Pontifícia Universidade Católica de Minas Gerais

Fundamentos de Projeto e Análise de Algoritmos

Relatório do Trabalho Prático em Grupo

Caio Elias Rodrigues Araujo e Victor de Souza Friche Passos

2024

Belo Horizonte

# Informações

Critério de avaliação escolhido: **Individual**

Linguagem de programação: **Java**

Algoritmos implementados: **Backtracking e Programação Dinâmica**

Link do repositório: https://github.com/DisciplinasProgramacao/projetodealgoritmos-leilaoenergia-projetodealgoritmos\_leilaoenergia

Divisão de tarefas

|  |  |
| --- | --- |
| **Integrante** | **Tarefas executadas** |
| Victor | * Implementação do método de Backtracking * Elaboração da seção Backtracking, Informações,   Comparação entre Métodos e Considerações Finais   * Participação na seção de Comparação |
| Caio | * Implementação do método de Programação Dinâmica * Participação na seção de Comparação |

**Backtracking**

Victor de Souza Friche Passos

**Primeira implementação (Corresponde a Backtracking no repositório) Estratégia Geral:**

Definem-se variáveis para armazenar a energia total disponível (energiaTotal), os lances disponíveis (lances), a melhor combinação de lances (melhorLances), o melhor valor total obtido (melhorValor) e a energia total utilizada na melhor combinação (energiaUtilizada).

A função encontrarMelhorVenda é a função principal que inicializa o processo de busca e, ao final, imprime os melhores resultados encontrados. Ela chama recursivamente uma funcao encontrarMelhorVenda de backtracking com parâmetros que tenta todas as combinações possíveis de lances, respeitando o limite de energia total.

A cada combinação de lances, se a soma dos valores dos lances for maior do que o melhor valor obtido até o momento (melhorValor), a combinação atual é armazenada como a melhor solução.

**Critério de Poda:**

O critério de poda é garantir que a soma da energia dos lances atuais não exceda a energia total disponível. Isto é feito com a verificação “if (energiaUsada + lances[i].megawatts <= energiaTotal)” antes de adicionar um novo lance à combinação atual.

**Passo a passo:**

1. Inicializa as variáveis.
2. Início da busca: a função *encontrarMelhorVenda* chama a função recursiva *encontrarMelhorVenda* com uma lista vazia para lances atuais, índice inicial 0, energia usada 0 e valor atual 0.
3. Verificação de condição: a função recursiva verifica se a energia utilizada até o momento não excede a energia total disponível. Se sim, continua a busca; se não, volta (backtrack).

1. Se a combinação atual de lances proporciona um valor total maior do que o melhor valor obtido até o momento, atualiza-se a melhor solução.
2. Para cada lance a partir do índice atual, tenta-se adicionar o lance à combinação atual, respeitando o limite de energia, e a busca recursiva continua com o próximo lance.
3. Após tentar um lance, o lance é removido da combinação atual para explorar outras combinações possíveis.
4. Quando todas as combinações foram exploradas, a melhor solução é impressa.

**Testes:**

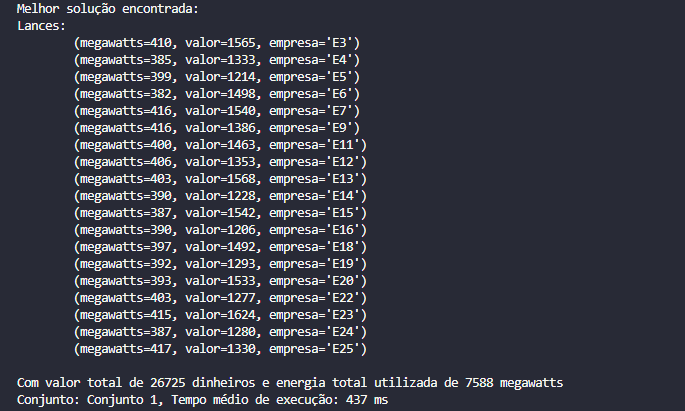
**Teste 1**

**\* Conjunto de empresas interessadas 1 (nome, quantidade, valor):**

**E1;430;1043  
E2;428;1188  
E3;410;1565  
E4;385;1333  
E5;399;1214  
E6;382;1498  
E7;416;1540  
E8;436;1172  
E9;416;1386  
E10;423;1097  
E11;400;1463  
E12;406;1353  
E13;403;1568  
E14;390;1228  
E15;387;1542  
E16;390;1206  
E17;430;1175  
E18;397;1492  
E19;392;1293  
E20;393;1533  
E21;439;1149  
E22;403;1277  
E23;415;1624  
E24;387;1280  
E25;417;1330**

**Considerando 8000 megawatts.**

**Resultado:**

****

A distribuição de energia entre as empresas participantes no leilão resultou em uma utilização total de 7588 megawatts dos 8000 megawatts disponíveis. Esse resultado foi obtido após a execução do algoritmo de backtracking que buscou maximizar o valor total dos lances aceitos, que durou 437ms, ou seja, nem 1 segundo! O valor total obtido com a melhor combinação de lances (19 lances) foi de 26725 dinheiros. Esse é o valor máximo encontrado pelo algoritmo considerando todas as combinações possíveis dentro da restrição de energia.

