Lista de Exercícios de PAA - Módulo 3 - 2019.1
Profs. Vinícius Fernandes dos Santos e Sebastián Urrutia

1. Considere um vetor de $n \geq 2$ posições, cada uma preenchida com um dígito no intervalo 0-9. Queremos decompor este vetor em subvetores, ou seja, em trechos consecutivos. Além disso, todo subvetor deve ser par, isto é, o número formado pelos dígitos do subvetor deve ser par. Por exemplo, caso o vetor de entrada seja 805438452, este pode ser particionado como "8054, 38, 452", com 3 subvetores, ou como "8, 0, 54384, 52", com 4 subvetores, dentre outras possibilidades. Escreva um algoritmo que determine o maior número de subvetores em que o vetor pode ser decomposto. Determine a complexidade de seu algoritmo. Você pode assumir que o último dígito do vetor é par.

2. Dizemos que uma sequência é alternante se não existe três elementos consecutivos a_i, a_{i+1}, a_{i+2} tal que $a_i < a_{i+1} < a_{i+2}$ ou que $a_i > a_{i+1} > a_{i+2}$. Em outras palavras, para todo i, os sinais de $a_i - a_{i+1}$ e $a_{i+1} - a_{i+2}$ são distintos. Dado um vetor de número inteiros, você deve determinar o tamanho da maior subsequência alternante. Para o vetor $4, 3, 2, 5, 8, 7, 1, 10, 11$, uma possível subsequência alternante seria $4, 2, 8, 7, 11$. Forneça um algoritmo para resolver este problema e determine sua complexidade. Você pode assumir que todos os elementos do vetor de entrada são distintos.

3. Esta questão novamente trata de sequências alternantes. Desta vez, diferentemente da questão anterior, não estamos interessados em uma subsequência, mas sim em um subvetor. Ou seja, queremos um conjunto de posições consecutivas que seja alternante. Para o mesmo vetor de exemplo da questão anterior, $3, 2, 5$ e $7, 1, 10$ seriam subvetores alternantes. Estamos interessados em determinar o tamanho do maior subvetor alternante. Note que quaisquer dois elementos distintos consecutivos formam um subvetor alternante. Forneça um algoritmo para resolver este problema e determine sua complexidade. Você pode assumir que todos os elementos do vetor de entrada são distintos.

4. Dada uma string binária (ou seja, uma string de zeros e uns), descreva uma estratégia para determinar o tamanho da maior sequência de zeros consecutivos desta string. **Você deve utilizar um dos paradigmas** visto durante o curso para resolver este problema. Descreva qual paradigma foi utilizado e explique porque, de fato, seu algoritmo pode ser interpretado como um exemplo de uso de tal paradigma.

5. Considere um algoritmo de programação dinâmica que resolva o problema da mochila. Considere que há n elementos, a mochila tem capacidade C e dois vetores são fornecidos, com p_i e v_i correspondendo ao peso e ao valor do i-ésimo elemento, respectivamente.

   Considere que o resultado do algoritmo foi armazenado em uma tabela, de forma que `pd[i][j]` armazena o maior lucro que pode ser obtido podendo-se utilizar quaisquer elementos do conjunto $\{1, \ldots, i\}$ em uma mochila de capacidade j.

   Mostre como reconstruir a lista de objetos que deve ser selecionada em uma solução ótima, fornecendo um algoritmo para isso. Para isso, escolha uma das seguintes abordagens.

   (a) Descrever a solução apenas a partir da tabela `pd[][]`, supondo que ela já foi computada anteriormente. Ou seja, todas as decisões a serem tomadas se baseiam nos índices e nos valores armazenados na tabela. Se escolhida (e resolvida corretamente) esta abordagem pode lhe conferir pontos extras.

   (b) Descrever a solução completa, fornecendo o algoritmo para computar os valores de `pd[][]` e quaisquer outras estruturas auxiliares que julgar necessárias.

6. Um grande esquema de corrupção assola a Federação dos Planetas Unidos. Os órgãos competentes iniciaram uma investigação, para entender como a quadrilha opera. Disfarçados de pessoas bem intencionadas, esta quadrilha é estruturada de forma extremamente hierárquica, com um único líder. Com exceção do líder, cada membro da quadrilha possui um chefe imediato. Desta forma, a cadeia de comando é muito bem definida.

   As autoridades resolveram tomar uma providência, após o surgimento de algumas provas contra a quadrilha. Entretanto, as provas não são muito contundentes: o ideal seria a obtenção de "delações premiadas" para validarem as suspeitas e para possibilitar a obtenção de provas inquestionáveis. Como o "prêmio" de tais delações consiste de redução de pena, de certa forma, quanto mais delatores, menos criminosos serão punidos. Desta forma, deseja-se que o número de delações seja o menor possível. Por outro lado, para condenar todos os criminosos, para cada membro da quadrilha é necessário que ele ou pelo menos um de seus associados (ou seu chefe imediato, ou seus subordinados imediatos) faça uma delação.

   Forneça um algoritmo para determinar o menor número de delações necessárias para condenar toda a quadrilha. Você pode assumir que os membros da quadrilha estejam numerados de 1 a n, onde o número 1 corresponde ao líder, e que existe um vetor `chefe`, onde o valor `chefe[i]` corresponde ao chefe imediato de i. Além disso, você pode assumir que `chefe[1] = 0`. Analise a complexidade de seu algoritmo.

