Sliding Window Maximum - LeetCode 239

https://leetcode.com/problems/sliding-window-maximum

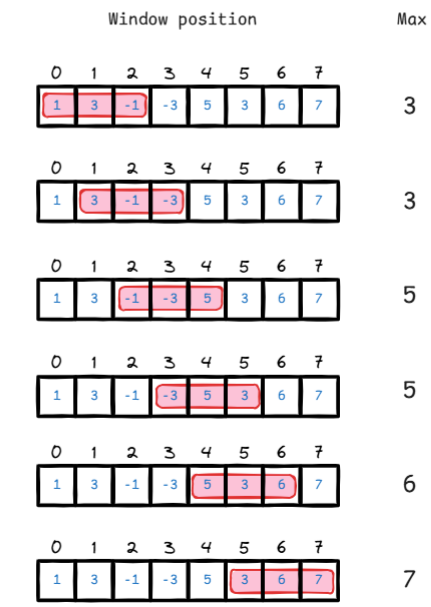
Dado um array de inteiros nums e uma janela deslizante de tamanho k que se move da esquerda do array para a direita. Olhando para os k números na janela cada vez que a janela deslizante se move para a direita em uma posição.

Retorne o valor máximo na janela deslizante para cada posição

Exemplo:

```
Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
Output: [3,3,5,5,6,7]
```

Explicação:



Como Implementar?

Força Bruta:

```python
class Solution:
    def maxSlidingWindow(self, nums: List[int], k: int) -> List[int]:
        n = len(nums)
        if n * k == 0:
            return []
        if k == 1:
            return nums

        result = []
        for i in range(n - k + 1):
            window_max = max(nums[i:i + k])
            result.append(window_max)
        return result
```

Complexidade: O(n*k)

LeetCode: TLE

Como Melhorar?

Queremos obter o valor máximo, ao que isso remete?

Heap!

É possível aplicar uma heap nesse problema?

1) Adiciona elementos da janela na Heap
2) Pega o maior valor
3) Verifica se está dentro da janela atual
   3.1) Se estiver na janela, adiciona ele ao vetor resultado
   3.2) Se não estiver, remove da heap e pega o próximo

```python
class Solution:
    def maxSlidingWindow(self, nums: List[int], k: int) -> List[int]:
        maxHeap = [ (-nums[i], i) for i in range(k) ]
        heapq.heapify(maxHeap)


        output = []
        startWindow = 0
        endWindow = k-1
        while endWindow<len(nums):
            while True:
                maxVal, index = maxHeap[0] #peek maxHeap
                if index>=startWindow and index<=endWindow:
                    break #stops when peek maxHeap is in sliding window
                heapq.heappop(maxHeap) #remove elements out of sliding window
            output.append( -maxVal )
            startWindow += 1
            endWindow += 1
            if endWindow<len(nums):
                heapq.heappush(maxHeap, (-nums[endWindow], endWindow))


        return output
```

Longest Common Subsequence - LeetCode 1143

https://leetcode.com/problems/longest-common-subsequence

Dadas duas strings text1 e text2, retorne o comprimento da maior subsequência comum. Se não houver subsequência comum, retorne 0. Uma subsequência é formada removendo alguns caracteres sem alterar a ordem dos restantes.

Exemplo1:

Input: text1 = "abcde", text2 = "ace"

