PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Bacharelado em Engenharia de Software

Projeto e Análise de Algoritmos

Guilherme Julio

Lebana Martins

Maria Eduarda

Victor Augusto

DOCUMENTAÇÃO DE CÓDIGO E TESTES:

Problema do Supermercado - Guilherme Julio

Belo Horizonte

2020

Guilherme Julio
Lebana Martins
Maria Eduarda
Victor Augusto

DOCUMENTAÇÃO DE CÓDIGO E TESTES:

Problema do Supermercado - Guilherme Julio

Documentação referente ao designado trabalho teórico prático de Projeto e Análise de Algoritmos, do problema do supermercado desenvolvido na disciplina de Projeto e Análise de Algoritmos, do curso de bacharelado em Engenharia de Software, da Pontifícia Universidade Católica de Minas Gerais, unidade Praça da Liberdade.

Orientador: Prof. João Caram

Belo Horizonte
2020

SUMÁRIO

1 IMPLEMENTAÇÃO DE CÓDIGO

Essa seção, é destinada em mostrar os detalhes da implementação do algoritmo do problema do supermercado, abrangendo as classes que foram implementadas, abstração realizada, bem como detalhes adicionais.

1.1 Abstração

Conforme estudado em sala de aula, o carrinho de supermercado é uma adaptação do problema da mochila. Com o objetivo de realizar uma técnica de abstração, foi optado a criação de classes que representassem o problema, com isso nós temos que:

Classe Carrinho de supermercado: Representa a instância de um carrinho de supermercado, o carrinho possui uma lista de produtos, bem como um peso máximo, e orçamento máximo.

Classe Lista de produtos: Uma lista de produtos, possui um ArrayList de produtos, um preço da lista, e um peso da lista.

Classe Produtos: Representa 1 produto, com valor e peso, classe fornecida pelo professor.

1.2 Escopo definido e execução de código

O escopo definido para o código, foi o solicitado no item b. do trabalho da disciplina, onde:

Figura 1 - Escopo

b) Dentre os demais problemas (P2 a P7), escolher UM deles e implementar sua solução com força bruta e mais uma técnica à sua escolha. Neste caso, deve ser investigado (a) o tempo necessário para resolver instâncias crescentes do problema proposto com as duas técnicas e (b) o tamanho do maior conjunto possível com solução obtida por força bruta no tempo-limite de 5 segundos.

Fonte: Enunciado do trabalho

Tendo em mente o escopo do enunciado do trabalho, a execução das técnicas definidas, é baseado em instâncias crescentes no código, ou seja, a finalização de execução de uma técnica, a instância cresce de acordo com uma determinação no código em uma estrutura de repetição.

1.3 Técnicas definidas

As técnicas no qual o problema da mochila foi desenvolvido, são: Força Bruta e Algoritmo Guloso. A força bruta, é a técnica obrigatória definida no escopo, e o algoritmo guloso a próxima segunda técnica escolhida.

1.4 Limites dos algoritmos

Para a técnica de força bruta, o limite máximo de instâncias é de 24 produtos, a instância inicial começa em 4, e avança de 2 em 2, até chegar a 24 produtos. Para o algoritmo guloso, o limite máximo de instâncias é de 1.048.576 produtos, onde a instância cresce de n * 2.
A diferença grande entre os limites, mostra que o algoritmo guloso, tem eficiência muito maior do que o algoritmo de força bruta.

1.5 Saída de dados e marcação de tempo

A marcação de tempo dos algoritmos é realizada através de NanoTime, porém, por motivo de formatação da saída, eles são enviados para saída como Milissegundos ou segundos, a depender do tempo de execução obtido.
A saída de dados escolhida foi a escrita em arquivos, a figura abaixo, representa o exemplo de como os dados de saída são escritos pelo algoritmo:

