MO417 — Complexidade de Algoritmos I — 2s2019

Lista de Exercício 6

Além dos exercícios abaixo, recomendo que façam a maior quantidade possível de exercícios dos livros texto (CLRS e Manber) dos capítulos relacionados.

Questão 1. Faça um algoritmo de tempo $O(nk)$ que calcula o número $S(n,k)$ de possíveis partições de n elementos em k conjuntos. Qual é a subestrutura ótima?

Questão 2. Crie um algoritmo que, dado um conjunto S de números inteiros positivos e um número inteiro k, decide se existe um conjunto $S' \subseteq S$ que a soma é exatamente k. Qual é a subestrutura ótima? Qual é o tempo de execução do seu algoritmo?

Questão 3. Faça um algoritmo que dadas três sequências, encontra a subsequência comum máxima das três sequências dadas. Qual é a subestrutura ótima? Qual é recorrência que o problema apresenta? Qual é o tempo de execução do seu algoritmo?

Questão 4. Faça um algoritmo que dado um valor k em reais e centavos, encontra a menor quantidade de moedas necessárias que valham exatamente k. Por exemplo, é possível pagar R\$2,37 usando duas moedas de 1 real, três moedas de 10 centavos, uma moeda de 5 centavos e duas de 1 centavo, totalizando 8 moedas, mas existem outra forma de obter o valor usando apenas 6 moedas. Generalize o seu algoritmo para considerar um conjunto fixo S de diferentes valores de moedas.

Questão 5. (CLRS) Considere o problema de imprimir um parágrafo com perfeição usando uma fonte monoespaçada (todos os caracteres têm a mesma largura). O texto de entrada é uma sequência de n palavras de comprimentos $l_1, l_2, \ldots, l_n$, medidos em caracteres. Queremos imprimir este parágrafo de forma organizada em várias linhas que contenham no máximo M caracteres cada. Nosso critério de "organização" é o seguinte. Se uma determinada linha contém palavras de i a j, onde $i \leq j$ e deixamos exatamente um espaço entre as palavras, o número de espaços a mais no final da linha é $M - j + i - \sum_{k=i}^{j} l_k$, que deve ser não negativo para que as palavras se encaixem na linha. Desejamos minimizar a soma sobre todas as linhas com exceção da última dos cubos dos números de espaço a mais nas extremidades das linhas. Dê um algoritmo baseado em programação dinâmica para imprimir um parágrafo de n palavras. Analise o tempo de execução e os requisitos de espaço do seu algoritmo.

Questão 6. O problema da mochila inteira é como a mochila binária, mas neste caso cada item pode ser repetido várias vezes, i.e., um item e pode não ser colocado na solução, ou ser colocado uma ou várias vezes. Resolva este problema por programação dinâmica.

Questão 7. Considere a seguinte versão bidimensional do problema da subsequência consecutiva máxima. Considere uma matriz M, de dimensões $m \times n$ com números inteiros (positivos ou negativos). O objetivo é encontrar uma submatriz N cuja soma dos elementos é máxima. Uma submatriz é exatamente isso que você está pensando.

Questão 8. (CLRS) Cortando uma string

Uma certa linguagem de processamento de strings permite que uma programadora corte uma string em duas partes. Como esta operação copia toda string, ela custa n unidades de tempo para cortar uma string com n caracteres em duas partes. Suponha que uma programadora queira quebrar uma string em muitos pedaços. A ordem em que os cortes ocorrem pode afetar o tempo gasto. Por exemplo, suponha que a programadora deseja quebrar uma sequência de 20 caracteres após os caracteres 2, 8 e 10 (numerando os caracteres a partir de 1 e começando na extremidade esquerda). Se ela programar os cortes para ocorrerem na ordem da esquerda para a direita, o primeiro corte custará 20 unidades de tempo, o segundo

18 unidades de tempo (quebrando a string dos caracteres 3 a 20 no caractere 8) e o terceiro 12 unidades de tempo, totalizando 50 unidades de tempo. Se ela programa os cortes para ocorrerem na ordem da direita para a esquerda, no entanto, o primeiro custará 20 unidades de tempo, o segundo 10 unidades de tempo e o terceiro 8 unidades de tempo, totalizando 38 unidades de tempo. Em um outra ordem de cortes, ela poderia partir primeiro em 8 (custando 20), depois quebrar a parte esquerda em 2 (custando 8) e, finalmente, a parte direita em 10 (custando 12), para um custo total de 40.

Projete um algoritmo que, dados os números de caracteres após os quais se deve cortar a string, determina a sequencia de cortes de menor custo. Mais formalmente, dada uma string S com n caracteres e um vetor $L[1...m]$ contendo os pontos de interrupção, calcule o menor custo para uma sequência de intervalos, juntamente com uma sequência de intervalos que tem esse custo.