Universidade Estadual Vale do Acaraú - UVA
Curso: Ciências da Computação
Disciplina: Construção e Análise de Algoritmos
Professor: Cláudio Carvalho

Lista 03 - Programação Dinâmica

1. Algumas aplicações de biologia precisam comparar o DNA de dois ou mais organismos para ver o grau de similaridade entre eles. Uma cadeia de DNA consiste em uma sequência de bases - *adenina*, *timina*, *guanina*, *citosina* - representadas por A, T, G e C, respectivamente. Proponha um algoritmo de complexidade $O(mn)$ que, dadas duas cadeias de DNA com tamanhos m e n, diga o tamanho da maior subcadeia comum a elas. Por exemplo, o tamanho da maior subcadeia comum a $ACCGGTCGAGTGCGCGGAAG$ e $GCGTTCGGAATGCCGTTGC$ é 12.

2. Um *palíndromo* é uma sequência de caracteres que é igual ao seu reverso. Por exemplo, Na sequência $ACGTAGTCAAATCG$, a subsequência destacada é palíndromo. Proponha um algoritmo de complexidade $\Theta(n^2)$ que receba uma sequência S de tamanho n e retorne o tamanho da sua maior subsequência que seja um palíndromo.

3. Você fará uma viagem longa (iniciada no Km 0). No percurso, há n hotéis, cujas distâncias a partir do início são $a_1 < a_2 < \ldots < a_n$. Você deve fazer uma parada por dia, e só pode parar nesses hotéis. A viagem termina no Km a_n. Você deseja viajar $200km$ por dia, mas nem sempre é possível. Viajando X Km por dia, você recebe uma penalização de $|200 - X|$. Escreva um algoritmo de complexidade $O(n^2)$ que, dada uma sequência de n paradas, diga a penalização mínima que você receberá.

4. Você recebe uma palavra S com n caracteres, que parece ser um texto corrompido em que não há pontuação nem acentuação (por exemplo, "euadoroprogramacaodinamica"). Você deseja reconstruir o texto usando um dicionário que disponibiliza uma função booleana $dict(w)$, que diz em tempo constante se uma palavra w está no dicionário. Escreva um algoritmo de complexidade $O(n^2)$ que determine se o seu texto pode ser reconstruído como uma sequência de palavras válidas.

5. Escreva um algoritmo de complexidade $O(nT)$ que receba um inteiro positivo T e uma lista S com n inteiros positivos, e diga se há um subconjunto de S cuja soma seja T.

6. Um *segmento* de um vetor $A[p \ldots r]$ é um subvetor da forma $A[i \ldots k]$, com $p \leq i \leq k \leq r$. A condição $i \leq k$ garante que o segmento não é vazio. Seja a *solidez* de A a maior soma de um segmento deste. Dado um vetor A com n valores inteiros, proponha um algoritmo de de complexidade $O(n)$ que retorne a sua solidez.
   Ex.: A solidez de $A = \langle 5\ 15\ -30\ 10\ -5\ 40\ 10\ -5\ 3 \rangle$ é 55.

7. Você deseja cortar um pedaço de madeira em vários pedaços. Suponha uma máquina que faça apenas um corte de cada vez, e que o custo do corte é igual ao tamanho do pedaço de madeira que é posto na máquina. Observe que a sequência dos cortes faz diferença no custo. Por exemplo, considere um pedaço de madeira de 10 metros que deve ser cortado nos pontos 2, 4 e 7. Se os cortes forem feitos na ordem $2, 4, 7$, o custo será 24; se forem feitos na ordem $4, 2, 7$, o custo sera 20. Proponha um algoritmo de programação dinâmica que, dados um pedaço de madeira de comprimento L e uma sequência de k pontos de corte, diga o menor custo para efetuar os cortes.

8. Dados um pedaço de madeira de tamanho n e duas sequências de k valores $A = \langle a_1, \ldots a_k \rangle$ e $B = \langle b_1, \ldots, b_k \rangle$, indicando que o preço de um pedaço de madeira de tamanho a_i é b_i, $1 \leq i \leq k$. Proponha um algoritmo de complexidade $O(n^2)$ que diga o maior lucro que se pode obter com o pedaço de madeira dado.