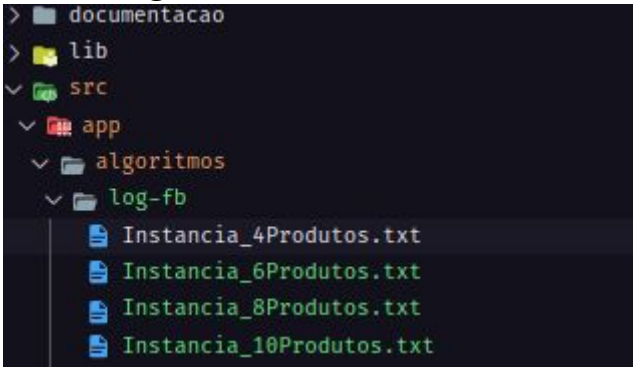
Figura 2 - Exemplo saída do algoritmo força bruta

```
- Carrinho de Compras -

Orçamento definido: R$ 1075.0
Peso maximo: 140 kg

Melhor lista de produtos encontrada:

Produtos:
Codigo do Produto: 0 | Peso: 27 kg | Valor: R$ 69.265724
Codigo do Produto: 6 | Peso: 33 kg | Valor: R$ 84.54614
Codigo do Produto: 18 | Peso: 14 kg | Valor: R$ 45.45964
Codigo do Produto: 24 | Peso: 44 kg | Valor: R$ 125.22525
Codigo do Produto: 30 | Peso: 9 kg | Valor: R$ 25.095139


Quantidade de itens da lista: 5
Valor de todos produtos: R$ 349.59192
Peso dos produtos: 127 kg

Numero total de possibilidades validas para o carrinho: 58
Numero total de possibilidades encontradas (válidas ou inválidas): 63

Tempo de execução: 2 Milisegundo(s)
```

Fonte: Gerado pelo algoritmo força bruta

Sendo assim, para cada tamanho de instância, será criado um arquivo texto, mostrando as informações conforme a figura acima, para o Algoritmo de Força Bruta, os dados serão salvos na pasta **'log-fb'** que está dentro da pasta **algoritmos**, a figura abaixo mostra o exemplo dos arquivos após a execução das instâncias definidas:

Figura 3 - Estrutura de saída



Fonte: VS Code do projeto

1.6 Ambiente de desenvolvimento e testes

O ambiente de desenvolvimento onde foi desenvolvido a solução, bem como os testes da solução, é o registrado abaixo:

Notebook: Modelo Lenovo Yoga 510.
Processador: Dual Core Intel® Core™ i5-6200U CPU @ 2.30GHz
Memória RAM: 4gb DDR4 2400Mhz
Sistema Operacional: Linux Elementary OS 5.1.7
Editor de código fonte: Visual Studio Code

2 FORÇA BRUTA

A seção apresenta os detalhes da implementação do algoritmo de força bruta para resolução do problema do supermercado.

2.1 Implementação

A implementação do algoritmo de força bruta, conforme definido, é baseado em instâncias, é construído em duas etapas conforme a teoria do algoritmo, a primeira etapa, trata-se da criação de todas as possibilidades possíveis baseados em uma lista de produtos, essas possibilidades são geradas sem nenhuma inteligência, de verificação se a lista de produtos cabe no orçamento e/ou no preço. A segunda etapa, trata-se da verificação se as listas geradas são válidas ou não são válidas, ou seja se cabem ou não no carrinho, bem como é encontrado, a melhor lista de produtos atual, baseado na instância, sendo a melhor, aquela que possui maior número de produtos, sem ultrapassar um peso e um orçamento.
A base da execução do algoritmo, são dois ArrayList, um definido como um ArrayList temporário, e outro como um ArrayList auxiliar, esses ArrayLists, são instâncias de listas de produtos, que irão cobrir todas as possibilidades possíveis de serem geradas.
O ArrayList temporário, terá em sua estrutura as listas anteriores, e o auxiliar, armazenará as combinações de lista baseado nas listas anteriores.

Veja um exemplo de funcionamento desta lógica:

Leia { } → como uma lista de produtos

Seja uma lista de produtos { 1 2 3 4 }.