Output: 3

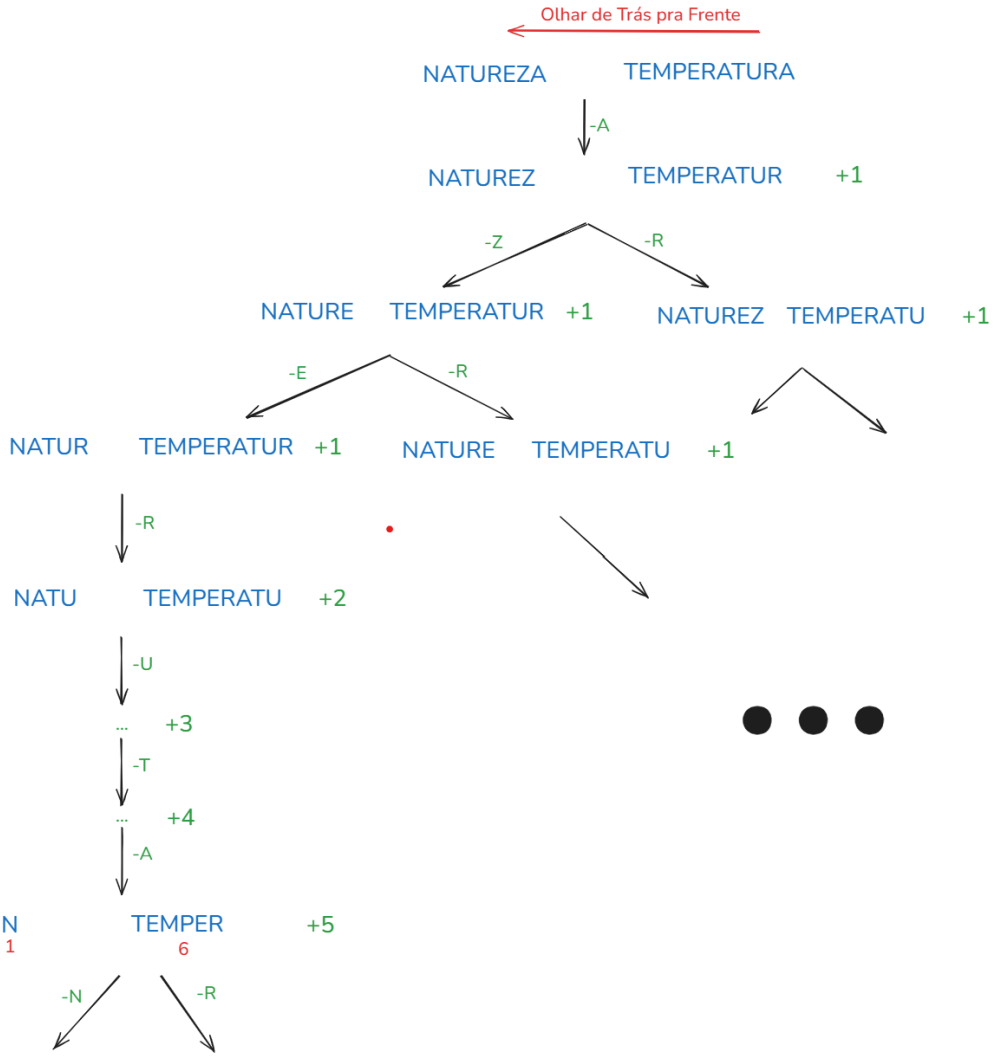
Explicação: "abcde" "ace"

Exemplo2:

Input: text1 = "natureza", text2 = "temperatura"

Output: 5

Como Implementar?