9. Dada uma sequência de inteiros $A = \langle a_1, a_2, \ldots, a_n \rangle$, dizemos que S é uma subsequência de A de tamanho k se está na forma: $\langle a_{i_1}, a_{i_2}, \ldots, a_{i_k} \rangle$, onde $1 \leq i_1 < i_2 < \ldots \leq i_k$. S é crescente se $a_{i_1} < a_{i_2} < \ldots < a_{i_n}$. Proponha um algoritmo de programação dinâmica que retorne o tamanho da maior subsequência crescente de uma sequência dada. Por exemplo, para $A = \langle 5, 2, 8, 6, 3, 5, 9, 6, 8 \rangle$, o algoritmo deve retornar o valor 5.

10. A *Distância de Edição* entre duas sequências pode ser vista como o menor número de edições (inserções, remoções e substituições de caracteres) necessárias para deixá-las iguais. Por exemplo, a distância de edição entre $SNOWY$ e $SUNNY$ é 3 (a inserção do U, a troca do O por N, e a remoção do W). Proponha um algoritmo de programação dinâmica que receba duas sequências e calcule a distância de edição entre elas.

11. Uma balsa leva carros de um lado para o outro de um rio. A balsa tem duas pistas, cada uma com tamanho L. Os carros que devem ser embarcados na balsa estão em fila (ou seja, um carro só é embarcado quando chega à frente da fila. A fila tem n carros $\langle c_1, c_2, \ldots, c_n \rangle$ com tamanhos $\langle t_1, t_2, \ldots, t_n \rangle$. Queremos colocar o maior número de carros na balsa, decidindo em que faixa cada carro deve ficar. Proponha um algoritmo de programação dinâmica que resolva esse problema.

    Dica: Use uma matriz $M[k, A, B]$ que indica o máximo de carros que podem ser embarcados, dados a fila $\langle c_k, \ldots, c_n \rangle$, a pista 1 com tamanho A, e a 2 com tamanho B.

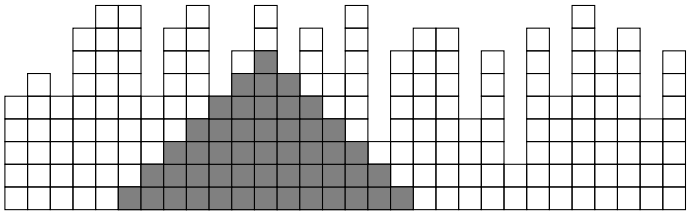
12. Dada uma peça retangular de tecido, de tamanho $X \times Y$, onde X e Y são inteiros positivos, e uma lista de n peças dimensões distintas que podem ser feitos utilizando esta peça de tecido. Cada produto p_i é um retângulo de dimensões $a_i \times b_i$ e tem preço de venda c_i. Assuma que a_i, b_i e c_i são inteiros positivos. Você tem uma máquina que pode cortar uma peça retangular (na horizontal ou na vertical) em duas peças retangulares. Proponha um algoritmo de programação dinâmica que determine o maior lucro que pode ser obtido ao fazer cortes na peça dada.

13. Considere o problema de arrumar a impressão de um parágrafo com uma fonte de tamanho fixo. O texto de entrada é uma sequência de n palavras com tamanhos $t_1, \ldots, t_n$ medidos em caracteres. Queremos imprimir este parágrafo de forma arrumada, considerando que cada linha comporta no máximo m caracteres. Se uma dada linha contém palavras $w_i, \ldots, w_j$, com $i \leq j$, e temos exatamente um espaço entre as palavras, o número de caracteres de espaço extras no final de cada linha é $m-j+i+t_i+\ldots+t_j$, que deve ser não negativo, de forma que as palavras caibam na linha. Queremos minimizar a soma do cubo do número de caracteres extras de cada linha, com exceção da última. Proponha um algoritmo de programação dinâmica para imprimir as n palavras em um parágrafo de forma arrumada.