1º Passo do Força Bruta: Gerar combinações isoladas para cada item
ArrayList Temporário terá → { 1 } , { 2 }, { 3 }, { 4 }

2º Passo do Força Bruta: Gerar combinações sequentes baseados nas isoladas:

ArrayList auxiliar terá → { 1,2 }, { 1, 3 }, { 1, 4}, { 2, 3} { 2, 4 }

Quando for gerado todo mundo com 2 elementos, no caso do exemplo, acima, o algoritmo irá chamar um método que verifica cada possibilidade do ArrayList temporário, validando se cada possibilidade é válida ou não, e se a possibilidade é a melhor, armazenando a mesma no carrinho de compras.
Ao fim da verificação, o auxiliar se tornará o novo temporário:

logo → Temporário = New ArrayList <ListaProdutos>(aux)
e aux = new ArrayList<ListaProdutos>( )

Sendo assim, Temporário contém → { 1,2 }, { 1, 3 }, { 1, 4}, { 2, 3} { 2, 4 }
E será gerado novas possibilidades baseado no temporario.

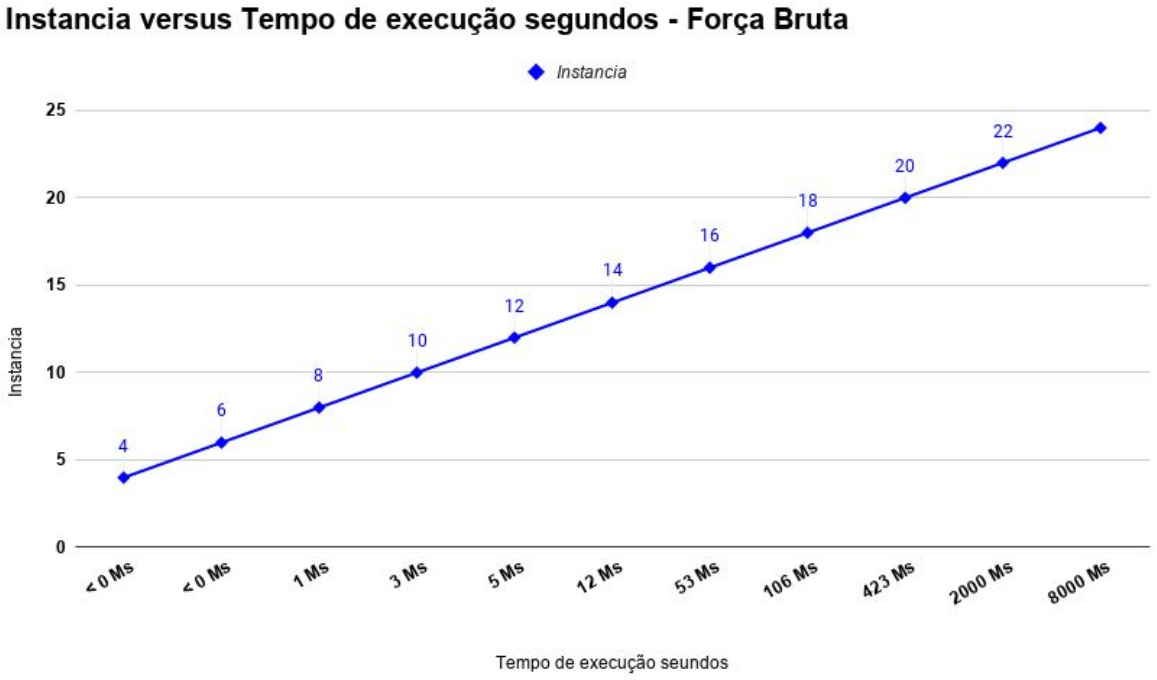
Sendo assim, Auxiliar conterá → {1, 2 3}, {1,2 ,4}, {1,3,4}, {2, 3 4}

E será feito o processo mencionado anteriormente, até que o algoritmo tenha o trabalho de gerar todas as possibilidades possíveis, e verificar cada uma delas.

