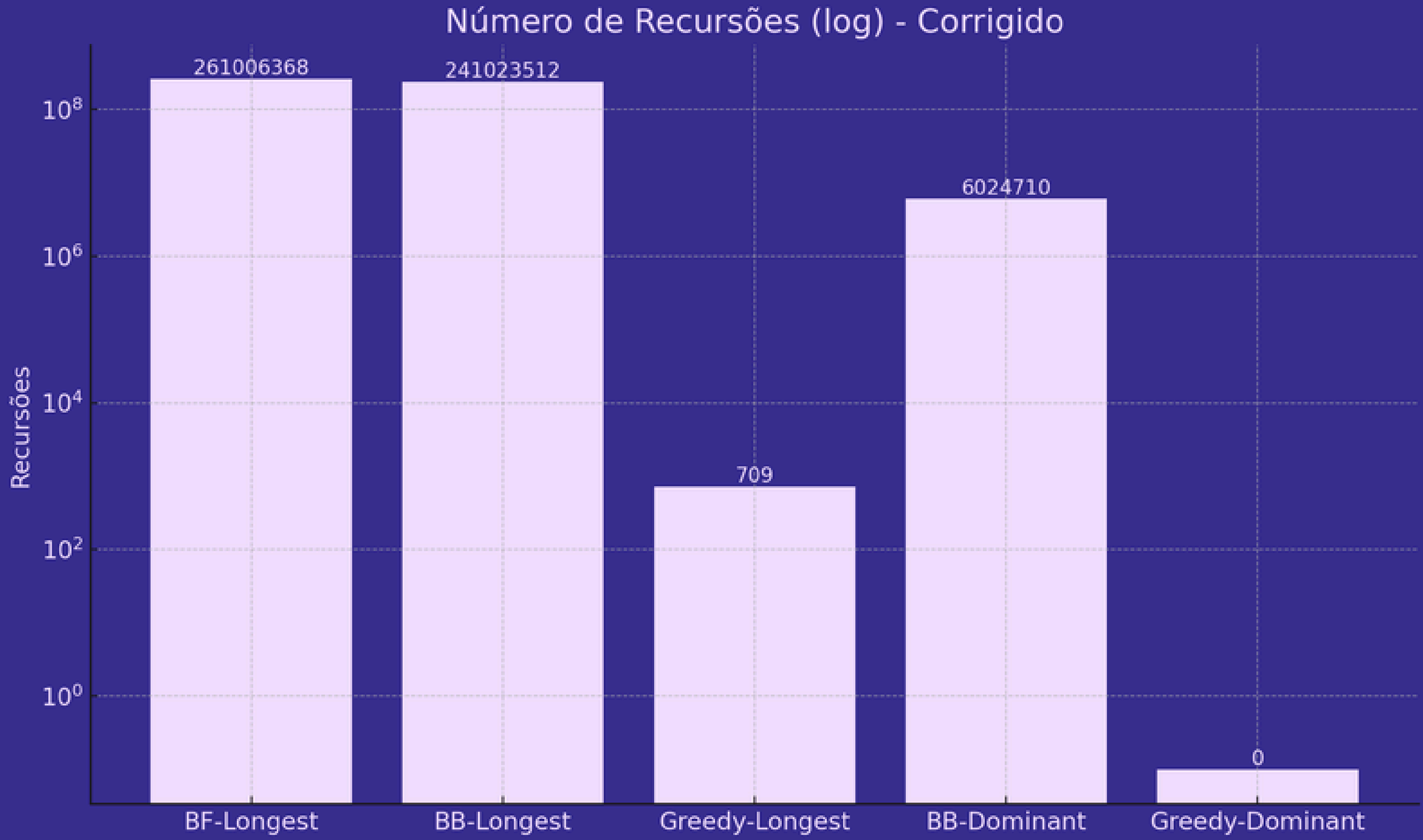
Velocidade de Execução




Tempo de Execução (log)
Tempo (s)
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
230.395
289.485
0.004
20.403
0.006
BF-Longest
BB-Longest
Greedy-Longest
BB-Dominant
Greedy-Dominant

Chamadas Recursivas




Número de Recursões (log) - Corrigido
Recursões
10^8
10^6
10^4
10^2
10^0
261006368
241023512
709
6024710
0
BF-Longest
BB-Longest
Greedy-Longest
BB-Dominant
Greedy-Dominant

CONCLUSÃO

Branch and Bound possui overhead, e nem sempre é melhor do que Backtracking.

Heuristica gulosa é EXTREMAMENTE rapida. Mas a acurácia da solução pode variar.

Um problema com uma complexidade maior (N! vs 2^n) pode possuir melhores critérios de poda, e, logo, tempo real menor.

O algoritmo a ser usado depende do problema sendo abordado.

Não existe bala de prata.

Bibliografia:

1 - Schrijver, Alexander (2003), Combinatorial Optimization: Polyhedra and Efficiency, Volume 1, Algorithms and Combinatorics, vol. 24, Springer, p. 114, ISBN 9783540443896.

2 - Bulterman, R.W.; van der Sommen, F.W.; Zwaan, et al. (2002), "On computing a longest path in a tree", Information Processing Letters, 81 (2): 93–96, doi:10.1016/S0020-0190(01)00198-3

3 - Ioannidou, Kyriaki et al (2011), "The longest path problem has a polynomial solution on interval graphs", Algorithmica, 61 (2): 320–341, CiteSeerX 10.1.1.224.4927, doi:10.1007/s00453-010-9411-3, S2CID 7577817.

4 - Fomin, Fedor V.; Grandoni, Fabrizio; Kratsch, Dieter (2009), "A measure & conquer approach for the analysis of exact algorithms", Journal of the ACM, 56 (5): 25:1–32, doi:10.1145/1552285.1552286, S2CID 1186651

5 - T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. Fundamentals of Domination in Graphs. Marcel Dekker Inc., 1998

INTERFACE WEB

Obrigado!

Victor Hugo Braz **Vinicius Dutra Goddard**