MATA52 - Exercícios da Semana 06

- Grupo: Paládio
- Autores:
  - Elis Marcela de Souza Alcantara (Responsável)
    - Resolvi a primeira questão.
  - Bruno de Lucas Santos Barbosa
    - Resolvi a quarta questão, apesar de reconhecer que não consegui provar de fato seguindo um modelo matemático, tentei provar através de uma forma mais conceitual como eu enxergo a propriedade gulosa no algoritmo da mochila.
  - Lucas dos Santos Lima
    - Resolvi a segunda questão.
  - Monique Santos da Silva
    - Resolvi a terceira questão, pesquisei e estudei sobre algoritmo guloso, como argumentar que um algoritmo guloso funciona, teste de eficiência e como encontrar um contraexemplo que mostre que nem sempre o algoritmo irá retornar uma solução ótima.

Instruções (não apagar)

1. **Responsável**: Após copiar este notebook, altere o nome do notebook/documentação incluindo o nome do seu grupo. Por exemplo, se você é do grupo Ouro, altere o nome do notebook para "MATA53-Semana02-Ouro.ipynb"
2. **Responsável**: Compartilhe este documento com todos os membros do grupo (para participarem da elaboração deste documento). É importante que o notebook utilizado seja o mesmo compartilhado para que os registros de participação e colaboração fiquem salvos no histórico. Sugira uma divisão justa e defina um prazo aceitável para a inserção das soluções no Colab.
3. **Responsável**: Ao concluir a atividade, compartilhe o notebook com januario.ufba@gmail.com (dando permisão para edição) e deixando o aviso de notificação marcado, para que eu receba o seu e-mail. Identificar o nome do grupo na mensagem de compartilhamento.
4. **Cada membro**: Incluia o *seu próprio nome completo* na lista de autores que auxiliaram na elaboração deste notebook. Relate brevemente a sua contribuição na solução desta lista. O responsável aparece como sendo o(a) primeiro(a) autor(a).

5. **Cada membro**: Utilize os recursos de blocos de texto e de código para adicionar as suas respostas, sem alterar os blocos de texto e código existente. Não economize, esses blocos são de graça.

Exercícios

Considere o problema do troco para n centavos utilizando o menor número possível de moedas. Assuma que cada valor de moeda é um número inteiro.

1. Apresente um algoritmo que retorne o troco usando apenas moedas de 1, 5, 10, 25, e 50. Prove que o seu algoritmo obtém a solução ótima.

O algoritmo guloso desenvolvido tem como princípio ordenar o conjunto de moedas de um modo decrescente e não exceder o valor do troco a ser pago. Como o conjunto de moedas apresentado é de $[1, 5, 10, 25, 50]$, ele sempre vai retornar uma solução ótima por possuir as moedas $[1, 5, 10, 25]$. Um caso em que o algoritmo demonstra a apresentação de uma solução ótima é no caso do valor 30, tem diversas formas de retornar o resultado desse valor, uma das soluções não ótimas seria devolver três moedas de 10. O algoritmo apresentado retorna como melhor solução uma moeda de 25 e uma de 5.

```
MOEDAS = [1, 5, 10, 25, 50] # Conjunto de moedas
MOEDAS.sort(reverse=True) # Ordenando de forma decrescente


def calcularTroco(valor): # Parâmetro: valor para troco
    solucao = [] # Conjunto de solução

    for i in range(len(MOEDAS)): # Loop com parada no fim do conjunto de moedas
        solucao.append("Moeda: " + str(MOEDAS[i]) + " Vezes utilizada: " + str(valor
        valor %= MOEDAS[i] # Regulando o valor para troco

    return solucao # Retorna a solução
```

```
quantidadeCasosTeste = int(input())

for i in range(quantidadeCasosTeste):
    valorTroco = int(input())

    print("Valor para troco: " + str(valorTroco))
    print(calcularTroco(valorTroco))
```

```
5
30
Valor para troco: 30
['Moeda: 50 Vezes utilizada: 0', 'Moeda: 25 Vezes utilizada: 1', 'Moeda: 10 Veze
51
Valor para troco: 51
['Moeda: 50 Vezes utilizada: 1', 'Moeda: 25 Vezes utilizada: 0', 'Moeda: 10 Veze
33
Valor para troco: 33
['Moeda: 50 Vezes utilizada: 0', 'Moeda: 25 Vezes utilizada: 1', 'Moeda: 10 Veze
6
Valor para troco: 6
['Moeda: 50 Vezes utilizada: 0', 'Moeda: 25 Vezes utilizada: 0', 'Moeda: 10 Veze
25
Valor para troco: 25
['Moeda: 50 Vezes utilizada: 0', 'Moeda: 25 Vezes utilizada: 1', 'Moeda: 10 Veze
```

2. Suponha que as moedas disponíveis possuem valores expressos como potências de c, ou seja, os valores são $c^0, c^1, c^2, \ldots, c^k$, para algum inteiro $c > 1$ e $k \geq 1$. Prove que um algoritmo guloso sempre encontra a solução ótima.

