Instituto Federal de Educação, Ciência e Tecnologia de Brasília – Câmpus Taguatinga
Ciência da Computação – Análise de Algoritmos
Lista de Exercícios – Algoritmos Gulosos
Prof. Daniel Saad Nogueira Nunes

Aluno: _____
Matrícula: _____

Exercício 1

Implemente na sua linguagem de programação favorita o algoritmo para obter o conjunto maximal de eventos compatíveis.

Exercício 2

(Problema da mochila fracionária)

João Cláudio Andaime depois de "bater umas lajes" encarou um restaurante do estilo siga bem caminhoneiro, onde pode-se comer (quase) à vontade. Como bom pedreiro, João está com uma baita fome. Neste restaurante é possível colocar no máximo $k \in \mathbb{R}^*$ gramas de alimentos em um prato. O restaurante possui n alimentos e cada alimento i tem um limite máximo de $w[i]$ gramas por pessoa e proporciona $v[i]$ unidades de satisfação por $w[i]$ gramas. No caso, João pode pedir qualquer quantidade de um alimento desde que respeite a quantidade máxima de peso para cada alimento e o peso máximo por pessoa. Escreva uma algoritmo que dê a maior quantidade de satisfação que João pode obter com o limite de k gramas por pessoa. Seu algoritmo deve possuir complexidade de tempo $O(n \log n)$.

- Entrada: $k \in \mathbb{R}^*, w[], v[]$.
- Saída: Maximização de $\sum_{i=0}^{n-1} x[i] \cdot v[i]$, para quantidades $x[i] \leq w[i]$ obedecendo $\sum_{i=0}^{n-1} x[i] \leq k$.

Exercício 3

Modifique o algoritmo anterior para que além de informar a maior quantidade de satisfação que João, também informe quais os alimentos escolhidos e as quantidade de suas porções.

Exercício 4

(Desafio)

Elabore um algoritmo $O(n)$ para o problema da mochila fracionária.

Exercício 5

(Problema de *caching offline*)

Seja k a quantidade de *frames* disponíveis em memória principal e suponha que a sequência de páginas referenciadas seja dada por $(v_0, v_1, \ldots, v_{n-1})$ e conhecida a priori, onde cada v_i é um inteiro representando o identificador da página. Projete um algoritmo que encontre a menor quantidade de *caches miss* possível. Assuma que a *cache* comece vazia.

- Entrada: k, $v[]$.
- O menor número de *caches miss* possível.

Exercício 6

Epaminondas está querendo viajar para a cidade de sua namorada, Astrogilda, que encontra-se do outro lado do Brasil. No decorrer da rota, existem n pontos, de modo que p_0, o primeiro ponto, é a casa de Epaminondas e p_{n-1}, o último ponto, é à casa de Astrogilda. Os pontos intermediários $p_1, \ldots, p_{n-2}$ são postos de gasolina. Assuma que os pontos encontram-se em uma linha reta e que o tanque de Epaminondas está inicialmente cheio. Projete um algoritmo que retorne o número mínimo de paradas que Epaminondas deve fazer até chegar a casa de Astrogilda dado um parâmetro c, denotando a autonomia do tanque em km e um vetor d, de modo que $d[i]$ contém a distância entre p_i e p_{i+1} em km. Seu algoritmo deve retornar -1 se Epaminondas não conseguir chegar a casa de Astrogilda.

- Entrada: o vetor $d[]$ e a autonomia c.
- Saída: o número mínimo de paradas ou -1, caso Epaminondas não consiga chegar à casa de Astrogilda.

Exercício 7

(Inclusão de subintervalos)

Seja S um conjunto de n intervalos sobre a reta real $([l_0, r_0], [l_1, r_1], \ldots, [l_{n-1}, r_{n-1}])$ de modo que, para todo intervalo $[l_i, r_i]$, temos $l_i \leq r_i$. Elabore um algoritmo que forneça o menor conjunto $S' \subseteq S$ de modo que, os intervalos em S' cubram todos os intervalos em S. Formalmente temos que um intervalo $a = [l, r]$ cobre outro intervalo $b = [l', r']$ quando $l \leq l'$ e $r \geq r'$. Seu algoritmo deve ter custo $o(n^2)$.

- Entrada: $S = ([l_0, r_0], [l_1, r_1], \ldots, [l_{n-1}, r_{n-1}])$ um conjunto de intervalos sobre a reta real.
- Saída: $S' \subseteq S$, o menor subconjunto de S que cobre todos os intervalos de S.

Exercício 8

Seja S um conjunto de n intervalos sobre a reta real $([l_0, r_0], [l_1, r_1], \ldots, [l_{n-1}, r_{n-1}])$ de modo que, para todo intervalo $[l_i, r_i]$, temos $l_i \leq r_i$. Elabore um algoritmo que informe uma localização $[a, b]$ nesta reta real em que ocorrem o maior número de sobreposições. Seu algoritmo deve ter custo $o(n^2)$.