2.2 Testes e resultados

Conforme mencionado, a instância para o força bruta, será de 4 a 24 produtos, aumentando de 2 em 2, sendo assim, foi possível mensurar, analisar e apresentar os dados em um gráfico, estudando o tempo de execução para cada número de instâncias, sendo possível realizar comparações entre as instâncias, e outras técnicas, como no caso do algoritmo guloso. O gráfico abaixo, mostra os dados de tempo e instância de acordo com os testes realizados;

Gráfico 1 - Tamanho da instância sobre tempo de execução do força bruta



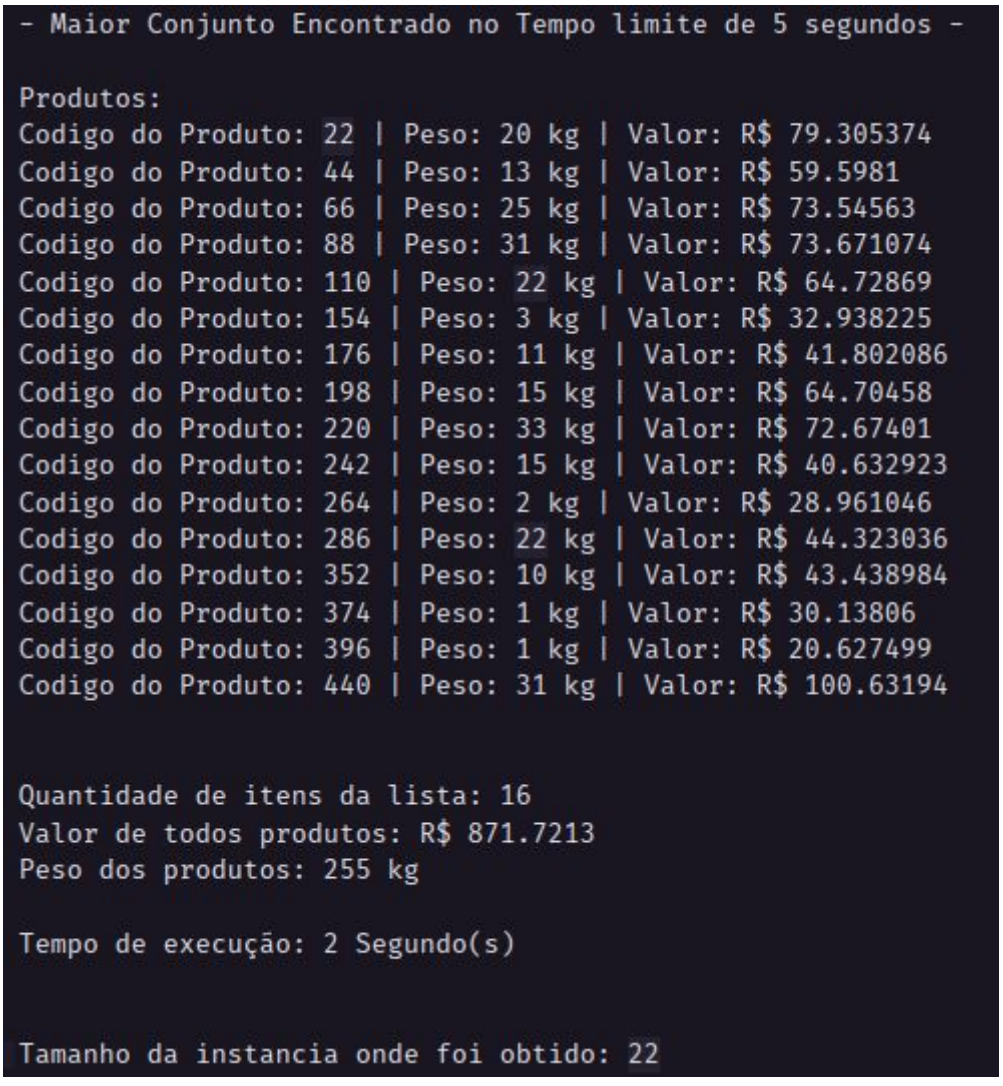
Fonte: Gerado com os dados de execução

Com o gráfico, é possível constatar que o tempo de execução cresce à medida que o número de instâncias aumenta, o crescimento do tempo acontece de forma exponencial, pode ser constatado no ponto da instância 20, onde o tempo é de 423 ms, já adiante no ponto da instância 22, o tempo salta para 2000 ms, e adiante a 8000 ms.