Pensando nos valores possíveis que encontraremos para essas moedas, percebemos que é um número capaz de somarmos e resultar em qualquer número proposto na entrada, visto que sempre teremos o valor 1 em c^0, dessa forma podemos atingir qualquer valor ímpar proposto, e após isso temos potências de c, logo poderemos encontrar qualquer número proposto ao fazer essa soma. Utilizando os seguintes exemplos com o algoritmo na questão 1, com alterações que contemplem o cenário da questão 2.

```
def calcularTroco(valor, MOEDAS): # Parâmetro: valor para troco
    MOEDAS.sort(reverse=True)
    solucao = [] # Conjunto de solução
```

```
    for i in range(len(MOEDAS)): # Loop com parada no fim do conjunto de moedas
        solucao.append("Moeda: " + str(MOEDAS[i]) + " Vezes utilizada: " + str(valor
        valor %= MOEDAS[i] # Regulando o valor para troco

    return solucao # Retorna a solução


c_moedas = int(input('Digite a variável c: '))
k_power = int(input('Digite a variável k: '))
quantidadeCasosTeste = int(input('Quantidade de testes: '))
MOEDAS = []
for i in range(0, k_power): MOEDAS.append(c_moedas**i)
print(MOEDAS)
for i in range(quantidadeCasosTeste):
    valorTroco = int(input(f'Troco {i+1}: '))

    print("Valor para troco: " + str(valorTroco))
    print(calcularTroco(valorTroco, MOEDAS))
```

```
Digite a variável c: 3
Digite a variável k: 5
Quantidade de testes: 3
[1, 3, 9, 27, 81]
Troco 0: 54
Valor para troco: 54
['Moeda: 81 Vezes utilizada: 0', 'Moeda: 27 Vezes utilizada: 2', 'Moeda: 9 Vezes
Troco 1: 791
Valor para troco: 791
['Moeda: 81 Vezes utilizada: 9', 'Moeda: 27 Vezes utilizada: 2', 'Moeda: 9 Vezes
Troco 2: 9854
Valor para troco: 9854
['Moeda: 81 Vezes utilizada: 121', 'Moeda: 27 Vezes utilizada: 1', 'Moeda: 9 Vez
```

3. Encontre um conjunto de valores para as suas moedas de modo que o algoritmo guloso usado até o momento não seja capaz de obter a solução ótima, caso contrário, prove que o seu algoritmo funciona para quaisquer valores de moedas.

Algoritmos gulosos geralmente envolvem uma sequência de escolhas, sempre fazendo a escolha que parece melhor no momento. Contudo, eles não agem como os algoritmos de backtracking, uma vez que fazem uma escolha, eles não retrocedem para desfazer dessa escolha por uma melhor que resultaria em menos passos para uma solução ótima, por isso é fundamental que eles nunca façam uma escolha ruim.

Na primeira questão, consideramos um conjunto de moedas $[1, 5, 10, 25, 50]$, cada moeda pode ser usada quantas vezes forem necessárias até atingir o valor do troco.

O algoritmo da questão seleciona a maior moeda possível até que a soma do troco tenha sido zerada. O conjunto escolhido são conhecidos da moeda brasileira, então podemos garantir que o algoritmo guloso sempre resultará na solução ótima com o menor número possível de moedas.

No entento, usando um conjunto de moedas qualquer, o algoritmo guloso não produz necessariamente uma solução ótima.

Exemplo: Usaremos o conjunto $[1, 4, 9, 13, 15]$ e escolheremos como alvo obter o troco de 18 e

```
MOEDAS = [1, 4, 9, 13, 15] # Conjunto de moedas
MOEDAS.sort(reverse=True) # Ordenando de forma decrescente


def calcularTroco(valor): # Parâmetro: valor para troco
    solucao = [] # Conjunto de solução

    for i in range(len(MOEDAS)): # Loop com parada no fim do conjunto de moedas
        solucao.append("Moeda: " + str(MOEDAS[i]) + " Vezes utilizada: " + str(valor
        valor %= MOEDAS[i] # Regulando o valor para troco

    return solucao # Retorna a solução


quantidadeCasosTeste = int(input())

for i in range(quantidadeCasosTeste):
    valorTroco = int(input())

    print("Valor para troco: " + str(valorTroco))
    print(calcularTroco(valorTroco))
```

```
2
18
Valor para troco: 18
['Moeda: 15 Vezes utilizada: 1', 'Moeda: 13 Vezes utilizada: 0', 'Moeda: 9 Vezes
26
Valor para troco: 26
['Moeda: 15 Vezes utilizada: 1', 'Moeda: 13 Vezes utilizada: 0', 'Moeda: 9 Vezes
```

Como pode ser observado no código, no primeiro output para o valor 18, o algoritmo guloso retorna a solução 15 + 1 + 1 + 1 enquanto que a solução ótima é 9 + 9. Já no segundo output para o valor 26, o algoritmo guloso retorna a solução 15 + 9 + 1 + 1 enquanto que a solução ótima é 13 + 13.

Conclusão: O algoritmo guloso é efetivo em retornar a solução ótima para moedas que são padronizadas, como as nacionais, porém não é efetivo quando se usa valores um conjunto de

moedas incomuns.

4. Prove que o problema da mochila fracionária tem a propriedade da escolha gulosa.

R: É possível enxergar a propriedade gulosa no algoritmo pois dada uma sequência de W números crescente, o algoritmo pode selecionar um maior valor em ordem descrescente, de tal forma que de forma "gulosa" capta os maiores valores.

Pensando numa sequência de itens crescente de números, o algoritmo priorizaria os itens de maior valor, pois agregariam maior valor a mochila. E pensando numa possibilidade que essa sequência W tenha valores iguais, o algoritmo priorizaria outra prioridade dos itens, que são os mais leves.