Lista 6 — MC458
Tema: Programação Dinâmica

1. **O Robô no Tabuleiro com Obstáculos**

   Um robô está posicionado na célula $(1, 1)$ de um tabuleiro $N \times N$ e precisa chegar à célula (N, N). A cada passo, ele pode mover-se apenas uma célula para a direita ou uma para baixo. Algumas células contêm obstáculos intransponíveis, representados por uma matriz de controle $O[i][j]$, onde $O[i][j] = 1$ indica célula bloqueada e $O[i][j] = 0$ indica célula livre. Se a célula inicial $(1, 1)$ ou a final (N, N) contiver um obstáculo, o número total de caminhos é 0.

   a) **Modelagem:** Seja $DP[i][j]$ o número de caminhos válidos de $(1, 1)$ até (i, j). Formule a equação de recorrência para $DP[i][j]$ tratando explicitamente as células livres e com obstáculos, e estabeleça as condições de contorno (casos base).

   b) **Prova de Corretude:** Prove por indução sobre a soma dos índices $s = i + j$ que a relação de recorrência calcula corretamente o total de caminhos válidos de $(1, 1)$ a qualquer célula (i, j) para qualquer configuração de obstáculos.

2. **Lançamentos de Dado**

   Em um jogo, um jogador precisa acumular uma soma exata de N pontos lançando um dado comum de 6 faces quantas vezes forem necessárias. A ordem dos lançamentos importa: por exemplo, para obter a soma $N = 3$, há 4 sequências distintas: $(1, 1, 1)$, $(1, 2)$, $(2, 1)$ e (3).

   a) **Modelagem Genérica:** Seja $DP[n]$ o número de sequências cuja soma é n. Analisando o valor do último lançamento obtido, formule a equação de transição válida para qualquer $n \geq 1$ (lembrando que o dado possui faces de 1 a 6) e justifique a condição inicial $DP[0] = 1$.

   b) **Prova para $1 \leq n \leq 6$:** Prove por indução forte sobre n que para $1 \leq n \leq 6$ vale a fórmula fechada $DP[n] = 2^{n-1}$.

   c) **Quebra do Padrão em $n = 7$:** Calcule $DP[7]$ aplicando a recorrência do item (a) e explique por que a fórmula $2^{7-1} = 64$ deixa de funcionar, identificando em palavras qual sequência de lançamentos se torna impossível a partir de $n = 7$.

3. **A Mochila do Aventureiro**

   Um aventureiro dispõe de n objetos. O objeto i tem peso inteiro $w_i > 0$ e valor $v_i > 0$. A mochila suporta um peso máximo W. Cada objeto deve ser levado por inteiro ou deixado. Por exemplo, se $W = 5$ e os objetos forem $(w, v) \in \{(2, 3), (3, 4), (4, 5), (1, 1)\}$, a combinação ótima resulta em valor 7.

   a) **Modelagem:** Seja $DP[i][w]$ o maior valor alcançável usando apenas os i primeiros objetos com capacidade limite w. Escreva a relação de recorrência e defina os casos base.

   b) **Subestrutura Ótima:** Prove formalmente a propriedade da subestrutura ótima: se um subconjunto $S \subseteq \{1, \ldots, n\}$ é ótimo para a capacidade W, então para qualquer item $k \in S$, o conjunto $S \setminus \{k\}$ deve ser uma solução ótima para o subproblema com itens $\{1, \ldots, n\} \setminus \{k\}$ e capacidade $W - w_k$.

c) **Reconstrução:** Descreva o procedimento algorítmico de rastreamento (*backtracking*) para reconstruir o conjunto exato de objetos que compõe a solução ótima a partir da matriz de DP.

