Prova 2

PAA - Semestre 2 - 2018

Data: 10/12/2018

1. (2,5) Para cada item, responda se a afirmação é verdadeira ou falsa e motive sua resposta.
    a. Uma solução gulosa é aquela que toma decisões locais para minimizar o valor da variável objetivo do problema. No entanto, nem sempre é garantida a resposta ótima com esse tipo de solução.

    F. "Minimizar" é a palavra errada. Um algoritmo guloso também pode maximizar a variável objetivo. Utilizar a palavra "otimizar" já resolveria o problema da afirmação. Vale metade se não pegar a palavra, mas falar que a segunda parte está correta.

    b. O problema da mochila binária pode ser considerado um problema pseudo-polinomial, uma vez que sua complexidade pode ser dominada por um valor que não está relacionado ao tamanho da entrada (número de itens).

    V. Esse valor é a capacidade da mochila, que pode ir a infinito.

    c. A vantagem dos algoritmos para encontrar caminhos mínimos em grafos Bellman-Ford e de Floyd-Warshall sobre o algoritmo de Djikstra são suas complexidades computacionais. Especificamente, eles obtêm o menor caminho entre todos os pares e, para executar o algoritmo de Djikstra tantas vezes quanto fosse necessário para obter o mesmo resultado, ele ficaria mais custoso.

    F. Não tem a ver com complexidade (o que está escrito sobre isso está errado), mas com a presença de arestas negativas. Djikstra (com PQ) fica mais eficiente, mesmo sendo rodado |V| vezes. Na verdade, o Bellman-Ford também pode ser usado para todos os pares, ficando mais custoso que o Djikstra de todo modo. Metade se falar da aresta, mas não acertar em relação à complexidade.

    d. Encontrar uma árvore geradora mínima é um problema de otimização, portanto pode ser resolvido por um algoritmo de programação dinâmica. Porém, há

algoritmos gulosos para esse problema, que são mais eficientes que a solução por programação dinâmica.

V. Até dar para fazer com PD, mas não vai ser melhor que Prim ou Kruskal (que o ser humano já saiba, pelo menos).

e. A ⩽ B significa que B pode ser reduzido para A. Ou seja, A não é mais difícil que B.

F. A pode ser para B..

2. (2,5) Adaptado de ENADE 2017 (BCC): Considere o problema do troco (dado um valor, minimizar o número de moedas que o contempla) em um cenário em que se há moedas de 1, 5, 10, 25 e 50 unidades monetárias. Para resolver esse problema, alguém implementou o seguinte código, que recebe o valor *v* a ser trocado e retorna um vetor contendo a quantidade de moedas para cada valor possível.

```
1.  public static int[] troco(int valor){
2.          int[] moedas = new int[5];
3.
4.          moedas[4] = valor / 50;
5.          valor = valor % 50;
6.          moedas[3] = valor / 25;
7.          valor = valor % 25;
8.          moedas[2] = valor / 10;
9.          valor = valor % 10;
10.         moedas[1] = valor / 5;
11.         valor = valor % 5;
12.         moedas[0] = valor;
13.
14.         return(moedas);
15. }
```

Considerando o cenário apresentado, esse código garante a resposta ótima? Discuta o motivo da sua resposta, ou seja, prove que a solução está correta ou errada.

Sim, para esse cenário é ótima. Dá para discutir a prova por absurdo. Mas eu espero mais uma discussão do que a prova em si. O caminho é pensar na proporção do valor de cada moeda com as moedas de valor menor. Resposta mais completa: http://people.cs.ksu.edu/~sathish/coinset.pdf

A solução gulosa para esse problema funciona para quaisquer valores de moedas (considerando que sempre haverá moedas de 1)? Explique ou mostre um exemplo.

Basta encontrar um contra-exemplo para mostrar que não é válida. Ex: moedas de 1, 3, 4 e 5 para troco de 7. Guloso daria (5,1,1) e PD daria (4,3).

3. (2,5) A maior subsequência comum (ou subsequência comum máxima) entre duas *strings* é um subconjunto das letras em comum entre as duas cadeias, respeitando sua ordem inicial. Por exemplo, (2,3,2,1) é uma subsequência comum máxima de (1,2,3,2,4,1,2) e (2,4,3,1,2,1). DESCREVA UM ALGORITMO...
   Solução bem descrita (na parte de PD, pois começa-se falando da solução ineficiente) em: http://edirlei.3dgb.com.br/aulas/paa/PAA_Aula_04_LCS_2015.pdf

4. (2,5) O problema da cobertura mínima de vértices consiste em achar o menor subconjunto S de vértices de um grafo G tal que todas as arestas de G sejam incidentes de, ao menos, um dos vértices de S. Esse é um problema NP-Difícil.
   a. (1,0) Mostre que esse problema é NP-Completo. Caso não consiga demonstrar alguma das etapas necessárias para isso, diga quais são as etapas e ao menos discuta um possível caminho para cada uma delas. (PARA A VERSÃO DO PROBLEMA DE DECISÃO)
      É necessário mostrar um algoritmo polinomial para verificar se os vértices cobrem todas as arestas (além de ver se o tamanho de S é menor que um dado k). Tranquilo, é só iterar marcando as arestas e ver se não sobrou nenhuma.
      A segunda parte seria a redução, que não espero que mostrem a redução, mas que mostrem que entendam e comentem sobre algum problema em NP que pode ser que dê uma caminho para a redução.
   b. (1,5) Proponha uma heurística para encontrar uma solução aproximada para o problema e demonstre com um exemplo ilustrativo.
      Aceito de várias maneiras. Dá para pensar em heurística pelo problema do grafo bipartido, por decisão gulosa pelo grau dos vértices, entre outras.