14. Um empresário deseja abrir uma série de restaurantes ao longo de uma estrada. Os n locais possíveis estão ao longo de uma linha reta, e as distâncias destes ao início da estrada, em km e ordem crescente, são $d_1, d_2, \ldots, d_n$. Em cada local i, $1 \leq i \leq n$, deve ser aberto no máximo um restaurante, e o lucro esperado com isso é p_i, $p_i > 0$. Além disso, quaisquer dois restaurantes devem estar a pelo menos k quilômetros um do outro. Proponha um algoritmo de complexidade $\Theta(n^2)$, que calcule o máximo lucro esperado e apresente uma possível distribuição dos restaurantes que dá esse lucro.

15. Considere o seguinte problema chamado de *3-Partição*. Dada uma coleção de inteiros $X = \langle x_1, \ldots, x_n \rangle$, queremos determinar se é possível fazer uma partição destes valores em três grupos disjuntos I, J e K tais que: $\sum_{x_i \in I} x_i = \sum_{x_j \in J} x_j = \sum_{x_k \in K} x_k = \frac{1}{3} \times \sum_{i=1}^{n} x_i$. Por exemplo, para $X = \langle 1, 2, 3, 4, 4, 5, 8 \rangle$, a resposta é *sim*. Uma possível partição é $I = \langle 1, 8 \rangle$, $J = \langle 4, 5 \rangle$ e $K = \langle 2, 3, 4 \rangle$.

16. (Maratona de Programação da SBC - 2016) Dois irmãos estavam brincando com cubos de madeira e queriam construir um muro, que acabou ficando incompleto, com colunas tendo diferentes alturas, como na figura a seguir:



Eles decidiram que a brincadeira agora seria retirar os cubos das colunas (de cima para baixo), de tal maneira que, no final, reste apenas um triângulo isósceles (nos moldes do que está destacado). Os cubos não podem ser recolocados nas colunas do muro. Proponha um algoritmo de complexidade $O(n)$ que, dada a sequência com as alturas de n colunas, retorne a altura do maior triângulo isósceles que pode ser obtido. Por exemplo, para a sequência de alturas de colunas na figura acima, a resposta é 7.

17. No quadro a seguir, estão definidas as operaçõs de multiplicação sobre os símbolos a, b e c (primeiro termo na vertical, e segundo na horizontal). Observe que, neste caso, a multiplicação não é associativa ($ac \neq ca$) e nem comutativa ($a(bc) \neq (ab)c$).

×	**a**	**b**	**c**
a	b	b	a
b	c	b	a
c	a	c	c

Encontre um algoritmo eficiente que examine uma sequência desses símbolos e diga se é possível fazer as multiplicações em uma ordem tal que o resultado seja um dado símbolo x, onde $x \in \{a, b, c\}$. Por exemplo, se a sequência dada for $bbbbac$, e $x = a$, seu algoritmo deverá responder sim, pois $((b(bb))(ba))c = a$.

Dica: Use uma matriz $P[i, j]$ que armazene todos os possíveis valores da sequência $S[i, \ldots, j]$.

18. Raul é apaixonado por Natália e decidiu apelar para a sorte para saber se ela o ama. Para isso, ele decidiu fazer um *"bem me quer"/"mal me quer"* binário, representados, respectivamente, por 1 e 0. Ele vai pedir que ela diga um número n ($0 \leq n < 10^6$) e fará transformações na sequência que ele tem, até obter a primeira sequência com tamanho igual ou superior a $n+1$, da seguinte forma: cada 0 na sequência atual deve ser trocado por 01, e cada 1 deve ser trocado por 10, para gerar a outra sequência. A resposta que Raul procura estará na posição $n + 1$. Ele acha que terá mais chances de se dar bem iniciando com a sequência $\langle 0 \rangle$. O problema é que Natália , só de mal, pode dizer um número muito grande. Para ajudá-lo, proponha um algoritmo de complexidade $\Theta(n)$ que, dado o n, gere a sequência e diga se ela o ama.

Ex.: Se Natália disser o número 14, Raul precisará fazer as seguintes tranformações: $\langle 0 \rangle \rightarrow \langle 01 \rangle \rightarrow \langle 0110 \rangle \rightarrow \langle 01101001 \rangle \rightarrow \langle 0110100110010110 \rangle$.