**Teste 2**

**\* Conjunto de empresas interessadas 2 (nome, quantidade, valor):**

**E1;430;1043  
E2;428;1188  
E3;410;1565  
E4;385;1333  
E5;399;1214  
E6;382;1498  
E7;416;1540  
E8;436;1172  
E9;416;1386  
E10;423;1097  
E11;400;1463  
E12;406;1353  
E13;403;1568  
E14;390;1228  
E15;387;1542  
E16;390;1206  
E17;430;1175  
E18;397;1492  
E19;392;1293  
E20;393;1533  
E21;439;1149  
E22;403;1277  
E23;415;1624  
E24;387;1280  
E25;417;1330  
E1;313;1496  
E2;398;1768  
E3;240;1210  
E4;433;2327  
E5;301;1263  
E6;297;1499  
E7;232;1209  
E8;614;2342  
E9;558;2983  
E10;495;2259  
E11;310;1381  
E12;213;961  
E13;213;1115  
E14;346;1552  
E15;385;2023  
E16;240;1234  
E17;483;2828  
E18;487;2617  
E19;709;2328  
E20;358;1847  
E21;467;2038  
E22;363;2007  
E23;279;1311  
E24;589;3164  
E25;476;2480**

**Considerando 8000 megawatts.**

**Resultado:** ao realizar o segundo teste com um conjunto maior de lances no leilão, o algoritmo de backtracking não conseguiu concluir a execução dentro de um tempo razoável. Isso porque no teste 2 foi utilizado um conjunto de dados aproximadamente 2 vezes maior (25 para 50 lances) e a complexidade do algoritmo de backtracking aumenta exponencialmente de acordo com o tamanho da entrada. Isso indica que o critério de poda usado pode precisar ser mudado ou complementado para conseguir solucionar esse conjunto de dados.

**Teste 3**

**Gerador de Problemas:** o algoritmo de geração de problemas é utilizado para gerar conjuntos de lances (ofertas de empresas) de forma randômica dentro de um intervalo.

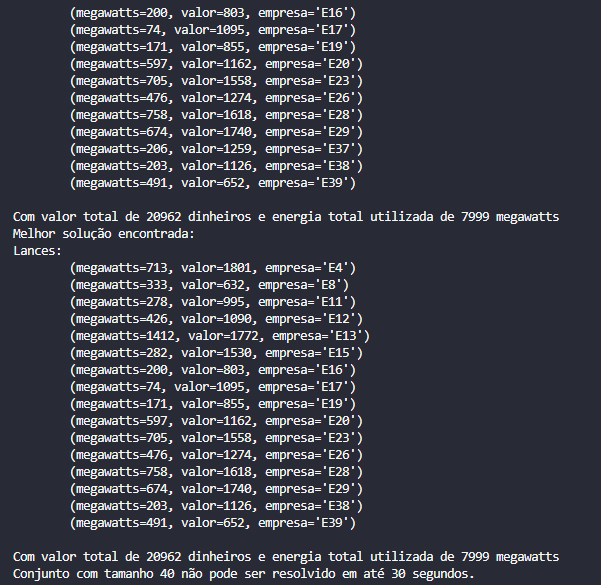
Informações importantes:

TAM\_BASE é um valor de referência escolhido como ponto de partida para a geração de valores. Ele é uma constante que facilita a criação de valores de teste escaláveis.

**Dispersão** é um parâmetro que controla a variabilidade dos valores gerados. Ela permite que o algoritmo seja testado com diferentes graus de variação nos dados de entrada.

Utilizando TAM\_BASE = 13 e Dispersão de 0.5 são utilizados valores parecidos com os conjuntos de dados anteriores já testados.

**Resultado:** no terminal o resultado ficou gigante, então vou deixar uma print do final da execução e um arquivo mostrando pelo o menos as 10 execuções de teste para o conjunto 39.



O algoritmo rodou/testou 10 conjuntos para cada tamanho T parando em T = 40 lances. 39 foi o último número de lances que não excedeu o tempo limite de 30 segundos, chegando a executar na média 7319 ms ou aproximadamente 7 segundos e meio.

10 testes de conjuntos de 40 lances ultrapassaram 30 segundos e o algoritmo parou sua execução. Confira no arquivo abaixo:

https://drive.google.com/file/d/1GpqgF94-MGX3IMJhM5nnQEWg0C78ZTZF/view?usp=sharing