Trabalho Prático 1

- **Como esse problema pode ser modelado para o paradigma guloso?**

O Problema foi modelado de acordo com o custo benefício de cada prato seguindo a ordem de um vetor construído relevando a ordem de melhor custo benefício e a regra de cozinhar o mesmo prato em dias seguidos.

Para isso, foi criado um vetor com os pratos na seguinte ordem: Os pratos de melhor custo benefício levando em conta os que foram cozinhados somente em um dia e os que foram cozinhados em dois dias seguidos. Logo após, concatenou-se a esse vetor os pratos com o menor custo e que não dão lucro algum, pois serão cozinhados em três dias seguidos ou mais.

Junto a isso, para resolver o problema, foi implementado a seguinte lógica de estrutura gulosa: Para cada dia em que o cozinheiro deverá cozinhar, coloca o prato da posição j do vetor criado numa variável prato, onde na primeira iteração j=0. Enquanto não achar um prato no vetor que caiba no orçamento, deve-se testar os pratos do vetor de pratos e adicionar um a j e a k. Caso não encontre um prato que caiba no orçamento do vetor, o problema não haverá solução, caso encontre coloque na variável prato o prato de índice j. É subtraído do valor do orçamento o valor do prato e o prato adicionado ao vetor resposta. Para finalizar, verifica se há um prato no dia anterior, se houver e o prato do dia atual é igual o do dia anterior, coloca a variável anterior com o valor de j e soma o valor de j. Caso o prato do dia atual não seja igual ao do dia anterior, coloca a variável j com o valor de k. Caso ainda haja dias para cozinhar, volta no loop com o dia atualizado.

A solução dos pratos que serão cozinhados será dada pelo vetor resposta, onde terá a ordem dos pratos cozinhados em todos os dias pelo cozinheiro.

- **Seu algoritmo guloso dá a solução ótima? Por quê?**

Não, pois a solução gulosa nesse problema pega o melhor valor no momento atual, ou seja, o cozinheiro irá cozinhar o prato que lhe dará o melhor lucro no dia em questão sem levar em conta todos os dias que ele tem disponíveis ou o orçamento total que ele possui. Dessa forma, o algoritmo guloso não garante a solução ótima para o problema como um todo, e sim para o problema nos dias em questão.

- **Como esse problema pode ser modelado para o paradigma de programação dinâmica?**

O problema foi modelado atráves de três tabelas para a consulta dinâmica do cáculo do melhor custo benefício dos pratos a serem cozinhados. Uma matriz, representando os pratos a serem escolhidos sem que já tenham sido escolhidos no dia anterior, outra representando os pratos a serem escolhidos pelo primeiro dia consecutivo e uma última representando os pratos a serem escolhidos pelo segundo ou mais dias consecutivos. Cada uma dessas matrizes é da terceira dimensão e elas são: O número de dias, o número de pratos e , por fim, o orçamento total.

Com as matrizes devidamente inicializadas, utilizasse a programação dinâmica buscando maximizar o lucro do restaurante. Para isso, percorre a matriz principal, buscando recorrer quando necessário as outras duas matrizes para obter o máximo lucro levando em conta possíveis repetições de pratos.

A solução dos pratos que serão cozinhados será dada pelo elemento da matriz que tiver o maior lucro no último dia e na posição orçamento.

- **Discuta a sub-estrutura ótima e a sobreposição dos problemas.**

A subestrutura ótima para o problema utilizando o algoritmo guloso, consiste em criar um vetor com o custo benefício dos pratos e, após isso, escolher os pratos que seu custo cabem no orçamento e e detém o melhor custo benefício entre todos.

Já para a subestrutura ótima utilizando a programação dinâmica é necessário construir três matrizes da terceira dimensão, onde cada uma das matrizes representam não repetir o prato, repetir o prato uma vez e repetir o prato duas vezes ou mais. Suas dimensões são os dias e os pratos a cozinhar e o orcamento total. Dessa forma, utilizasse a primeira matriz para pegar o prato melhor prato com o melhor custo benefício para o dia em questão e em caso de repetição de pratos utilizar as outras duas matrizes como auxílio.

Em relação a sobreposição de problemas, o algoritmo guloso precisa somente saber qual foi o último prato cozinhado e se foi cozinhado mais de uma vez para analisar a melhor opção atual que ele tem. Por outro lado, utilizando a programação dinâmica é necessário olhar a melhor opção do dia anterior e olhar nas matrizes auxiliáres para saber se o prato já fora cozinhado.

- **Se algum algoritmo clássico foi adaptado para resolver o problema, qual foi ele?**

O Algoritmo clássico adaptado para resolver o problema do Trabalho Prático, foi o algoritmo da mochila binária, onde o objetivo é maximizar o valor total dos itens que podem ser carregados na mochila. Dessa forma, foi possível adaptś-lo para um problema em que deve-se maximizar o lucro ao longo dos dias levando em consideração as regras impostas pelo problema.

