**Lucca Ferreira Paiva 240229**

# **Projeto de Algoritmo com Implementação 2**

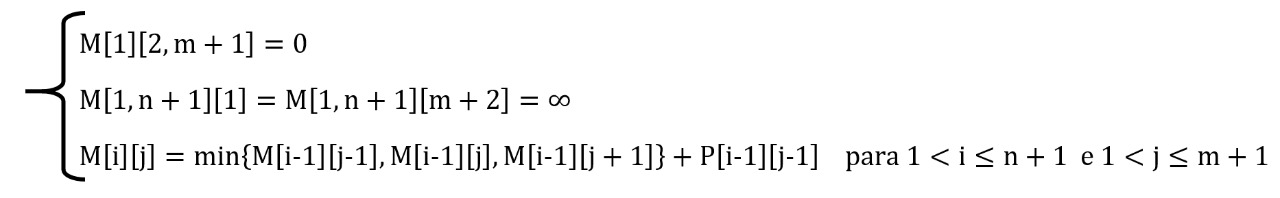
**Subestrutura ótima:**

Suponha um caminho X ótimo que saia do ponto A, na base da parede, e vá até C, no topo da escalada, passando por B. Podemos dividir esse caminho em dois, um de A até B denominado de Y, outro de B até C, chamado de W. Se X é um caminho ótimo, então Y e W também tem que ser, pois, se pelo menos um deles não fosse, X também não seria, já que seria possível reduzir ainda mais o custo de X. Logo, todo caminho ótimo é composto por subcaminhos também ótimos. Isso nos garante a subestrutura ótima do problema.

**Recorrência:**

Sabendo disso, podemos elaborar um algoritmo para determinar o custo do caminho de menor custo da base até o topo da parede. Vamos dizer que a parede é representada pela matriz P, de dimensões n x m, considerando que a sua primeira linha representa o custo da linha da base da escalada. Agora, vamos criar uma matriz M, de dimensões (n + 1) x (m + 2), em que a primeira linha são somente zeros, a primeira e a última coluna são compostas pelo valor infinito.

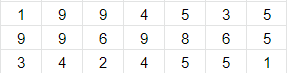
A matriz M tem o propósito de armazenar os custos até cada ponto. Isso quer dizer que cada posição da matriz M corresponde ao custo mínimo total para se chegar a esta posição. Para as linhas de M, cada elemento X = M[ i ][ j ] será dado pelo mínimo entre os elementos M[i - 1][j - 1], M[i - 1][j] e M[i - 1][j + 1], somados ao elemento de P correspondente, no caso P[i - 1][j - 1]. Em outras palavras, para um elemento X da linha i de M, ele é dado pelo mínimo dos três elementos superiores a ele, o superior da direita, do meio e da esquerda, somados ao valor do elemento correspondente em P. Esta relação é dada por:



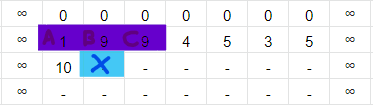
Nesta situação, os casos base são representados pela primeira linha, a de zeros, e partir disso, todas as demais têm o mesmo tratamento. E são calculadas justamente pela fórmula explicitada acima. Vale ressaltar que as colunas das extremidades são justamente para facilitar o cálculo dos valores de M, por isso j varia de 2 até m + 1, ignorando a coluna 1 e a m + 2.

**Descrição do algoritmo:**

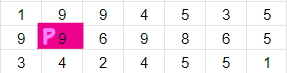
De forma mais gráfica, o processo pode ser visto por:



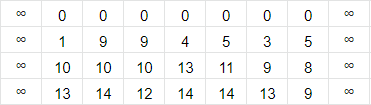
Matriz P que representa o custo de cada posição da escalada



Matriz M sendo construída, no caso o valor de X depende de A, B e C



Valor de p na matriz base P, conectado a X



Matriz M completa

Pelas imagens, temos que o elemento X da matriz M, é dado pelo mínimo entre A, B e C, somado de p da matriz P. Ou seja, X = min{A, B, C} + p = min{1, 9, 9} + 9, e portanto temos X = 1 + 9 = 10. E este mesmo processo é repetido para determinarmos todos os elementos de M.

Na verdade, o algoritmo utilizado foi com este princípio, mas apresentou pequenas mudanças. A mais significativa foi que, de ao invés de utilizar uma matriz (n + 1) x (m + 2) como base, se utilizou uma 2 x (m + 2), isso porque, como só estamos interessados no custo final, o processo todo pode ser feito com apenas duas linhas da matriz M. (Na prática, a primeira coisa a ser feita foi montar a matriz P, com base na entrada recebida pelo teclado). Feito isso, preencheu-se a matriz M com zeros na primeira linha **(1)**. Depois disso, na primeira e na última coluna colocou-se os valores de infinito**(2)** (na prática o valor de 2147483647, o maior inteiro possível possível). Estes valores serão úteis para facilitar na hora de calcularmos os mínimos das linhas anteriores.

Feito isso, agora começa o processo de realmente calcular os custos. Para tal, fez-se um laço que percorre cada linha, e, dentro dele, um segundo for da segunda até a penúltima coluna de M, ou seja, o primeiro de 1 até n + 1, e o outro de 2 até m + 1 **(3)**. Vale deixar claro que a primeira e a última coluna não entram porque contém o valor infinito, e isso facilita pois deixa de ser necessário tratar como exceções os valores das bordas de P. Dentro deste loop, há uma função responsável por, dada uma matriz e uma posição i e j, achar o mínimo entre os seus três antecessores(processo descrito nas imagens)**(4)**. Achado o mínimo, colocamos em M o valor deste mínimo somado ao valor do elemento de P na posição P[i - 1][j - 1] (O -1 é porque M possui uma linha a mais na posição zero, e uma coluna na posição zero também).

Ao terminarmos o primeiro laço, temos então que a segunda linha da matriz M corresponde ao custo para chegar a cada elemento desta linha. Neste ponto, desloca-se esta segunda linha para a primeira, pois os novos valores serão colocados na segunda linha**(5)**. Já que é justamente a nova primeira linha que será utilizada como base.

**Análise de Complexidade:**

Para a complexidade de tempo, temos que cada operação tem sua complexidade de tempo descrita por (utilizar os índices do texto acima como base):

1. Preencher a primeira linha de M com zeros: O(m)
2. Colocar nas colunas de M de 1 até m + 2 o valor infinito: O(1) (são apenas 4 valores)
3. Um laço de 2 até m + 1 ( O(m)), encadeado em um de 1 até n + 1 ( O(n)): O(m\*n)
4. Deslocar uma linha de tamanho m, em um total de n vezes: O(m\*n)

Complexidade de tempo total: O(n\*m), mas como m ∈ O(n)

**Complexidade de tempo total: O(n^2)**

Já para a complexidade de espaço, temos a matriz P, de tamanho n x m, é uma matriz M de 2 x m + 2, portanto, a **complexidade de espaço é de O(n^2)**. Porém, vale ressaltar que se desejado é possível coletar apenas uma linha da matriz P (no caso se tratando de recebê-la pela entrada padrão), e fazer seu processamento, resultando em uma complexidade de espaço de O(m).

Porém não fiz isso, pois considerando que a matriz é recebida pela entrada padrão, se não guardasse os valores eles seriam perdidos, e acho que seria estranho saber o custo mínimo sem saber para qual parede se refere.