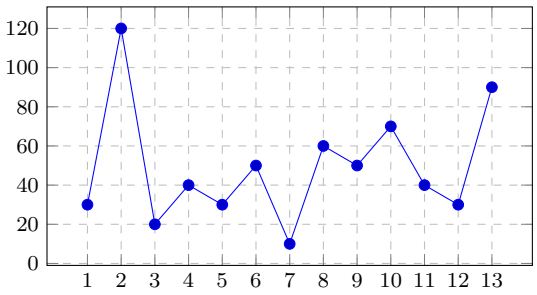
19. Para o problema anterior, considerando que Raul é um rapaz meio apressado e não está interessado na sequência que será produzida, proponha um algoritmo mais rápido que o anterior, não necessariamente de programação dinâmica, que diga apenas se Natália o ama ou não. Apresente a função de complexidade do seu algoritmo.

20. (Adaptada URI - 1900) Dada uma sequência A de valores inteiros; uma subsequência contígua A é dita nula, se não for vazia e a soma de seus elementos for 0. Por exemplo, se $A = \langle 2, -1, 0, -1, 1 \rangle$, são subsequências nulas: $\langle 2, -1, 0, -1 \rangle$, $\langle 0 \rangle$, $\langle 0, -1, 1 \rangle$ e $\langle -1, 1 \rangle$. Dada uma sequência $A = \langle a_1, \ldots, a_n \rangle$, onde $-9 \leq a_i \leq 9$ para $1 \leq i \leq n$, proponha um algoritmo de complexidade $O(n)$ que diga o número de subsequências nulas desta.

Dica: Sejam $A = \langle a_1, \ldots, a_n \rangle$ e $S(i) = \sum_{j=1}^{i} a_j$, $1 \leq i \leq n$.

- se $S(i) = 0$, então $A[1, \ldots, i]$ é nula;
- se $S(i) = S(j) = t$, $1 \leq i < j \leq n$, então a subsequência $A[i + 1, \ldots, j]$ é nula.

21. Você deseja investir na bolsa de valores. Como não tem experiência, selecionou uma única empresa e acompanhou os valores diários de sua ações durante n dias, e ficou curioso para saber o quanto teria ganho se tivesse feito um investimento nas ações dessa empresa nesse período. Você decidiu nunca ter mais do que uma ação simultaneamente. A corretora de valores cobra uma taxa fixa a cada compra de ação. Proponha um algoritmo que, dados a quantidade n de dias, a taxa da corretora e a sequência de valores de ações nos n dias, indique o lucro máximo que pode ser obtido investindo no período, podendo inclusive não investir. Analise a complexidade do seu algoritmo.



Taxa da corretora = 30,00.

Compra		Venda		Lucro
dia	valor	dia	valor	
01	30,00	02	120,00	60,00
07	10,00	10	70,00	30,00
12	30,00	13	90,00	30,00
			Total	**120,00**

Na figura acima, estão representados os valores das ações para um período de 13 dias. Supondo que a taxa para compra de ações é 30, o lucro máximo que pode ser obtido no período é 120 (ver quadro ao lado da figura).

22. Dada uma sequência de n matrizes $M = M_1 \times M_2 \times \ldots \times M_n$, com dimensões $m_0 \times m_1, m_1 \times m_2, \ldots, m_{n-1} \times m_n$, respectivamente. Proponha um algoritmo de programação dinâmica que diga o menor número de multiplicações para calcular o produto das matrizes na sequência dada.

Observações:

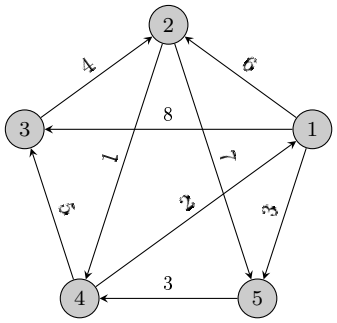
- a multiplicação de matrizes não é comutativa ($A \times B \neq B \times A$), mas é associativa ($A \times (B \times C) = (A \times B) \times C$);
- a multiplicação de uma matriz de ordem $m \times n$ por outra de ordem $n \times p$ exige $m \times n \times p$ operações de multiplicação, e resulta em uma matriz de ordem $m \times p$.

