Universidade Federal do Ceará
Construção e Análise de Algoritmos
Lista de Exercícios 2 (Programação Dinâmica)

1. Seja $P : \mathbb{N} \to \mathbb{N}$ uma função tal que: $P(0) = P(1) = P(2) = P(3) = P(4) = 0$ e, para $n \geq 5$,

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

(a) 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.

(b) Escreva um algoritmo de programação dinâmica para o mesmo problema e calcule a complexidade.

(c) Escreva um algoritmo de memoização e calcule a complexidade.

2. 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 $O(n^2)$ que recebe uma sequência $S[1 \ldots n]$ e retorna a subsequência palíndroma de tamanho máximo.

3. Você recebe uma sequência $S[1 \ldots n]$ com n dígitos de 0 a 9 e deseja saber se é possível quebrá-la em números que sejam quadrados ou cubos perfeitos. Por exemplo, se $S = 125271448164$, então a resposta é SIM, pois S pode ser quebrada da seguinte forma $125, 27, 144, 81, 64$, cujos números são quadrados ou cubos perfeitos ($125 = 5^3$, $27 = 3^3$, $144 = 12^2$, $81 = 9^2$, $64 = 8^2$). Outra possibilidade seria: $1, 25, 27, 144, 8, 16, 4$. Escreva uma algoritmo de programação dinâmica que determina se sua sequência S satisfaz ou não esta condição. A complexidade deve ser no máximo $O(n^2)$. Caso a resposta seja SIM, faça seu algoritmo escrever a sequência correta de quadrados e/ou cubos perfeitos.

4. 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 de programação dinâmica 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.

5. Você recebe $n + 1$ números reais positivos $X = (x_0, x_1, \ldots, x_n)$ e uma sequência de n operadores em $\{+, \times, \wedge\}$, onde $+$ significa soma, $\times$ significa multiplicação e $\wedge$ significa exponenciação. Essas sequências de números e operadores representam uma expressão matemática. Por exemplo, se $X = (0.3, 1, 4, 0.7, 0.2)$ e a sequência de operadores é $(+, \times, +, \times)$, então temos a expressão: $0.3 + 1 \times 4 + 0.7 \times 0.2$. Desejamos colocar parêntesis na expressão de modo que o resultado final seja o mínimo possivel. Também desejamos colocar parêntesis na expressão de modo que o resultado final seja o máximo possivel. Por exemplo:

- $(0.3 + 1) \times (4 + (0.7 \times 0.2)) = 5.382$,
- $0.3 + (1 \times 4) + (0.7 \times 0.2) = 4.44$,
- $(0.3 + 1) \times (4 + 0.7) \times 0.2 = 1.222$,
- $(0.3 + (1 \times 4) + 0.7) \times 0.2 = 1$.

Nesse exemplo, o máximo é 5.382 e o mínimo é 1. Escreva um algoritmo de programação dinâmica que obtém o modo de colocar parêntesis para obter o valor máximo e o modo de colocar parêntesis para obter o valor mínimo. A complexidade deve ser no máximo $O(n^3)$.

6. Altere o algoritmo da questão anterior para permitir também números reais negativos e também as operações de subtração ($-$) e de divisão ($\div$).

7. O algoritmo de Floyd mostrado em sala de aula tinha um gasto de memória de $\Theta(n^3)$, onde n é o número de vértices do grafo, pois usava $n+1$ matrizes $d_0, d_1, \ldots, d_n$ com n linhas e n colunas. Escreva o algoritmo de Floyd, usando apenas uma matriz d com n linhas e n colunas e explique porque é possível evitar o uso de tantas matrizes usando apenas uma única matriz d.

8. Escreva um algoritmo $O(nT)$ que recebe um inteiro positivo T e uma lista com n inteiros positivos $(a_1, a_2, \ldots, a_n)$ e decide se existe algum subconjunto desses inteiros cuja soma é igual a T. (**Dica1:** Observe subconjuntos $(a_1, a_2, \ldots, a_k)$ e verifique se a soma é s onde $1 \leq k \leq n$ e $1 \leq s \leq T$). (**Dica2:** semelhante ao problema da mochila).

9. Uma balsa leva carros de um lado do rio para o outro. A balsa tem duas pistas para colocar os carros. Cada pista tem tamanho de L metros. Os carros que querem entrar na balsa estão em fila e devem ser colocados na ordem da fila. A fila tem n carros $C_1, C_2, \ldots, C_n$ com tamanhos $T_1, T_2, \ldots, T_n$. Os tamanhos podem ser bastante diferentes. Queremos colocar o maior número de carros na balsa decidindo em qual faixa cada carro deve ser colocado. Elabore um algoritmo de programação dinâmica que resolva esse problema. Como dica, use uma matriz $M[k, A, B]$ que representa o maior número de carros que podem ser colocados na balsa considerando a fila de carros $C_k, \ldots, C_n$, a pista 1 da balsa tendo comprimento de A metros e a pista 2 tendo B metros. Se descobrirmos o valor de $M[1, L, L]$ resolvemos a questão (explique rapidamente porque). (a) Explique sucintamente a propriedade de subestrutura ótima desse problema. (b) Escreva uma recursão para $M[k, A, B]$. (c) Escreva um algoritmo de programação dinâmica para obter $M[1, L, L]$. (d) Altere seu algoritmo para que ele diga em qual pista cada carro deve ser colocado.