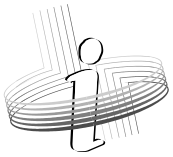

UNIVERSIDADE FEDERAL DE VIÇOSA
DEPARTAMENTO DE INFORMÁTICA
INF332 – Projeto e Análise de Algoritmos

Exercícios

1. Dada uma sequência de n números ordenados tais que a diferença entre os termos consecutivos seja constante. Baseado na técnica Divisão e Conquista, escreva um algoritmo de complexidade $O(\log_2 n)$ para encontrar o elemento que falta. Suponha que o primeiro e o último elemento sejam sempre parte da sequência de entrada e o elemento ausente estará entre as posições 1 a n-1.
Por exemplo:
Sequência de entrada: [1, 7, 10, 13, 16]
Saída: O elemento ausente é 4.

2. Dado um array ordenado de tamanho n contendo elementos repetidos. Aplicando Divisão e Conquista escreva um algoritmo para encontrar a frequência de cada elemento sem percorrer todo o array.
Por exemplo:
Para o array de entrada: A = [2, 2, 2, 2, 4, 4, 5, 5, 8, 8, 8, 9]
Temos as seguintes frequências como saída: f(2) = 4, f(4) = 2, f(5) = 2, f(8) = 3 e f(9) = 1.

3. Dada uma string S = [c1, c2, ...,cn] de n caracteres. Alguns caracteres devem ser removidos para que a string se converta em um palíndromo. Encontre o número mínimo de remoções para convertê-la em um palíndromo.
Exemplo: S = [A C R A D R A M]
A C **R A** D **R A** M : Removendo C D e M obtemos [A R A R A] que é palíndromo.
A C **R** A **D R A** M : Ou removendo C A e M obtemos [A R D R A] que é palíndromo.
Escreva um algoritmo baseado em Programação Dinâmica para resolver o problema.

4. Considere um conjunto de n atividades a serem realizadas. Para cada atividade i tem-se, um prazo para ser realizado (*deadline*) $d_i \geq 0$, e um lucro $p_i > 0$. O lucro pela realização de uma atividade será obtido somente se a atividade é concluída dentro do seu prazo. Suponha que, para realizar qualquer atividade gasta-se o mesmo tempo (uma unidade de tempo), e somente uma atividade deve ser realizado por vez. O problema consiste em selecionar um subconjunto de atividades que possam ser realizadas dentro dos seus prazos e que o lucro total (soma dos lucros das atividades selecionadas) seja o máximo. Também deve ser determinada a ordem de execução das atividades selecionadas. Uma solução será inviável se uma atividade escolhida finaliza após seu prazo. A execução das atividades deve ser inicializada num tempo zero (ou seja, a partir desse tempo inicial zero serão contabilizados os prazos para a execução das atividades).
Exemplo: n = 6 atividades:

Atividades	**1**	**2**	**3**	**4**	**5**	**6**
Prazos (d_i)	2	3	1	2	2	4
Lucros (p_i)	100	40	80	20	60	10

Uma solução viável é escolher as atividades 1, 5, 2 e 6 (executar nesta ordem). Suponha que a unidade de tempo seja hora. Note que a atividade 1 gastará 1h para ser executada, e como tem um prazo de 2, não estará atrasada. A atividade 5 finalizará uma hora depois da atividade 1, dentro do seu prazo (seu prazo é 2). A atividade 2 finalizará após 3h (dentro do seu prazo que é 3). A atividade

6 finalizará após 4h, dentro do seu prazo (seu prazo é 4). Como todas as atividades finalizam dentro dos seus respectivos prazos, o lucro total será: $p_1 + p_5 + p_2 + p_6 = 100 + 60 + 40 + 10 = 210$. É a solução ótima?

a) Escreva uma estratégia gulosa para resolver o problema (para selecionar as atividades, determinar a ordem de execução e maximizar o lucro total).
b) Aplique a estratégia gulosa para resolver o exemplo numérico mostrado acima. A estratégia sempre determinará a solução ótima?
c) Escreva o pseudocódigo do algoritmo guloso e calcule sua complexidade.

5. Considere uma matriz de tamanho com *n*x*n* números. Deseja-se escolher *n* números que estejam em linhas e colunas diferentes, cuja soma seja o máximo possível. Por exemplo, na tabela abaixo podem ser escolhidos 180 + 110 + 140 + 30 + 10 = 470 (é o valor máximo possível?).

	1	2	3	4	5
1	50	40	100	60	**180**
2	20	30	**110**	95	65
3	**140**	120	25	35	90
4	160	10	200	**30**	85
5	45	**10**	55	60	170

a) Escreva uma estratégia gulosa para resolver o problema.
b) Aplique sua estratégia para resolver a instância da tabela acima. A estratégia sempre determinará a solução ótima? Caso não, apresente um contraexemplo.

6. Seja uma sequência de *n* números inteiros S[1..n] = $\{s_1, s_2, ..., s_n\}$. Deseja-se determinar uma subsequência $\{s_i...s_j\}$ de S (não necessariamente consecutiva) formada pelo maior número de elementos ordenados de forma estritamente crescente, ou seja, $s_i < ... < s_j$.

Por exemplo, para S = {5, 2, 8, 6, 3, 6, 9, 4}, algumas subsequências crescentes são: {5, 6}, {5, 8, 9}, {2, 8, 9}. Não são consideradas subsequências: {3, 5} e {3, 6, 8}. A melhor solução (solução ótima) é a subsequência {2, 3, 6, 9}, formada por 4 elementos. Resolver o problema utilizando PD, da seguinte maneira:

a) Defina a função de recorrência F[j] para calcular o número de elementos de uma subsequência crescente de S[1..j]. O maior valor contido em F deve ser o valor da solução ótima.
b) Aplique a relação de recorrência, de forma *bottom-up*, para encontrar a maior subsequência crescente de S = {50, 20, 80, 60, 30, 60, 90, 40}. Mostre, passo a passo, todos os cálculos realizados para a construção da tabela.
c) Escreva um algoritmo de PD que recebe a sequência S[1..n], e monta a tabela F[] utilizando a função de recorrência, e retorna o tamanho da maior subsequência crescente contida em S[1..n]. Calcule a complexidade do algoritmo.
d) Escreva um algoritmo para imprimir os elementos da maior subsequência crescente contida em S[1..n]. O algoritmo deve usar a tabela F[].