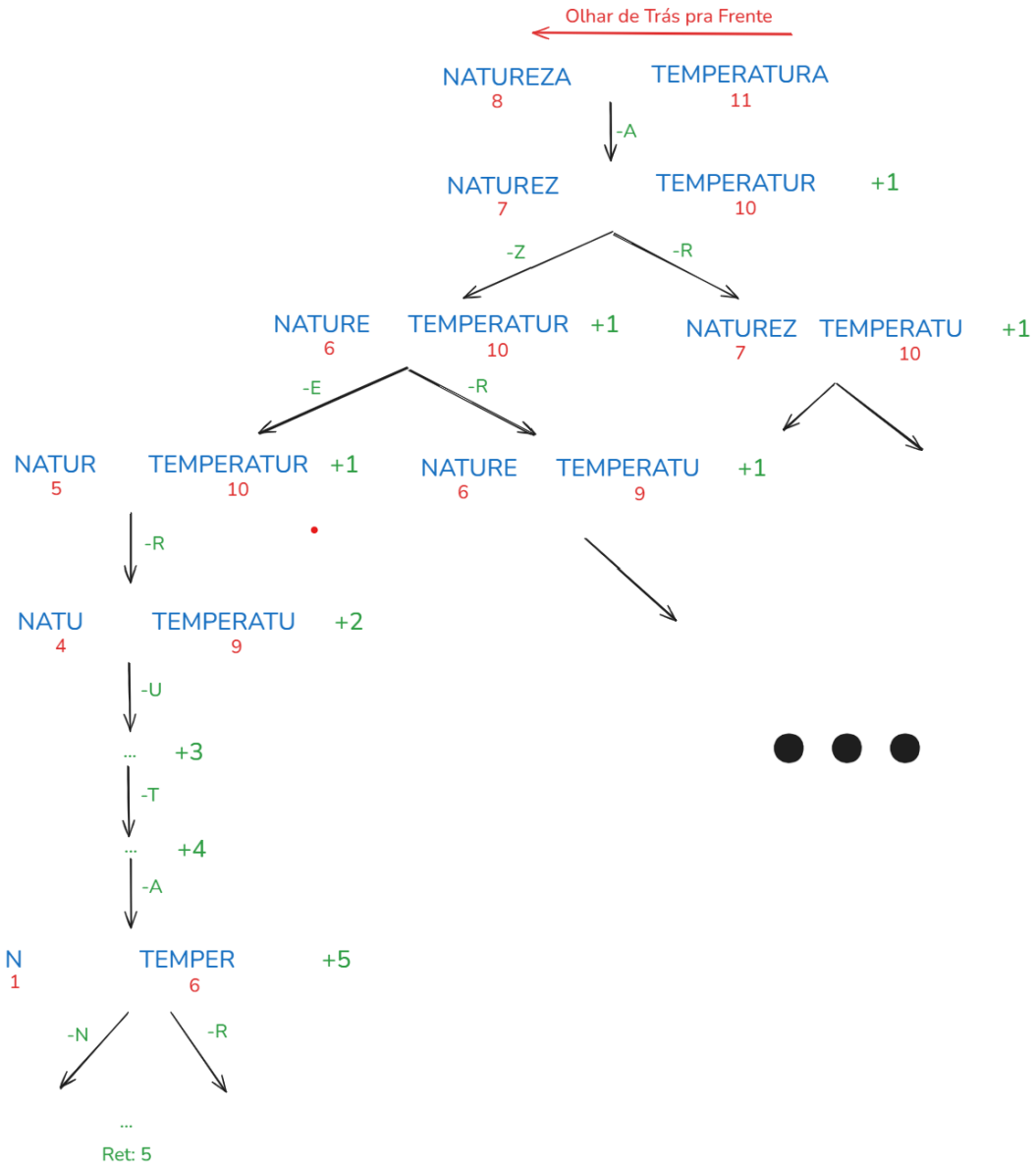

```python
def longestCommonSubsequence(text1, text2):
    def lcs_recursive(i, j):
        if i == 0 or j == 0:
            return 0
        elif text1[i - 1] == text2[j - 1]:
            return 1 + lcs_recursive(i - 1, j - 1)
        else:
            return max(lcs_recursive(i, j - 1), lcs_recursive(i - 1, j))

    return lcs_recursive(len(text1), len(text2))
```

Complexidade: O(2^(m+n))

Como eliminar necessidade de checagens repeditas?