Exemplo: A sequência $\langle 200, 2, 30, 20, 5 \rangle$ indica que são quatro matrizes de ordens 200×2, 2×30, 30×20 e 20×5, respectivamente. Algumas formas de fazer a multiplicação são mostradas no quadro a seguir:

Ordem de Multiplicação	Cálculo do Custo					Custo
$(((M_1 \times M_2) \times M_3) \times M_4)$	$200 \times 2 \times 30$	$+$	$200 \times 30 \times 20$	$+$	$200 \times 20 \times 5$	152.000
$((M_1 \times M_2) \times (M_3 \times M_4))$	$200 \times 2 \times 30$	$+$	$30 \times 20 \times 5$	$+$	$200 \times 30 \times 5$	45.000
$(M_1 \times ((M_2 \times M_3) \times M_4))$	$2 \times 30 \times 20$	$+$	$2 \times 20 \times 5$	$+$	$200 \times 2 \times 5$	3.400
$((M_1 \times (M_2 \times M_3)) \times M_4)$	$2 \times 30 \times 20$	$+$	$200 \times 2 \times 20$	$+$	$200 \times 20 \times 5$	29.200

23. Dados um fornecimento ilimitado de moedas de valores $m_1, \ldots, m_n$ e um valor v, proponha um algoritmo de complexidade $O(nv)$ que diga se é possível pagar exatamente o valor v com as moedas existentes. Por exemplo, se os valores disponíveis de moedas são $\{5,\ 10\}$, não é possível pagar 12, mas é possível parar 35.

24. Dado um digrafo $G = (V, A)$, onde V é um conjunto de vértices, e A é um conjunto de arcos. Sabendo-se que D é ponderado, e que o peso de um arco (x, y) é o tamanho do caminho que liga x diretamente a y (se não há caminho diretamente de x para y, admita que o tamanho deste é ∞), a distância entre dois vértices x e y é o tamanho do menor caminho direcionado de x para y. Proponha um algoritmo programação dinâmica, de complexidade $O(n^3)$ (onde $n = |V|$), que calcule a distância entre todos os pares de vértices de um digrafo D.



	1	2	3	4	5
1	0	6	8	∞	3
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	5	0	∞
5	∞	∞	∞	3	0

Pesos

	1	2	3	4	5
1	0	6	8	6	3
2	3	0	6	1	6
3	7	4	0	5	10
4	2	8	5	0	5
5	5	11	8	3	0

Distâncias

Para o grafo acima, cuja matriz de pesos está ilustrada ao lado, a matriz com todas as distâncias entre todos os pontos está representada à direita.

Dica: Algoritmo de Floyd-Warshall.

25. Considere o jogo em que um mediador produz uma sequência $s_1, \ldots, s_n$ de cartas, cada cada carta s_i tem valor v_i, $1 \leq i \leq n$. Em seguida, dois jogadores devem pegar uma carta da sequência, mas apenas a primeira ou a última carta da sequência restante podem ser escolhidas. Assuma que n é par, de modo que, ao final, ambos os jogadores fiquem com o mesmo número de cartas. O vencedor será aquele cujas cartas somam o maior valor. Sabe-se que o Jogador 1 deve fazer a primeira escolha, e você quer ajudá-lo a não perder o jogo.

    a) Mostre uma sequência de cartas tal que não é ótimo para o primeiro jogador começar pegando a carta de maior valor, dentre as duas cartas das extremidades;

    b) Proponha um algoritmo de complexidade $O(n^2)$ que diga as cartas que o Jogador 1 deve selecionar, de forma que ele não perca o jogo.

    Dica: Dada a sequência inicial, seu algoritmo deve computar em tempo $O(n^2)$ algumas informações, para que, com base nelas, o jogador faça suas escolhas em tempo $O(1)$. Seja $M(i, j)$ a melhor jogada para o conjunto de cartas $s_i, \ldots, s_j$, onde $M(i, j) = max\{s_i - M(i + 1, j), s_j - M(i, j - 1)\}$.

