Construção e Análise de Algoritmos

lista de exercícios 24

1. **Multiplicação a, b, c**

   Considere uma operação de multiplicação envolvendo os termos a, b, c, definida pela seguinte matriz

	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

   Quer dizer, $a \cdot b = b$, $b \cdot a = c$, e assim por diante.

   Note que a multiplicação definida por essa matriz não é comutativa nem associativa.

   Isso significa que, ao trocar a ordem dos termos em uma expressão, ou a posição dos parênteses, nós podemos mudar o resultado da expressão.

   Por exemplo,

   $$((b(bb))(ba))c = a \qquad \text{e} \qquad (b(b(bb)))(ac) = c$$

   O nosso problema consiste em, dada uma sequência de multiplicações de termos a, b, c, determinar se existe alguma maneira de colocar parênteses nessa expressão de modo que o seu resultado seja igual a a.

   Apresente um algoritmo de programação dinâmica para esse problema, e estime a sua complexidade.

2. **Quadrado de 1's**

   Considere uma matriz A de dimensão $n \times m$ formada apenas por `0`'s e `1`'s.

   O problema consiste em encontrar o maior quadrado de `1`'s em A.

   Mais precisamente, uma solução qualquer para o problema consiste em um par de coordenadas (i, j) e um número d tal que

   - $A[k, l] = \texttt{1}$, para todo $i \leq k \leq i + d - 1$ e $j \leq l \leq j + d - 1$

   E uma solução ótima é uma solução válida com o maior d possível.

   Apresente um algoritmo de programação dinâmica para esse problema e estime a sua complexidade.

   Dica: Esse é um daqueles problemas onde pensar sobre a programação dinâmica em

termos de preenchimento de uma tabela pode facilitar o processo de chegar até a solução (veja a Discussão 21b).

Abaixo nós temos alguns passos que levam você até o meio do caminho

- defina uma matriz auxiliar `Q` com dimensão $n \times m$, onde cada `Q[i, j]` irá indicar o maior quadrado de `1`'s encontrado até o momento
- inicialize a matriz `Q` da seguinte maneira

$$\texttt{Q[i, j]} \;=\; \begin{cases} \texttt{0} & \text{, se } A[i,j]] \;=\; 0 \\ \texttt{1} & \text{, se } A[i,j]] \;=\; 1 \end{cases}$$

- a seguir, observe que
  - se `Q[i, j]` $=$ `1` e seus 3 vizinhos

    `Q[i + 1, j]` `Q[i, j + 1]` `Q[i + 1, j + 1]`

    são todos `1` também, então nós podemos atualizar `Q[i, j]` para `2`
- observe também que
  - se `Q[i, j]` $=$ `1` e seus 3 vizinhos

    `Q[i + 1, j]` `Q[i, j + 1]` `Q[i + 1, j + 1]`

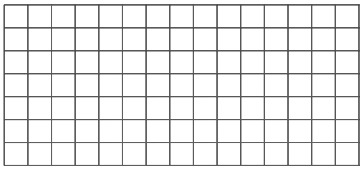
    são todos iguais a `2`, então nós podemos atualizar `Q[i, j]` diretamente para `3`
  - examine a situação em detalhe, e obtenha uma regra que determina o valor de `Q[i, j]` a partir dos valores dos seus 3 vizinhos indicados acima.
  - utilize essa regra para calcular as entradas da matriz auxiliar `Q` em uma única passagem (a ordem em que você faz as coisas é importante).
  - Explique porque isso nos dá uma algoritmo de tempo $O(nm)$ para o problema.

3. **Expedição nas montanhas**

Uma expedição precisa atravessar uma cadeia de montanhas e, para maximizar a sua chance de sucesso, eles decidem utilizar um mapa topográfico como guia.

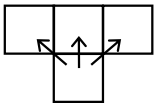
A ideia é que o mapa indica a altitude das diferentes partes do terreno.

De maneira simplificada, nós podemos imaginar que a região é um retângulo, e que o mapa fornece estimativas aproximadas para a altitude em cada quadradinho.



Quer dizer, nós podemos pensar no mapa como uma matriz $\texttt{A}[\texttt{n}, \texttt{m}]$ de números positivos.

Para simplificar ainda mais a situação, os expedicionários decidiram que só vão considerar trajetórias que fazem algum progresso a cada passo:



E o objetivo é minimizar o deslocamento vertical (subidas e descidas) ao longo do caminho.

Por outro lado, a trajetória pode começar em qualquer ponto na parte de baixo do mapa, e pode terminar em qualquer ponto na parte de cima.

a) Apresente um algoritmo de programação dinâmica que encontra uma trajetória para atravessar a cadeia de montanhas, com o menor deslocamente vertical total possível.

b) Estime a complexidade do seu algoritmo.

4. **Atravessando o rio**

Imagine que uma sequência de `*`'s e `-`'s representa a maneira como as pedras estão dispostas em um rio.

```
* * * - * - - * * - - * *
```

Quer dizer, logo em frente à margem esquerda existem 3 pedras, depois um espaço, e daí uma outra pedra, e logo depois um espaço um pouco maior.

Você calcula que para saltar o espaço maior você precisa chegar lá com um pouco mais de impulso ou velocidade.

Por outro lado, se você chegar rápido demais em outros pontos do rio, você pode acabar caindo na água.

Mais precisamente, a cada ponto do caminho você vai estar com alguma velocidade que, por simplicidade, nós vamos assumir que corresponde aos números 1,2,3,....

Quando você está com velocidade k, o seu próximo salto percorre uma distância de $k + 1$ espaços (ou pedras).

E quando você termina o salto, você tem a oportunidade de aumentar ou diminuir a sua velocidade em 1 unidade, ou mantê-la como está.

Sua velocidade nunca pode ser negativa, é claro, e no início ela é 0.

a) Apresente um algoritmo de programação dinâmica que determina se você consegue atravessar o rio ou não dessa maneira.

b) Estime a complexidade do seu algoritmo em termos de n (a largura do rio) e L (a sua velocidade máxima).

5. **Caixeiro-viajante euclidiano bitônico**

No problema do caixeiro-viajante euclidiano, nós temos uma coleção de pontos no plano, e queremos encontrar um caminho que começa em um ponto qualquer, passa por todos os outros pontos exatamente uma vez, e retorna para o ponto inicial.

O objetivo, é claro, consiste em encontrar um caminho que percorra a menor distância possível.

A figura abaixo apresenta duas soluções para um problema com 7 pontos.

Esse problema é realmente muito difícil, mas a seguinte versão simplificada pode ser resolvida por programação dinâmica.

Um *caminho bitônico* completo é que caminho que

- começa no ponto mais à esquerda
- segue sempre para a direita, passando por outros pontos, até alcançar o ponto mais à direita
- retorna andando sempre para a esquerda, passando pelos pontos que foram pulados na ida, até alcançar o ponto mais à esquerda de novo

Note que a segunda solução acima é um caminho bitônico completo.

a) Apresente um algoritmo de programação dinâmica que encontra um caminho bitônico completo com a menor distância possível para uma coleção com n pontos.

b) Estime a complexidade do seu algoritmo.