Tutorial: Bolhas

UVA - 11495

O problema consiste em determinar o vencedor de um jogo baseado em trocas de elementos adjacentes de uma permutação, o que se traduz em calcular a paridade do número de inversões da sequência inicial.

Representação do estado

- Uma **inversão** ocorre quando um elemento maior aparece antes de um elemento menor na sequência.
- Cada movimento válido no jogo (trocar um par de elementos adjacentes fora de ordem) reduz o número total de inversões em exatamente 1.
- Como o jogo termina quando a sequência está ordenada (zero inversões) e Marcelo faz o primeiro movimento, a vitória depende exclusivamente da paridade das inversões: se for ímpar, Marcelo fará o último movimento e vencerá; se for par, Carlos vencerá.

Estratégia de solução usando Divisão e Conquista (Merge Sort)

Para contar o número de inversões de forma eficiente, evitamos a contagem trivial de comparar todos os pares (que custaria $O(N^2)$) e utilizamos uma adaptação do algoritmo *Merge Sort*.

1. Leia o tamanho da sequência (N) e armazene a permutação inicial em um vetor.
2. Crie uma função recursiva baseada no *Merge Sort* que divide o vetor ao meio até que os subvetores tenham tamanho 1 (onde o número de inversões é 0).
3. Na etapa de conquista (*merge*):
    - Mantenha dois ponteiros, um para o subvetor da esquerda e outro para o subvetor da direita.
    - Compare os elementos apontados. Se o elemento da direita for menor que o da esquerda, ele precisará saltar todos os elementos restantes do subvetor da esquerda para ser ordenado.
    - Nesse caso, adicione ao contador de inversões a quantidade exata de elementos restantes no subvetor da esquerda.
    - Copie o menor elemento para a posição correta do vetor principal para mantê-lo ordenado.
4. Após a execução de toda a recursão, avalie o número total de inversões contabilizadas.
5. Se o número de inversões for ímpar, imprima "Marcelo". Caso contrário, imprima "Carlos".

Complexidade

- **Espacial:** $O(N)$, devido à criação de vetores temporários auxiliares durante as etapas de intercalação do *Merge Sort*, além de $O(\log N)$ referente ao espaço utilizado pela pilha de chamadas recursivas.
- **Temporal:** $O(N \log N)$. O algoritmo divide a sequência de tamanho N pela metade $\log N$ vezes. Em cada um desses níveis de recursão, a etapa de intercalação percorre os subvetores de forma linear, custando globalmente $O(N)$ por nível.