Universidade Estadual Vale do Acaraú - UVA
Curso: Ciências da Computação
Disciplina: Construção e Análise de Algoritmos
Professor: Cláudio Carvalho

Lista 03 - Programação Dinâmica (Definições das Soluções)

1. i. L[i,j]: Maior subsequência comum de X[0...i-1] e Y[0...j-1]

   ii. $$M[i,j] = \begin{cases} 0, \text{ se } i=0 \text{ ou } j=0; \\ 1 + L[i-1,j-1], \text{ se } i>0,\ j>0 \text{ e } X[i-1]=Y[j-1]; \\ \max(L[i,j-1],L[i-1,j]), \text{ se } i>0,\ j>0 \text{ e } X[i-1]\neq Y[j-1]. \end{cases}$$

   iii. Calcular L[m,n].

2. i. P[i,j]: Maior subsequência palíndromo de S[i...j]

   ii. $$P[i,j] = \begin{cases} 0, \text{ se } i>j; \\ 1, \text{ se } i=j; \\ 2 + P[i+1,j-1], \text{ se } i<j \text{ e } S[i]=S[j]; \\ \max(P[i,j-1],P[i+1,j]), \text{ se } i<j \text{ e } S[i]\neq S[j]. \end{cases}$$

   iii. Calcular P[1,n].

3. i. R[i]: Menor número de reclamações até o km a_i

   ii. $$R[i] = \begin{cases} 0, \text{ se } i=0; \\ \min\{R[k]+|200-(A[i]-A[k])|:\ 0\leq k\leq n \text{ e } A[k]\leq A[i]\}; \end{cases}$$

   iii. Calcular R[n].

4. i. T[i]: Diz se os primeiros i caracteres da sequencia formam um texto válido

   ii. $$T[i] = \begin{cases} 1, \text{ se } i=0; \\ 1, \text{ se } i>0 \text{ e existe } k<i \text{ tal que } T[k]=1 \text{ e dict}(S,k+1,i); \\ 0, \text{ caso contrário.} \end{cases}$$

   iii. Calcular T[n].

5. i. S[i,j] : maior soma de elementos de T[1...i] não superior a j.

   ii. $$S[i,j] = \begin{cases} 0, \text{ se } i=0 \text{ ou } j=0; \\ S[i-1,j], \text{ se } i>0,\ j>0 \text{ e } A[i]>j; \\ \max(S[i-1,j],A[i]+S[i-1,j-A[i]]), \text{ se } i>0,\ j>0 \text{ e } A[i]\leq j. \end{cases}$$

   iii. Verificar se S[n,T]=T.

6. i. S[i] : maior soma de elementos de A[0...i], incluindo o A[i].

   ii. $$S[i] = \begin{cases} A[0], \text{ se } i=0; \\ \max(S[i-1],0)+A[i], \text{ se } i>0. \end{cases}$$

   iii. max{S[i]: $0 \leq i \leq n-1$}.

7. i. C[i,j] : menor custo para fazer os cortes necessários entre as marcas i e j.

   ii. $$C[i,j] = \begin{cases} 0, \text{ se } j-i\leq 1; \\ 0, \text{ se } j-i>1 \text{ e não há um ponto de corte entre } i \text{ e } j; \\ \min\{C[i,Pk]+C[Pk,j]:\ i<Pk<j\} + (j-i), \text{ caso contrário.} \end{cases}$$

   iii. Calcular C[0,l].

8. i. L[i] : maior lucro obtido a partir de uma madeira de tamanho i.
dados os tamanhos A[1...k] e os valores B[1...k].
V[i]: valor de venda de um pedaço de madeira de tamanho i, caso exista, ou 0.

ii. $L[i] = \begin{cases} 0, \text{ se } i = 0; \\ \max\{L[j]+L[i-j]: \ 1 \leq j \leq i\}, \text{ se } i>0. \end{cases}$

iii. Calcular L[n].

9. i. S[i]: maior qtd. de elementos de uma subseq. crescente de A[0...i].

ii. $S[i] = \begin{cases} 1, \text{ se } i=0; \\ \max\{1+S[k]: \ 0 \leq k \leq i \text{ e } A[k]<A[i]\}, \text{ se } i>0. \end{cases}$

iii. Calcular S[n-1].

10. i. E[i,j]: distância de edição entre A[0...i-1] e B[0...j-1].