2.2.1 Maior conjunto obtido no tempo limite de 5 segundos

Conforme os testes, o maior conjunto obtido no tempo limite de 5 segundos, foi obtido no ponto da instância 22, com o tempo aproximado de 2 segundos, veja a saída abaixo mostrando um exemplo do maior conjunto obtido, no ambiente de testes.

Figura 4 - Maior conjunto



Fonte: Gerado na execução

Um detalhe que pode ser constatado na busca pelo maior conjunto possível, é que ele foi obtido no tempo de 2 Segundos, relativamente longe do tempo de 5 segundos, porém isso ocorre devido ao fato de que o algoritmo aumenta o tempo de execução de maneira exponencial, sendo assim, na próxima instância, de 24 produtos, o tempo de execução no ambiente de testes, passa a ser cerca de 9 segundos, passando do tempo limite definido de 5 segundos.

2.3 Considerações sobre a técnica aplicada ao algoritmo

Considerando que o estudado em sala de aula, no problema da mochila o número de subconjuntos de um conjunto de n é 2^n, conforme o supermercado é uma adaptação da mochila, o algoritmo é O(2n). Desta forma, podemos afirmar com toda certeza, que a solução é ineficiente para a resolução do problema da mochila, sendo um algoritmo muito limitado ao tamanho da instância.

3 ALGORITMO GULOSO

Conforme constatado no algoritmo de força bruta, para o problema do supermercado, a força bruta se mostra ineficiente na solução.
O algoritmo guloso segue a heurística de resolver o problema fazendo a melhor escolha localmente a cada iteração, com isso em mente, o algoritmo é usado na resolução do problema do supermercado, e nesta seção, será mostrado os detalhes da implementação, testes e resultados, e considerações sobre a técnica.

3.1 Implementação

A implementação do algoritmo guloso no problema, se mostrou em um código bem menor em comparação ao algoritmo de força bruta, basicamente, a solução do algoritmo é realizada em apenas 3 linhas de código, o restante do código trata-se de escrita da saída dos dados da resolução.
Para se obter a solução correta do problema em apenas 3 linhas de código, é necessário fazer com que o problema tenha uma subestrutura ótima, bem como é necessário que o algoritmo siga regras definidas, um critério guloso, e uma escolha gulosa, bem como possua um escopo de componentes definido, esses detalhes serão apresentados nas subseções a seguir, mas basicamente a solução do algoritmo é baseada na seguinte lógica:

1º Definição do critério guloso e escolha gulosa → Necessário para definição das regras e componentes;

2º - Ordenação da lista de produtos → Necessário para que o algoritmo chegue na solução, a ordenação deverá obter uma lista de produtos do menor para o maior, a lista de produtos ordenada representa o componente de candidatos a partir do qual uma solução será criada;

3º - Função de seleção → Escolher o melhor candidato a ser adicionado à solução → É uma escolha local realizada a cada iteração;

4º - Função de viabilidade → Usada para determinar se o candidato é ideal para a solução, no caso a função de viabilidade é se o produto cabe no carrinho (considerando peso e valor);

5º - Função de objetivo → Usada para atribuir um valor a uma solução, no caso a função de objetivo é adicionar o produto ao carrinho;

6º - Função de solução → A função de solução representa a condição de parada do algoritmo, ou seja, se já encontrou a solução esperada, não há mais trabalho a ser feito, no caso do problema, a função de seleção trata-se apenas de quando não cabe mais ninguém no carrinho.

