Lista de exercícios 3 (Programação Dinâmica)
Cana - 2023.1

**Questão 1.** Seja $P : \mathbb{N} \to \mathbb{N}$ uma função definida da seguinte forma: $P(0) = P(1) = P(2) = P(3) = P(4) = 0$ e, para $n \geq 5$,

$$P(n) = P\Big(\lfloor \frac{n}{2} \rfloor\Big) \ + \ P\Big(\lfloor \frac{n}{2} \rfloor + 1\Big) \ + \ P\Big(\lfloor \frac{n}{2} \rfloor + 2\Big) \ + \ n.$$

Escreva um algoritmo recursivo puro que recebe um número $n$ como entrada e retorna o valor exato de $P(n)$. Calcule a complexidade do seu algoritmo. Escreva agora um algoritmo de programação dinâmica para o mesmo problema e calcule a complexidade. Escreva também um algoritmo de memoização e calcule a complexidade. Qual dos três algoritmos é o mais rápido?

**Questão 2.** No problema da subsequência crescente mais longa, a entrada é uma sequência de números $a_1, \ldots, a_n$ . Uma subsequência é qualquer subconjunto desses números tomados em ordem, da forma, $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ onde $1 \leq i_1 < i_2 < \ldots, i_k \leq n$, e uma subsequência crescente é aquela na qual os números vão ficando estritamente maiores. Elabore um algoritmo de programação dinâmica para encontrar a subsequência crescente de maior comprimento.

**Questão 3.** Você recebe uma palavra com $n$ caracteres $S[1 \ldots n]$, que você pensa ser um texto corrompido no qual não há pontuação (por exemplo, "*euadoroprogramaçãodinâmica*"). Você deseja reconstruir o seu texto usando um dicionário que disponibiliza uma função booleana $dict(w)$ que retorna verdadeiro, se $w$ é uma palavra do dicionário, e falso, caso contrário. Escreva uma algoritmo que determina se seu texto pode ser reconstruído como uma sequência de palavras válidas. A complexidade deve ser no máximo $O(n^2)$, assumindo que a função *dict* leva tempo constante. Caso seu texto seja válido, faça seu algoritmo escrever a sequência correta de palavras.

**Questão 4.** Dado um vetor $A[1 \ldots n]$, dizemos que uma subsequência $S = [a_{i1}, a_{i2}, \ldots, a_{ik}]$ de $A$ é densa se para todo $j \in 1, \ldots, n$ ($j$ representa uma posição do vetor $A$) temos que, ou $a_j \in S$, ou $a_{j-1} \in S$, ou $a_{j+1} \in S$. Escreva uma algortimo que recebe uma sequência $A$ e encontra a subsequência densa de $A$ cuja soma dos elementos é a menor possível.

**Questão 5.** Uma subsequência é palíndroma se ela é igual lendo da direita para esquerda ou lendo da esquerda para direita. Por exemplo, a sequência (ACGTGTCAAAATCG) possui muitas subsequências palíndromas, como (ACGCA) e (AGTGA). Mas a subsequência (ACT) não é palíndroma. Escreva um algoritmo em $O(n^2)$ que recebe uma sequência $S[1 \ldots n]$ e retorna a subsequência palíndroma de tamanho máximo.

**Questão 6.** É dado um tabuleiro quadriculado com 4 linhas e $n$ colunas e um número inteiro escrito em cada quadrado do tabuleiro. Também é dado um conjunto de $2n$ pedras e queremos colocar algumas delas ou todas elas no tabuleiro (cada pedra pode ser colocada em exatamente um quadrado) para maximizar a soma dos inteiros nos quadrados que são cobertos pelas pedras. Há uma restrição: para que disposição das pedras seja legal, nenhum par delas pode estar em quadrados adjacentes horizontal ou verticalmente (adjacência diagonal é permitida).

(a) Determine o número de padrões legais que podem ocorrer em alguma coluna (isoladamente, ignorando as pedras nas colunas adjacentes) e descreva estes padrões.

Chame dois padrões de compatíveis se eles podem ser colocados em colunas adjacentes em uma disposição legal. Vamos considerar subproblemas consistindo nas primeiras $k$ colunas $1 \leq k \leq n$. A cada subproblema pode ser atribuído um tipo, que é o padrão ocorrendo na última coluna.

(b) Usando as noções de compatibilidade e tipo, forneça um algoritmo de programação dinâmica de tempo $O(n)$ para computar uma disposição ótima.