$diff(X,Y) = \begin{cases} 1, \text{ se } X \neq Y; \\ 0, \text{ caso contrário.} \end{cases}$

ii. $E[i,j] = \begin{cases} i, \text{ se } j=0; \\ j, \text{ se } i=0; \\ \min\{E[i-1,j],E[i,j-1],E[i-1,j-1] + diff(A[i-1],B[j-1])\}. \end{cases}$

iii. Calcular E[m,n].

11. i. M[k,i,j]: maior numero de carros da fila C[k...n-1] que podem ser colocados na balsa - pista 1 com tam. i, e 2 de tam. j.

ii. $M[k,i,j] = \begin{cases} 0, \text{ se } k \geq n; \\ 0, \text{ se } k<n \text{ e } C[k]>i \text{ e } C[k]>j; \\ 1 + M[k+1,i-C[k],j], \text{ se } k<n \text{ e } i \geq C[k]>j; \\ 1 + M[k+1,i,j-C[k]], \text{ se } k<n \text{ e } i<C[k] \leq j; \\ 1 + \max(M[k+1,i-C[k],j],M[k+1,i,j-C[k]), \text{ caso contrário.} \end{cases}$

iii. Calcular M[0,L,L].

12. i. L[i,j]: maiorlucro de uma peca de tecido de dimensoes ixj.
V[i,j]: maior lucro de uma peca de tamanho ixj (0, se não existir).

ii. $L[i,j] = \begin{cases} 0, \text{ se } i=0 \text{ ou } j=0; \\ \max\{V[i,j], \max_{1 \leq k<j}\{L[i,k]+L[i,j,k]\}, \max_{1 \leq k<i}\{L[k,j],L[i-k,j]\}\}. \end{cases}$

iii. Calcular L[X,Y].

13. Dada uma sequência de palavras W[0,...,n-1] na ordem inversa à que deve ser impressa.

i. D[i]: defeito mínimo do parágrafo formado pelas palavras w1...wi.
C(m,k) = Tm+1+...+1+Tk. (Obs.: C(m,k) é curto se for no máximo L.)

ii. $D[i] = \begin{cases} 0, \text{ se } C(i,0) \leq L; \\ \min\{(L-C(i,k))^3 + D[k-1]: \ 0<k \leq i \text{ e } C(i,k) \leq L\}. \end{cases}$

iii. Calcular D[n-1].

14. i. L[i]: lucro máximo esperado até a localidade i.

ii. $L[i] = \begin{cases} P[0], \text{ se } i=0; \\ \max\{P[i],L[i-1],\max\{P[i]+L[j]: \ 0 \geq k<i \text{ e } D[i]-D[j] \geq k\}\}, \text{ se } i>0 \end{cases}$

iii. Calcular L[n-1].

15. Sejam um conjunto de inteiros A[1..n], e T a soma dos elementos de A.
Se T for múltiplo de 3, faça s=T/3; senão, retornar 0.

   i. M[i,j,k]: 1, se existem subconjuntos de A[1...i] cujas somas são j e k.

   ii. $$\texttt{M[i,j,k]} = \begin{cases} \texttt{1, se i=0, j=0 e k-0;} \\ \texttt{0, se i=0, e j>0 ou k>0;} \\ \texttt{(A}_i \texttt{ } \leq \texttt{j } \cap \texttt{ M[i-1,j-A}_i\texttt{,k]) } \cup \texttt{ (A}_i \texttt{ } \leq \texttt{k } \cap \texttt{ M[i-1,j,k-A}_i\texttt{]) } \cup \texttt{ (M[i-1,j,k]).} \end{cases}$$

   iii. Calcular M[n,s,s].

16. Sejam H[0,...,n-1] a sequência de alturas das torres de cubos.

   i. E[i] a altura da maior escada a partir da esquerda, e
   D[i] a altura da maior escada a partir da direita, ambas incluindo a torre i.

   ii. $$\texttt{E[i]} = \begin{cases} \texttt{1, se i=0;} \\ \texttt{min(E[i-1]+1,H[i]), para 0<i} \leq \texttt{n-1.} \end{cases}$$
   $$\texttt{D[i]} = \begin{cases} \texttt{1, se i=n-1;} \\ \texttt{min(D[i+1]+1,H[i]), para 0} \geq \texttt{i<n-1.} \end{cases}$$

   iii. Calcular max{min(E[i],D[i]):0≤i≤n-1}

17. i. P[i,j]: conjunto de todos os possíveis valores da sequência S[i,...,j].