Programação Dinâmica -> Memorização

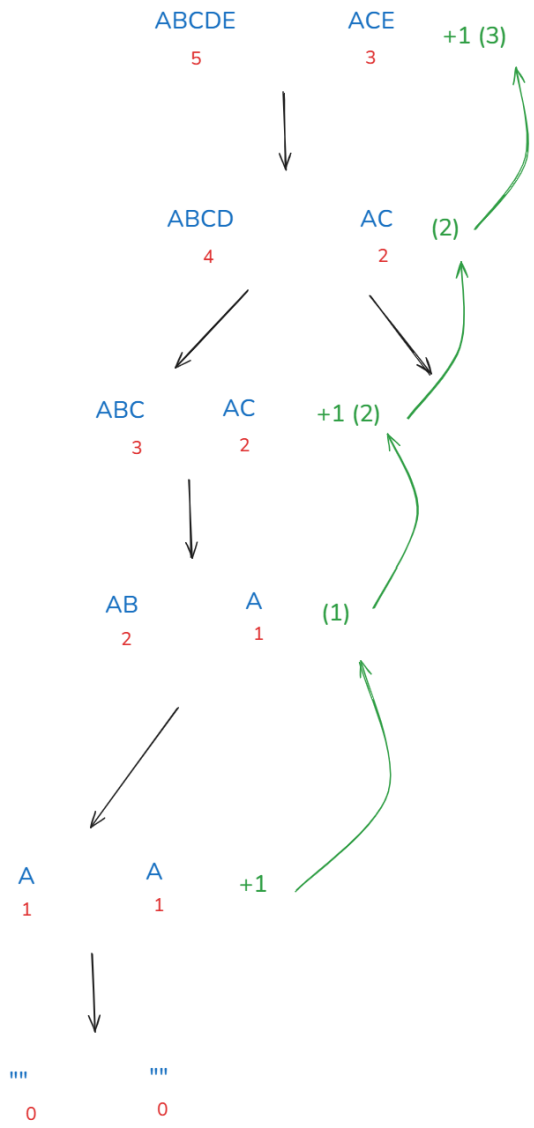
Como Memorizar?



text1 = "abcde"
text2 = "ace"

	0	1	2	3
0	-1	-1	-1	-1
1	-1	1	-1	-1
2	-1	1	-1	-1
3	-1	1	2	-1
4	-1	1	2	-1
5	-1	-1	-1	3

Memo[1][1] = 1
Memo[2][1] = 1
Memo[3][2] = 2
Lê do Memo[2][1] = 1
Memo[3][1] = 1
Memo[4][1] = 1
Memo[4][2] = 2
Memo[5][3] = 3



```python
def longestCommonSubsequence(self, text1: str, text2: str) -> int:

        memo = [ [-1]*(size2+1) for _ in range(size1+1) ]

        def pd(size1, size2):
            if size1==0 or size2==0:
                return 0
            if memo[size1][size2]!=-1:
                return memo[size1][size2]

            if text1[size1-1]==text2[size2-1]:
                memo[size1][size2] = 1+pd(size1-1, size2-1)
            else:
                memo[size1][size2] = max(pd(size1-1, size2), pd(size1, size2-1))

            return memo[size1][size2]

        return pd(len(text1), len(text2))
```

Coin Change II - LeetCode 518

https://leetcode.com/problems/coin-change-ii

Você recebe um array de inteiros representando moedas e um inteiro representando um valor total.

Retorne o número de combinações que somam esse valor.

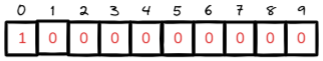
Se não for possível esse valor com nenhuma combinação de moedas, retorne 0.

Você pode assumir que possui um número infinito de cada tipo de moeda.

Como Resolver?

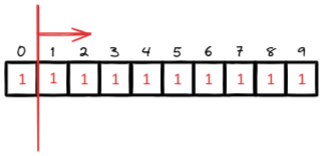
Input: 1, 2, 5

Amount: 9



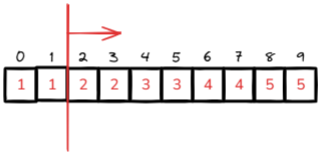
Consigo Obter Zero, com Zero moedas (1 forma)

Moeda 1



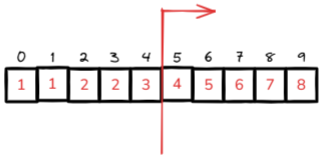
Consigo Obter valores de 1 a 9, com moedas de valor 1 de apenas (1 forma)

Moeda 2

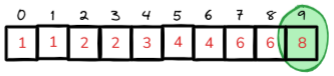


memo[i] += memo[i-coin]

Moeda 5



memo[i] += memo[i-coin]



Resultado para valor 9 com as moedas 1, 2 e 5:
8 Formas Diferentes