4. **O Caixa Eletrônico (Troco)**

Um caixa eletrônico precisa emitir uma quantia inteira V utilizando o menor número total de moedas. O sistema dispõe de m tipos de moedas $C = \{c_1, c_2, \ldots, c_m\}$ com estoque ilimitado de cada tipo.

a) **Modelagem:** Defina a função $DP[v]$ representando o número mínimo de moedas para somar v. Formule a relação de recorrência, o caso base $DP[0] = 0$ e explicite como representar estados impossíveis (como obter $v = 3$ usando apenas moedas de valor 2).

b) **Prova de Corretude:** Prove por indução forte sobre v que a equação de transição obtém a quantidade mínima de moedas.

c) **Falha do Algoritmo Guloso:** Apresente um contraexemplo concreto (um conjunto de moedas C e um valor V) demonstrando que a estratégia gulosa (escolher sempre a maior moeda possível) falha em encontrar a solução ótima.

5. **O Pulo do Sapo**

Um sapo precisa atravessar um rio saltando sobre N pedras dispostas em linha reta, numeradas de 1 a N. A pedra i tem altura h_i, e o custo de energia ao saltar da pedra i para a pedra j é $|h_i - h_j|$. Ele parte da pedra 1 e deseja atingir a pedra N com o menor custo acumulado.

a) **Salto Curto:** Supondo que o sapo possa saltar apenas 1 ou 2 pedras à frente ($i \to i+1$ ou $i \to i+2$), formule os casos base e a relação de recorrência para $DP[i]$.

b) **Salto Longo:** Generalize a recorrência para o caso em que o sapo pode saltar até k pedras à frente ($1 \leq j - i \leq k$). Determine a complexidade de tempo e espaço do algoritmo em função de N e k.

c) **Prova de Otimalidade:** Prove por indução forte sobre o índice i que a recorrência do item (b) calcula o custo mínimo global para atingir a pedra i.

6. **Alinhamento de DNA (Maior Subsequência Comum – LCS)**

Dadas duas cadeias de caracteres $A = a_1 a_2 \ldots a_m$ e $B = b_1 b_2 \ldots b_n$, deseja-se determinar o comprimento da maior subsequência comum entre elas. Uma subsequência preserva a ordem relativa dos elementos, mas não exige contiguidade. Por exemplo, para $A =$ `AGCAT` e $B =$ `GAC`, a maior subsequência comum tem tamanho 2 (`GC` ou `AC`).

a) **Formulação da Recorrência:** Defina $DP[i][j]$ para os prefixos $A[1..i]$ e $B[1..j]$. Escreva a equação de recorrência de DP separando o caso em que os caracteres finais coincidem ($a_i = b_j$) do caso em que diferem ($a_i \neq b_j$), acompanhada dos casos base.

b) **Prova Teórica:** Prove rigorosamente que quando $a_i = b_j$, a igualdade $DP[i][j] = 1 + DP[i-1][j-1]$ é satisfeita. (*Dica: demonstre separadamente as duas desigualdades,* $DP[i][j] \geq 1 + DP[i-1][j-1]$ *e* $DP[i][j] \leq 1 + DP[i-1][j-1]$.)

c) **Reconstrução:** Explique como recuperar a string da maior subsequência comum utilizando a matriz de DP pré-computada.

7. **O Fotógrafo de Silhuetas (Maior Subsequência Crescente – LIS)**

Um fotógrafo analisa uma sequência de n prédios com alturas $h_1, h_2, \ldots, h_n$. Ele quer selecionar um conjunto de prédios da esquerda para a direita de modo que as alturas escolhidas sejam estritamente crescentes. O objetivo é maximizar o número de prédios selecionados. Por exemplo, para $h = [1, 5, 2, 3, 0]$, a resposta é 3 (correspondendo às alturas $[1, 2, 3]$).

a) **Modelagem:** Defina $DP[i]$ como o comprimento da maior subsequência estritamente crescente que **obrigatoriamente termina** no elemento i. Formule a transição de DP e explicite como obter a resposta final do problema global a partir do vetor DP.

b) **Prova de Corretude:** Prove por indução sobre o índice i a corretude da transição formulada no item (a).

c) **Reconstrução e Complexidade:** Apresente a complexidade de tempo da solução e explique como utilizar um vetor de ponteiros de predecessores (`parent[i]`) para reconstruir a sequência de alturas escolhidas.