

# Tutorial: Maior subsequência Crescente II

Arthur Andrade D'Oliveira

## 1 Solução do Problema

O problema de encontrar a maior subsequência crescente (*Longest Increasing Subsequence* - LIS) possui uma solução clássica em  $O(N^2)$ , mas quando  $N$  é grande, podemos otimizá-la para  $O(N \log N)$  combinando **programação dinâmica com busca binária**. A grande sacada dessa otimização é mudar a perspectiva: em vez de guardar o tamanho da maior subsequência que termina em um índice, guardamos o **menor valor final** possível para uma subsequência de um determinado comprimento.

### 1.1 Definição do Subproblema

Seja  $a$  o nosso vetor de números inteiros de tamanho  $N$ . Definimos o nosso estado da programação dinâmica como:

$d[l]$  = o menor valor que encerra uma subsequência crescente de comprimento exato  $l$ .

O vetor  $d$  terá uma propriedade muito importante: ele estará sempre rigorosamente ordenado de forma crescente. A resposta final será o maior  $l$  tal que  $d[l] \neq \infty$ .

### 1.2 Função de Transição

Para calcular  $d[l]$ , iteramos sobre cada elemento  $a[i]$  da nossa sequência original. Como queremos que os elementos finais de cada comprimento sejam os menores possíveis, tentamos usar  $a[i]$  para melhorar o nosso vetor  $d$ .

Como  $d$  está ordenado, podemos usar **busca binária** (`upper_bound` ou `lower_bound`) para encontrar rapidamente a posição  $l$  onde  $a[i]$  deve ser inserido. Se  $a[i]$  for maior que o final de uma subsequência de tamanho  $l - 1$  ( $d[l - 1] < a[i]$ ) e, ao mesmo tempo, puder substituir um final pior de tamanho  $l$  ( $a[i] < d[l]$ ), nós o atualizamos:

$$d[l] = a[i]$$

Isso significa que encontramos uma subsequência crescente de comprimento  $l$  que termina em  $a[i]$ , e esse final é melhor (menor) do que o que tínhamos registrado anteriormente.

### 1.3 Casos Base

Para que a lógica de busca e as comparações iniciais funcionem sem estourar os limites, inicializamos o vetor  $d$  de tamanho  $N + 1$  da seguinte forma:

$$\begin{aligned} d[0] &= -\infty \\ d[l] &= \infty \quad \text{para todo } 1 \leq l \leq N \end{aligned}$$

Isso representa que uma subsequência de tamanho 0 termina em um valor infinitamente pequeno, enquanto os outros comprimentos ainda não foram alcançados.

## 1.4 Recuperação da Decomposição (Elementos da subsequência)

Para recuperar os elementos exatos da LIS nesta abordagem otimizada, o vetor  $d$  sozinho não basta (pois ele pode conter uma mistura de elementos de diferentes subsequências válidas ao longo do tempo). Precisamos de duas estruturas de rastreamento:

- $pos[l]$ : armazena o **índice original** do elemento que atualmente ocupa  $d[l]$ .
- $p[i]$ : armazena o predecessor do elemento de índice  $i$ , assim como na versão  $O(N^2)$ .

Durante a transição, sempre que atualizamos  $d[l] = a[i]$ , nós também registramos que esse elemento se encontra no índice  $i$  original fazendo  $pos[l] = i$ . Ao mesmo tempo, sabemos que o elemento imediatamente anterior a ele na subsequência é o elemento que termina o comprimento  $l-1$ . Logo, conectamos o predecessor:  $p[i] = pos[l-1]$ .

Ao final de todas as iterações, sabemos que **ans** é o comprimento máximo alcançado. O índice do último elemento dessa subsequência estará em  $cur = pos[ans]$ . A partir desse  $cur$ , reconstruímos a sequência retrocedendo  $cur = p[cur]$  até atingir  $-1$ , e por fim invertemos a lista para exibi-la na ordem correta.