Lista 01 de Análise de Algoritmos

Turma do 4º ano

Para cada problema abaixo, desenvolva um algoritmo, descreva o algoritmo (pode ser em alto nível, pseudo-código ou linguagem de programação) e diga qual é a complexidade do seu algoritmo na notação O.

Utilize estratégia de programação dinâmica com memorização.

OBS: O algoritmo pode usar operações como saber o tamanho do vetor e pegar um elemento do vetor da posição i sem custo de cálculo.

Fibonacci: Os números de Fibonacci correspondem a uma sequência infinita na qual os dois primeiros termos são 0 e 1. Cada termo da sequência, à exceção dos dois primeiros, é igual à soma dos dois anteriores, conforme a relação de recorrência abaixo:

$$f_n = f_{n-1} + f_{n-2}$$

Desenvolva dois algoritmos que retorna o i-ésimo termo da sequência.

Tribonacci: Seguindo o exercício anterior, escreva um algoritmo para o Tribonacci. Uma variação do Fibonacci em que o valor de um termo depende dos **3** termos anteriores.

$$f_n = f_{n-1} + f_{n-2} + f_{n-3},$$

e os três primeiros termos são 0, 0 e 1.

Subida das Escadas: Dada uma escada com n degraus, uma pessoa parada em um degrau pode subir um ou dois degraus. Qual a quantidade de combinações diferentes que uma pessoa pode usar para subir a escada.

Máximo Número de Segmentos: Dado um segmento de tamanho n. O objetivo é cortar o segmento na maior quantidade de partes possível. Sendo que cada corte pode somente ter 3 tamanhos possíveis: x, y e z.

Roubo de Casas: Dado um array de tamanho n, com números inteiros positivos. Escolher posições no array tal que a soma dos seus elementos seja máxima, atendendo à restrição de que não se pode escolher duas posições vizinhas. Caso a posição i esteja entre as escolhidas, as posições $i - 1$ e $i + 1$ não poderão ser escolhidas.

Troco em Moedas: Dada uma lista de tipos de `moedas[i]` de tamanho n e um valor soma objetivo S, onde cada elemento `moedas[i]` é um valor diferente de moeda. Considere que você tem uma quantidade infinita de cada moeda. O objetivo é encontrar a mínima quantidade de moedas para juntar a soma S. Se não for possível juntar a soma, retorne o número -1.

Problema da Mochila: Dados dois arrays, `valor[]` e `peso[]`, onde cada elemento na posição `i` de `valor` representa o valor do item `i`, e cada elemento na posição `i` de `peso` representa o peso do item `i`. Também é dado um valor `W` representando o peso total que a mochila aguenta. O objetivo é juntar um conjunto de itens, tal que a soma tenha valor máximo e não ultrapasse o peso `W` da mochila.

OBS: Se um item for incluído, ele deve ser incluído por inteiro, não pode ser usado uma fração do item.