- **Qual o custo assintótico (em Big-O) do tempo de execução e espaço em memória utilizado? Não se esqueça de formular a equação de recorrência da abordagem baseada em programação dinâmica (não é necessário resolver a equação explicitamente).**

GULOSO:

- O(n): Pois é preciso somente navegar no vetor de custo benefício para achar o próximo prato a ser cozinhado.

- Uso de memória: 13296 bytes em média

DINÂMICO:

- $O(N^3)$: Pois é preciso olhar os valores em três matrizes de três dimensões.

- Uso de memória: 13316 bytes em média

- T(n) = Não foi utilizado recorrência para resolver o problema.

- **Em qual máquina o tp foi testado?**

MÁQUINA:

descrição: CPU
produto: Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
fabricante: Intel Corp.
informações do barramento: cpu@0
versão: Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
slot: SOCKET 0
tamanho: 2461MHz

capacidade: 3GHz
largura: 64 bits
clock: 100MHz

descrição: Memória do sistema
slot: Placa do sistema ou placa-mãe
tamanho: 6GiB

- **Como sua implementação se comporta em termos de tempo de execução e memória alocada? Apresente uma variação experimental da solução, com entradas de tamanho diferente tentando, pelo menos, discutir também o melhor e o pior caso.**

GULOSO

➔ 5 pratos com orçamento de 20

- 1 dia
time: 0.07900
Uso Memória: 13296.00000 - Porção de memória Ocupada: 388

- 3 dias
time: 0.08400
Uso Memória: 13296.00000 - Porção de memória Ocupada: 381

- 7 dias
time: 0.11900
Uso Memória: 13296.00000 - Porção de memória Ocupada: 412

- 12 dias
time: 0.20500
Uso Memória: 13296.00000 - Porção de memória Ocupada: 386

- 21 dias
time: 0.20000
Uso Memória: 13296.00000 - Porção de memória Ocupada: 410

➔ 50 pratos com orçamento de 100
- 1 dia
time: 0.16700
Uso Memória: 13296.00000 - Porção de memória Ocupada: 445

- 3 dias
time: 0.17500
Uso Memória: 13296.00000 - Porção de memória Ocupada: 409

- 7 dias
time: 0.19000

Uso Memória: 13296.00000 - Porção de memória Ocupada: 410

- 12 dias
  time: 0.24200
  Uso Memória: 13296.00000 - Porção de memória Ocupada: 409

- 21 dias
  time: 0.34600
  Uso Memória: 13296.00000 - Porção de memória Ocupada: 395

DINÂMICO

➔ 5 pratos com orçamento de 20

- 1 dia
  time: 0.21300
  Uso Memória: 13316.00000 - Porção de memória Ocupada: 392

- 3 dias
  time: 0.37500
  Uso Memória: 13316.00000 - Porção de memória Ocupada: 395

- 7 dias
  time: 0.75400
  Uso Memória: 13316.00000 - Porção de memória Ocupada: 395

- 12 dias
  time: 2.31100
  Uso Memória: 13316.00000 - Porção de memória Ocupada: 395

- 21 dias
  time: 3.04700
  Uso Memória: 13316.00000 - Porção de memória Ocupada: 414

➔ 50 pratos com orçamento de 100

- 1 dia
  time: 16.21800
  Uso Memória: 13448.00000 - Porção de memória Ocupada: 779

- 3 dias
  time: 60.04900
  Uso Memória: 13712.00000 - Porção de memória Ocupada: 846

- 7 dias
  time: 126.92600
  Uso Memória: 14108.00000 - Porção de memória Ocupada: 897

- 12 dias
  time: 244.88900
  Uso Memória: 14636.00000 - Porção de memória Ocupada: 1064

- 21 dias
    time: 442.06900
    Uso Memória: 15692.00000 - Porção de memória Ocupada: 1319

O melhor caso do problema é quando a quantidade de dias é mínima, ou seja, há apenas um dia para cozinhar os pratos e por consequência em ambos os algoritmos muitas verificações do dia anterior não são feitas. Já o pior caso, no caso do guloso é quando há muitos dias para cozinhar e todos os pratos tem custo acima do orcamento menos os dois com pior custo benefício. No caso do dinâmico é quando há muitos dias para cozinhar e um prato é cozinhado várias vezes segidas, pois é preciso verificar em várias matrizes.

- **Bibligrafia?**

CORMEN, Thomas H. et al. Algoritmos: teoria e prática. Rio de Janeiro, RJ: Elsevier, Campus, c2012. xvi, 926 p. ISBN 9788535236996., Nº de Exemplares: 14.

MASSAND, Ashish. Solution for SPOJ Dynamic Programming. Disponível em: <https://www.quora.com/What-is-the-solution-for-the-MENU-problem-on-Sphere-Online-Judge-SPOJ/> Acesso em: 15 de maio de 2018