3.1.1 Critério Guloso

O critério guloso definido para o problema do supermercado, é o critério de maximização, a justificativa é devido ao fato de que queremos a lista que contém a maior quantidade de produtos possível, baseado em uma lista de produtos do supermercado, de acordo com um orçamento e peso máximos definido.
Sendo assim, a escolha gulosa para o problema, é exatamente a escolha do menor produto, possibilitando assim uma lista com a quantidade maior de produtos, o menor produto é encontrado baseado no peso e valor do mesmo.

3.1.2 Componentes

Para a solução do problema, basicamente 5 componentes que são encontrados na maioria das soluções por algoritmo guloso, são também usados na resolução do problema, são eles:

Conjunto de candidatos: O conjunto de candidatos é a lista de produtos do supermercado;

Função de seleção: A função de seleção é cada seleção que é realizada na iteração do algoritmo, é uma decisão localmente ótima, no algoritmo, é realizado pela estrutura de repetição *while;*

Função de viabilidade: A função é a verificação se o produto escolhido na seleção cabe no carrinho ou não;

Função de objetivo: A função é a adição de um produto ao carrinho de compras.

Função de solução: É a verificação se foi descoberto a solução completa, a função de solução será verdadeira quando não couber mais nenhum produto no carrinho, com a realização da ordenação, quando o algoritmo encontrar o primeiro produto que não cabe, ele saberá que acabou, pois os restantes também não vão caber.

3.2 Testes e resultados

Os testes do algoritmo, foram realizados baseados em instâncias crescentes, a instância é iniciada com 4 produtos, com crescimento de n * 2, até que a instância alcance o limite de 1.048.576 (um milhão e quarenta e oito mil e quinhentos e setenta e seis) produtos, o crescimento de instâncias, é possível de ser realizada de maneira brutal em comparação ao algoritmo de força bruta, pois o limite de instâncias para o força bruta era de apenas 24 produtos.
Em média no ambiente de teste, o algoritmo é executado em menos de 1 milissegundos, número que aumenta muito pouco em relação ao tamanho da instância, por esse motivo, a criação do gráfico que represente não se mostra tão eficaz, porém a tabela abaixo, apresenta algumas medições de tempo realizadas, como efeito de comparação;

Tabela 1 - Crescimento da instância algoritmo guloso.

Tamanho da Instância	Tempo de execução
32.768 Produtos	19 ms
65.536 Produtos	48 ms
131.072 Produtos	73 ms
262.144 Produtos	95 ms
524.288 Produtos	298 ms
1.048.576 Produtos	464 ms

Fonte: Dados da execução

Com a tabela, é possível constatar que o algoritmo guloso, é brutalmente mais rápido do que o algoritmo de força bruta, observe que no algoritmo de força bruta, com apenas 24 produtos, o algoritmo levou cerca de 8 segundos para execução, já no algoritmo guloso, com 1.048.576 produtos, o algoritmo levou cerca de 464 milissegundos para encontrar a melhor lista de produtos, não é 1 segundo completo.

3.3 Considerações sobre a técnica aplicada ao algoritmo

Com os resultados da técnica de algoritmo guloso aplicada ao problema do supermercado, pode-se considerar que o algoritmo representa a melhor solução para resolver o problema diante as demais técnicas, comparado ao força bruta, a execução do algoritmo guloso é brutalmente mais rápida, possibilitando trabalhar com uma lista de produtos muito grande, sendo que o limite definido neste objeto de estudo foi um número bem alto, comparado ao limite definido no algoritmo de força bruta.

A solução desenvolvida, pode então ser considerada como uma solução gulosa legítima por possuir os componentes necessários citados no nas seções anteriores.

REFERÊNCIAS

GREEDY Algorithms. [S. l.], 2020. Disponível em: https://www.geeksforgeeks.org/greedy-algorithms/. Acesso em: 30 nov. 2020.

JAVA error: Comparison method violates its general contract. [*S. l.*], 11 jul. 2012. Disponível em: https://stackoverflow.com/questions/11441666/java-error-comparison-method-violates-its-general-contract. **Acesso em:** 30 nov. 2020.