   ii. $$\texttt{P[i,j]} = \begin{cases} \emptyset\texttt{, se i>j;} \\ \{\texttt{S[i]}\}\texttt{, se i=j;} \\ \{\texttt{x}\times\texttt{y: i} \leq \texttt{k<j e } \forall \texttt{x} \in \texttt{ P[i,k] e } \forall \texttt{y} \in \texttt{ P[k+1,j]}\}\texttt{.} \end{cases}$$

   iii. verificar se x ∈ P[0,n-1]

18. Alocar vetor B com t=$2^{\lceil \log n \rceil}$ elementos.

   i. B[i]: valor da posição i da sequência.

   ii. $$\texttt{B[i]} = \begin{cases} \texttt{0, se i=0;} \\ \texttt{B[i/2], se i>0 e i for par;} \\ \neg\texttt{B[i/2], se i for ímpar.} \end{cases}$$

   iii. Apresentar o valor de B[n]

19. Fazer uma função recursiva bmq(i), com base na questão anterior, que diga o valor da posição i.
$$\texttt{bmq(i)} = \begin{cases} \texttt{0, se i=0;} \\ \texttt{bmq(i/2), se i>0 e i for par;} \\ \neg\texttt{bmq(i/2), se i>0 e i for ímpar.} \end{cases}$$

20. i. Seja S[i] = soma dos elementos $A_1 \dots A_i$.
   Contar quantas vezes cada valor S[i] acontece.

   ii. Calcular quantidade de sequências nulas, com base nas observações dadas na questão.

21. i. Dada a sequência de valores de ações $V[1 \dots n]$ e o valor da corretora C.
   L[i]: maior lucro no período $0 \dots i$.

   ii. $$\texttt{L[i]} = \begin{cases} \texttt{0, se i} \leq \texttt{ 1;} \\ \texttt{max}\{L[k-1] \texttt{ + max}(0, V[i] - V[k] - C)\} \end{cases}$$

   iii. calcular L[n].

22. i. Dada a sequência de valores $P[0 \ldots n]$ correspondente às ordens das matrizes.
Ordem da matriz $A_i = P_i \times P_{i+1}$, para $0 \leq i \leq n-1$.
M[i,j]: menor número de multiplicações escalares de $A_i \times \ldots \times A_j$.

ii. $\texttt{M[i,j]} = \begin{cases} \texttt{0, se i = j;} \\ \min\{M[i,k] + M[k+1,j] + P_i \cdot P_{k+1} \cdot P_{j+1} \mid i \leq k < j\} \end{cases}$

iii. calcular M[0,n-1].

23. i. V[i]: maior valor, não superior a i, que pode ser pago com as moedas $m_1 \ldots m_n$.

ii. $\texttt{V[i]} = \begin{cases} \texttt{0, se não existe moeda com valor menor ou igual a i;} \\ \max\{m_k + V[i - M_k] \mid m_k \leq i, 1 \leq k \leq n\} \end{cases}$

iii. verificar se V[T] = T.

24. i. D[i,j,k]: menor distância de i a j passando pelos vértices $[1 \ldots k]$.
W[i,j]: peso do arco ij.

ii. $\texttt{D[i,j,k]} = \begin{cases} \texttt{W[i,j], se k = 0;} \\ \text{mín}\{D[i,j,k-1], D[i,k,k-1] + D[k,j,k-1]\}\texttt{, se k > 0;} \end{cases}$

OBS.: Como, para um dado k, só é analisada a dimensão anterior, pode ser usada uma matriz de apenas duas dimensões.

25. i. Dada uma sequência de n cartas $\langle s_0 \ldots s_{n-1} \rangle$
S[i,j]: maior soma de cartas do conjunto $s_i \ldots s_j$.

ii. $\texttt{S[i,j]} = \begin{cases} s_i\texttt{, se i = j;} \\ \max(s_i - S[i+1,j],\ \ s_j - S[i,j-1])\texttt{, se i < j;} \end{cases}$

iii. calcular A[0,n-1]