Programação dinâmica

Programação Dinâmica

Questão 1. (CLRS) Exercícios: 15.2-1, 15.2-2, 15.2-3, 15.3-1, 15.3-2, 15.3-3, 15.3-4 (3ed), 15.3-5 (2ed), 15.4-1, 15.4-2, 15.4-4, 15.5-2,

Questão 2. (CLRS) Problemas: 15-3 (2ed), 15-5 (3ed), 15-4 (2ed), 15-6 (3ed),

Questão 3. (CLRS) (15-6) Planejando uma festa da empresa

O professor Stewart está consultando o presidente de uma corporação que está planejando uma festa da empresa. A empresa possui uma estrutura hierárquica, isso é, a relação do supervisor forma uma árvore enraizada no presidente. O escritório de pessoal classificou cada funcionário com uma classificação de convívio, que é um número real. Para tornar a festa divertida para todos os participantes, o presidente não deseja que um funcionário e seu supervisor imediato participem.

O professor Stewart recebe a árvore que descreve a estrutura da corporação, usando a representação do filho esquerdo e do irmão direito descrita na Seção 10.4. Cada nó da árvore possui, além dos ponteiros, o nome de um funcionário e a classificação de convívio desse funcionário. Descreva um algoritmo para compor uma lista de convidados que maximize a soma das classificações de convívio dos convidados. Analise o tempo de execução do seu algoritmo.

Questão 4. Relembre o problema da mochila: dados um inteiro K e n itens de pesos diferentes, $S = \{p_1, p_2, \ldots, p_n\}$, encontrar um subconjunto $S' \subseteq S$ cuja soma dos pesos é exatamente K, ou determine que tal conjunto não existe.

Na tabela seguinte, temos uma tabela de programação dinâmica para o problema da mochila parcialmente preenchida. O itens têm pesos $p_1 = 2$, $p_2 = 7$, $p_3 = 5$ e $p_4 = 2$. Preencha os dados que faltam para os itens p_3 e p_4. O símbolo **O** significa existe solução para o subproblema da mochila correspondente ao item, sem usar o item. O símbolo **I** significa que existe solução para a respectiva mochila, usando o respectivo item. O símbolo **-** significa que não existe solução para a respectiva mochila.

Tam. Mochila $\rightarrow$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$p_1 = 2$	O	-	I	-	-	-	-	-	-	-	-	-	-	-	-
$p_2 = 7$	O	-	O	-	-	-	-	I	-	I	-	-	-	-	-
$p_3 = 5$															
$p_4 = 2$															

Questão 5. Melhore o uso de espaço do algoritmo para o problema da mochila apresentado em sala. Há necessidade de usar uma matriz completa $n \times K$? Qual a complexidade de espaço do algoritmo melhorado?

Questão 6. O problema da mochila inteira é uma variante do problema da mochila binária. Nesse problema, cada item corresponde a um tipo e pode ser utilizado várias vezes, i.e., uma solução pode não conter itens de um tipo, conter apenas um, ou conter diversas cópias do mesmo item. 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 consecutiva N cuja soma dos elementos é máxima. Uma submatriz consecutiva é exatamente isso que você está pensando.

Questão 8. (Kleinberg e Tardos) Em um processador de texto, o objetivo da "impressão bela" é tomar texto com uma margem direita irregular, como

```
Call me Ishmael.
Some years ago,
never mind how long precisely,
having little or no money in my purse,
and nothing particular to interest me on shore,
I thought I would sail about a little
and see the watery part of the world.
```

e transformá-lo em um texto cuja margem direita é tão "uniforme" quanto possível, como

```
Call me Ishmael. Some years ago, never
mind how long precisely, having little
or no money in my purse, and nothing
particular to interest me on shore, I
thought I would sail about a little
and see the watery part of the world.
```

Para tornar isso suficientemente preciso, e criar uma impressão bela para o texto, precisamos descobrir o que significa as margens serem "uniformes". Portanto, suponha que nosso texto consista de uma sequência de *palavras*, $W = w_1, w_2, \ldots, w_n$, onde w_i é composta por c_i caracteres. Nós temos um comprimento de linha máximo de L. Vamos presumir que temos uma fonte de largura fixa e ignoramos problemas de pontuação ou hifenização.

Uma *formatação* de W consiste de uma partição das palavras de W em *linhas*. Nas palavras atribuídas a uma única linha, deve haver um espaço após cada palavra, exceto a última; e, portanto, se $w_j, w_{j+1}, \ldots, w_k$ forem atribuídas a uma linha, então devemos ter

$$\left[\sum_{i=j}^{k-1}(c_i+1)\right] + c_k \leq L.$$

Diremos que uma atribuição de palavras para uma linha é *válida* se ela satisfizer essa desigualdade. A diferença entre o lado esquerdo e o lado direito será denominada *folga* da linha, isto é, o número de espaços restantes na margem direita.

Construa um algoritmo eficiente para encontrar uma partição de um conjunto de palavras W em linhas válidas, de modo que a soma dos quadrados das folgas de todas as linhas (incluindo a última linha) seja minimizada.

Questão 9. (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.

Questão 10. Roberto tem uma sorveteria e quer fazer o planejamento de reabastecimento do estoque para um certo período. Ele deseja reduzir os custos de frete e armazenamento. Roberto sempre sabe com antecedência o custo do frete para encomendar sorvete e a demanda de sorvete dos próximos n dias. O valor de uma encomenda em um dia não muda independentemente do número de potes enviados. Ele também sabe qual o custo de manter cada pote de sorvete por uma noite no freezer. O problema é planejar quantos potes de sorvete devem ser encomendados em cada dia.

(a) Formalize o problema: descreva a entrada, uma solução e defina a função-objetivo.

(b) Escreva um algoritmo de programação dinâmica que resolva o problema (i.e., encontra um planejamento de custo mínimo.)