

# Tutorial: Estouro

Leetcode 312

## 1 Solução do Problema

O problema de maximizar a energia ao estourar balões pode ser resolvido de forma eficiente por meio de **programação dinâmica em intervalos**. A grande sacada para modelar este problema é pensar no processo de trás para frente: em vez de tentar adivinhar qual balão estourar primeiro, o que muda constantemente os vizinhos dos balões restantes, escolhemos qual será o **último balão** a ser estourado em um determinado subintervalo.

Para lidar com as bordas da fileira sem precisar de múltiplos `ifs`, adicionamos balões virtuais com valor 1 nas extremidades.

### 1.1 Definição do Subproblema

Seja a nossa fileira de balões original de tamanho  $N$ . Criamos um novo vetor  $B$  de tamanho  $N + 2$ , onde  $B[0] = 1$ ,  $B[N + 1] = 1$ , e as posições de 1 a  $N$  recebem os valores fornecidos.

Definimos o nosso estado da programação dinâmica como:

$dp[i][j]$  = a quantidade máxima de energia obtida ao estourar todos os balões no intervalo  $[i, j]$ .

Assim,  $dp[1][N]$  representará a nossa resposta ótima, correspondente ao intervalo que engloba todos os balões reais.

### 1.2 Função de Transição

Para calcular  $dp[i][j]$ , vamos iterar por um pivô  $k$  (com  $i \leq k \leq j$ ) que representa o **último balão** a ser estourado dentro desse subintervalo.

Como  $k$  é o último a sobreviver entre  $i$  e  $j$ , todos os outros balões internos já foram estourados. Consequentemente, no exato momento da explosão de  $k$ , seus vizinhos diretos garantidamente serão  $B[i - 1]$  e  $B[j + 1]$ . A energia liberada por esse estopim final será  $B[i - 1] \times B[k] \times B[j + 1]$ .

O valor total do intervalo será a soma da energia desse balão  $k$  com as energias máximas dos subproblemas que o antecederam na ordem cronológica de explosão: o intervalo à esquerda ( $i$  até  $k - 1$ ) e o intervalo à direita ( $k + 1$  até  $j$ ). Portanto:

$$dp[i][j] = \max_{i \leq k \leq j} (dp[i][k - 1] + B[i - 1] \cdot B[k] \cdot B[j + 1] + dp[k + 1][j])$$

### 1.3 Casos Base

Os casos base da nossa recursão (ou inicialização da matriz) ocorrem para intervalos vazios, ou seja, quando o índice de início ultrapassa o índice de fim ( $i > j$ ). Nesses casos, como não há balões para estourar, a energia liberada é naturalmente nula:

$$dp[i][j] = 0 \quad \text{para todo } i > j$$

Como a implementação iterativa inicializa a matriz completa com zeros, essa condição já é suprida por padrão.

## 1.4 Recuperação da Decomposição (Ordem das explosões)

Para além da energia máxima  $dp[1][N]$ , frequentemente queremos também a sequência de explosões que atinge esse valor. Para isso, mantemos uma matriz auxiliar  $cut[i][j]$  durante a computação, que armazena o índice  $k$  responsável pelo maior ganho em  $dp[i][j]$ .

Após preencher  $dp$  e  $cut$ , criamos uma função recursiva para reconstruir a resposta. Ao chamarmos a função para o intervalo  $[1, N]$ , descobrimos o pivô  $k = cut[1][N]$ . Sabemos que  $k$  é o *último* balão estourado. Em seguida, chamamos recursivamente para os subintervalos  $[1, k - 1]$  e  $[k + 1, N]$ . Ao empilharmos esses balões, obtemos a sequência cronológica invertida; basta inverter a lista no final para obtermos a sequência exata das explosões.