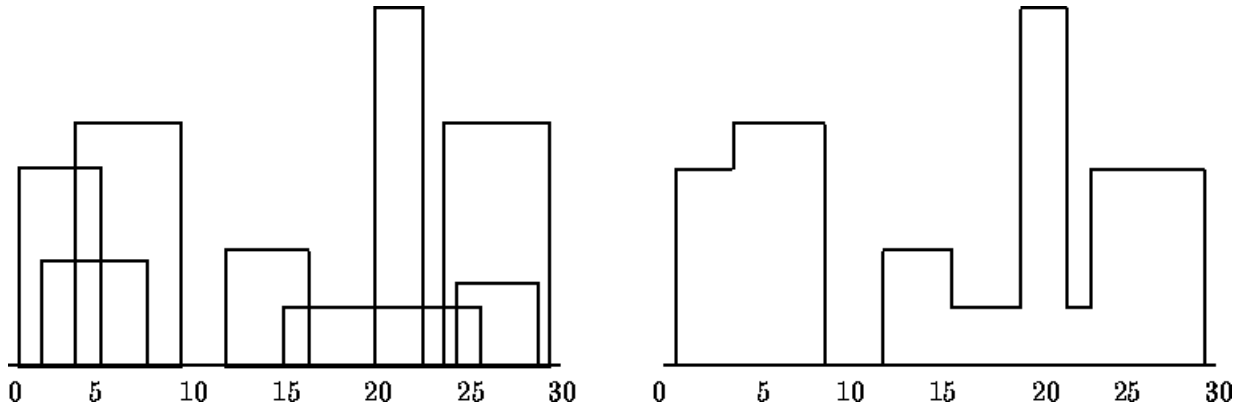

Figura 1: Ilustração do exemplo de entrada acima. Note, no lado direito, o *skyline* formado pelos retângulos fornecidos na entrada. Figura original do UVa Online Judge (http://uva.onlinejudge.org).

7. O problema do *skyline* consiste em determinar, a partir de uma lista de prédios, qual o visual formado quando olhamos de longe. Neste problema, vamos assumir que uma pessoa observa uma cidade de um ponto suficientemente distante, de forma que os prédios são vistos como retângulos em um plano. Tais prédios serão representados por triplas (a_i, b_i, c_i), correspondendo a um prédio de altura b_i, se estendendo da coordenada a_i até a coordenada c_i, no eixo x. A entrada do problema é uma sequência $(a_1, b_1, c_1), \ldots, (a_n, b_n, c_n)$. A saída deve ser uma sequência de números inteiros $(x_1, y_1, \ldots, x_n, y_n, x_{n+1})$, de forma que a altura do *skyline* entre os pontos x_i e x_{i+1} é y_i.

   Nesta questão você deve fornecer um algoritmo de tempo $\mathcal{O}(n \log n)$ para este problema. Você pode assumir que as triplas já estão armazenadas em alguma estrutura, ou que são passada como parâmetro. Não é necessária sua leitura.

   Dica: use divisão e conquista.

   Um exemplo de entrada seria o conjunto de triplas (1,11,5), (2,6,7), (3,13,9), (12,7,16), (14,3,25), (19,18,22), (23,13,29), (24,4,28). Para esta entrada, a saída esperada seria (1, 11, 3, 13, 9, 0, 12, 7, 16, 3, 19, 18, 22, 3, 23, 13, 29, 0). Note que para uma entrdaa com um único prédio, a saída corresponde exatamente à entrada.

8. A Nlogônia é uma próspera nação, que preza pela eficiência de seus processos. Para aumentar a eficiência de seu sistema monetário, seu sistema de cédulas está sendo modificado: agora n tipos de cédulas serão utilizados, de valores $c_1, \ldots, c_n$. Para garantir que todos os valores inteiros positivos podem ser formados, foi definido que $c_1 = 1$. Tal mudança implicará na reprogramação dos caixas eletrônicos do país que, prezando pela eficiência, sempre fornece o menor número de cédulas possível quando um nlogoniano saca uma quantia Q. Alguns testes preliminares estão sendo feitos com determinadas sequências $c_1, \ldots, c_n$ e valores Q. Nesta questão, vamos ajudar os nlogonianos com os testes para decidir o sistema ideal a ser utilizado.

   (a) Forneça um exemplo de sequência $c_1, \ldots, c_n$ e um valor Q tal que um algoritmo guloso que sempre tente utilizar o maior valor de cédula possível não produz a menor quantidade de cédulas possível. Explique qual seria a solução encontrada pelo algoritmo e qual seria a solução ótima.

   (b) Forneça uma relação de recorrêcia para determinar o valor $m(i, j)$ correspondendo ao menor número de cédulas para se fazer a quantia j, podendo-se utilizar os i primeiros tipos de cédulas. Não se esqueça dos casos base!

   (c) Escreva um algoritmo correspondente que determina o valor de $m(n, Q)$. Para que tal algoritmo seja eficiente, recomenda-se utilizar programação dinâmica. Analise sua complexidade de tempo. Para praticar, você pode fazer duas versões: uma top-down e outra bottom-up.

9. Dada uma sequência de números inteiros positivos $s = s_1, \ldots, s_n$, escreva um algoritmo de tempo linear que retorne o menor inteiro positivo k que não esteja em s. Pode-se interpretar o seu algoritmo como aplicação de algum dos paradigmas visto em sala? Justifique.