Universidade Federal do Ceará
Departamento de Computação
Bacharelado em Ciência da Computação
Construção e Análise de Algoritmos

Lista de exercícios 2 (Programação Dinâmica)

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?

2. Uma subsequência contígua de uma sequência S é uma subsequência de elementos consecutivos de S. Por exemplo, se $S = (5\ 15\ -30\ 10\ -5\ 40\ 10)$, então $(15\ -30\ 10)$ é uma subsequência contígua de S, mas $(5\ 15\ 40)$ não é. Escreva um algoritmo linear para a seguinte tarefa: receba como entrada uma sequência de números $(a_1, a_2, \ldots, a_n)$ e devolva a subsequência contígua cuja soma é máxima (uma subsequência de tamanho zero tem soma zero). No exemplo anterior, a resposta seria a subsequência $(10\ -5\ 40\ 10)$ cuja soma é 55. (Dica: Para cada $j \in \{1, 2, \ldots, n\}$, considere subsequências contíguas terminando exatamente na posição j).

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. 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.

5. Escreva agora um algoritmo polinomial que recebe duas sequências $S[1 \ldots n]$ e $T[1 \ldots m]$ e retorna a subsequência **palíndroma** de tamanho máximo **que é comum a ambas**.

6. Você vai iniciar uma viagem bastante longa. Você inicia a viagem no Km 0 (zero). No seu percurso, existem n hotéis com quilometragens iguais a $a_1 < a_2 < \ldots < a_n$, onde cada a_i é medido a partir do ponto de Km 0. Os únicos lugares que você pode parar são esses hotéis, mas você não precisa parar em todos. Sua viagem termina no hotél do Km a_n que é o seu destino. Você idealmente gostaria de viajar 200 Km por dia, mas nem sempre isso é possível (depende do espaço entre os hotéis). Se você viaja menos de 200 Km em um dia, seu pai reclama que quer chegar logo ao destino final, mas se você viaja mais de 200 Km em um dia, sua mãe reclama que está cansada. Mais especificamente, se você viaja X Km em um dia, você recebe $(200 - X)^2$ reclamações. Você deseja planejar sua viagem de forma a minimizar o número de reclamações recebidos e manter sua família em harmonia. Ou seja, minimizar o número total de reclamações recebidas

em todos os dias viajados. Escreva um algoritmo que determina a sequência ótima de hotéis em que você deve parar.

7. Você recebe uma palavra com n caracteres $S[1 \dots n]$, 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.

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

9. Você deve cortar uma tora de madeira em vários pedaços. A empresa mais em conta para fazer isso é a União Fácil Corte (UFC), que cobra de acordo com o comprimento da tora a ser cortada. A máquina de corte deles permite que apenas um corte seja feito por vez. Se queremos fazer vários cortes, é fácil ver que ordens diferentes destes cortes levam a preços diferentes. Por exemplo, considere uma tora com 10 metros de comprimento, que tem que ser cortada a 2, 4 e 7 metros de uma de suas extremidades. Há várias possibilidades. Podemos primeiramente fazer o corte dos 2 metros, depois dos 4 e depois dos 7. Tal ordem custa 10+8+6 = 24, porque a primeira tora tinha comprimento 10, o que restou tinha 8 metros de comprimento e o último pedaço tinha comprimento 6. Se cortássemos na ordem 4, depois 2, depois 7, pagaríamos $10 + 4 + 6 = 20$, que é mais barato. Seu chefe encomendou um algoritmo de programação dinâmica que, dado o comprimento L da tora e k pontos $p_1, \dots, p_k$ de corte da tora, encontre o custo mínimo para executar esses cortes na UFC.

10. 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 $C_1, C_2, \dots, C_n$ com tamanhos $T_1, T_2, \dots, T_n$. 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 $C_k, \dots, C_n$, 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.

11. Você recebe $n + 1$ números reais positivos $X = (x_0, x_1, \dots, x_n)$ 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, $0.3 + (1 \times 4) + (0.7 \times 0.2) = 4.34$, $(0.3 + 1) \times (4 + 0.7) \times 0.2 = 1.222$, mas $(0.3 + (1 \times 4) + 0.7) \times 0.2 = 1$. 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 $O(n^3)$.

12. Altere o algoritmo da questão anterior para permitir quaisquer números reais e quaisquer operadores em $\{+, \times, \wedge, -, \div\}$, onde $-$ representa subtração e $\div$ representa divisão.