

# Tutorial: Maximum Subarray Sum

Arthur Andrade D'Oliveira

O problema pode ser resolvido eficientemente utilizando uma abordagem de **programação dinâmica**, comumente conhecida como **Algoritmo de Kadane**. A ideia central é calcular, para cada posição  $i$  da sequência, a maior soma de uma subsequência contínua que termina exatamente naquela posição.

## Modelagem

Dada uma sequência de entrada  $A$  de tamanho  $n$  (com índices de 0 a  $n - 1$ ), queremos encontrar a maior soma possível  $\sum_{k=i}^j A[k]$  para quaisquer índices  $0 \leq i \leq j < n$ .

O algoritmo de Kadane resolve isso de forma linear, mantendo o controle da melhor subsequência encontrada até o momento.

## Definição da DP

Definimos  $dp[i]$  como o valor da **maior soma de uma subsequência contínua que termina no índice  $i$** .

## Função de Transição

Para calcular o valor de  $dp[i]$ , temos duas escolhas:

A subsequência máxima que termina em  $i$  é formada **apenas pelo elemento  $A[i]$** . Isso ocorre se a soma da subsequência anterior ( $dp[i - 1]$ ) for negativa, pois estendê-la apenas diminuiria o valor.

A subsequência máxima que termina em  $i$  é uma **extensão da subsequência máxima que terminava em  $i - 1$** , adicionando  $A[i]$ . A soma seria  $dp[i - 1] + A[i]$ .

Selecionamos a maior dessas duas opções. Portanto, a função de transição é:

$$dp[i] = \max(A[i], \quad dp[i - 1] + A[i])$$

## Casos Base

A primeira subsequência possível (que termina no índice 0) deve obrigatoriamente conter  $A[0]$ . O caso base é:

$$dp[0] = A[0]$$

## Resposta Final

É importante notar que  $dp[n - 1]$  é apenas a maior soma terminando na última posição, o que não é necessariamente a resposta final do problema.

A subsequência de soma máxima pode terminar em qualquer índice  $i$ . Portanto, a resposta final é o **valor máximo encontrado em todo o array  $dp$** .

$$\max_{0 \leq i < n} (dp[i])$$

## Complexidade

- **Tempo:**  $O(n)$ , pois percorremos a sequência de entrada uma única vez para preencher o array  $dp$ . Encontrar o máximo do array  $dp$  também leva  $O(n)$ , o que pode ser feito simultaneamente.
- **Espaço:**  $O(n)$  para armazenar o array  $dp$ .

**Otimização de Espaço:** Note que o cálculo de  $dp[i]$  depende apenas de  $dp[i - 1]$ . Podemos otimizar o espaço para  $O(1)$  mantendo apenas duas variáveis: uma para a "soma máxima terminando aqui" (equivalente a  $dp[i - 1]$ ) e outra para a "soma máxima global" (a resposta final).