

Algoritmos

Pedro Hokama

- [cirs] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [timr] Algorithms Illuminated Series, Tim Roughgarden
- Desmistificando Algoritmos, Thomas H. Cormen.
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
<https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623ckIH7V>
<https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZ17A5HEK6>
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420
Qualquer erro é de minha responsabilidade.

1 / 17

2 / 17

Divisão e Conquista: O paradigma

O paradigma de divisão e conquista tem três etapas:

- 1 **Dividir** o problema em subproblemas menores.
- 2 **Conquistar** os subproblemas (normalmente de forma recursiva).
- 3 **Combinar** a solução dos subproblemas para encontrar uma solução para o problema original.

Diferentes algoritmos tem a complexidade diferente em cada uma das fases, por exemplo, no MergeSort a divisão é trivial mas a combinação exige esforço. Já no QuickSort a divisão é bastante elaborada mas a combinação é trivial.

3 / 17

O Problema

- Suponha que você e um amigo escolheram 10 filmes que ambos assistiram.
- Cada um ordenou esses 10 filmes em ordem de preferência.
- Queremos saber a compatibilidade entre essas duas listas, e verificar se essa amizade pode dar certo.
- Um serviço de streaming poderia usar essa comparação para verificar usuários que tem gostos parecidos para fazer recomendações.

4 / 17

Problema do Número de Inversões

Problema do Número de Inversões

Dado um arranjo A contendo n inteiros em uma ordem arbitrária, encontrar o número total de inversões, ou seja, o número de pares (i, j) de índices $1 < i, j < n$ tais que $i < j$ e $A[i] > A[j]$.

Considere o vetor seguinte vetor

$$A = (1, 3, 5, 2, 4, 6)$$

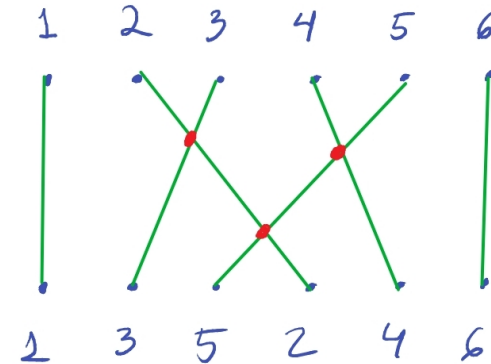
qual o número de inversões?

- os elementos 3 e 2, então os índices (2, 4) formam uma inversão
- os elementos 5 e 2, então os índices (3, 4) formam uma inversão
- os elementos 5 e 4, então os índices (3, 5) formam uma inversão

5 / 17

Problema do Número de Inversões

$$A = (1, 3, 5, 2, 4, 6)$$



6 / 17

Problema do Número de Inversões

Qual o número máximo de inversões em um vetor de tamanho 6?

- a 6
- b 15
- c 21
- d 36
- e 64

7 / 17

Problema do Número de Inversões

Qual o número máximo de inversões em um vetor de tamanho n ?

- O máximo de inversões acontece se todos os pares de índice estiverem invertidos.
- Ou seja, dos n elementos, quaisquer dois que escolhermos estará invertido.

$$\binom{n}{2}$$

- Lembrando:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- Então

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n \cdot (n-1) \cdot (n-2)!}{2!(n-2)!} = \frac{n^2 - n}{2}$$

8 / 17

Uma ideia ingenua

Como seria uma solução força bruta?

Algoritmo 1: ContarInversões

Entrada: Um vetor A de tamanho n

Saída: O número de inversões

```
1 t = 0;
2 para i de 1 até n - 1 faça
3   para j de i + 1 até n faça
4     se A[i] > A[j] então
5       t++;
6 devolva t;
```

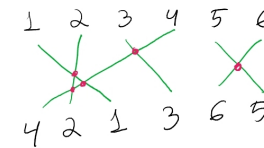
Qual a complexidade desse algoritmo?

- a $\log n$
- b n
- c $n \log n$
- d n^2
- e n^3

Podemos fazer melhor?
Yep!

Uma algoritmo de divisão e conquista

(4, 2, 1, 3, 6, 5)



Valor verdadeiro: 5 inversões

Divisão

(4, 2, 1) (3, 6, 5)

Conquista

3 inversões 1 inversão

Combinação?

Como contar as inversões entre as metades?

9 / 17

10 / 17

Uma algoritmo de divisão e conquista

- Ideia: dividir o vetor em 2 metades, contar o número de inversões.
- Mas ainda faltará as inversões entre os elementos da primeira e da segunda metade.
- Podemos então contar essas inversões?
- Para isso vamos então classificar as inversões em três tipos:
 - ▶ **Esquerda:** se $i, j \leq n/2$
 - ▶ **Direita:** se $i, j > n/2$
 - ▶ **Split:** se $i \leq n/2 < j$
- Nova ideia: contar as inversões esquerda, direita e split.

11 / 17

Uma algoritmo de divisão e conquista

Uma ideia (ainda incompleta) seria:

Algoritmo 2: Count

Entrada: Um arranjo A de comprimento n

Saída: O número de inversões

```
1 se  $n \leq 1$  então devolva 0;
2 senão
3    $x = \text{Count}(\text{Primeira metade de } A, n/2);$ 
4    $y = \text{Count}(\text{Segunda metade de } A, n/2);$ 
5    $z = \text{ContaSplit}(A, n);$ 
6 devolva  $x + y + z;$ 
```

- Se conseguirmos contar o número de inversões split em tempo linear $O(n)$ a árvore de recursão fica idêntica ao MergeSort.
- A complexidade total ficaria $O(n \log n)$
- O número de inversões de tipo split é $O(n^2)$. Será que podemos contar um número quadrático de coisas em tempo linear? 🤔
- Yep!

12 / 17

- Considere que Count conta o número de inversões, mas também ordena o vetor.
- E vamos dar uma olhada na função Merge do MergeSort

Algoritmo 3: Merge

Entrada: B e C arranjos ordenados com $m/2$

Saída: Arranjo D de tamanho m com os mesmos elementos de B e C mas ordenados

```

1  $i = 1; j = 1;$ 
2 para  $k$  de 1 até  $m$  faça
3   se  $B[i] < C[j]$  então
4      $D[k] = B[i];$ 
5      $i ++;$ 
6   senão
7      $D[k] = C[j];$ 
8      $j ++;$ 
9 devolva  $D;$ 

```

13 / 17

- Se não houver inversões do tipo split, o vai acontecer na hora de copiar B e C ?
- Nesse caso B seria copiado inteiramente antes de C .
- O que acontece significa, em número de inversões, quando copiamos um elemento do vetor C ?
- Isso significa que o elemento $C[j]$ copiado está em inversão do tipo split com todos os valores que ainda não foram copiados de B .
- Isso significa $|B| - i + 1$ elementos. (Se indexar em 0 não precisa do "+1")

14 / 17

Modificando o Merge para Contar Inversões Splits

Algoritmo 4: Merge

Entrada: B e C arranjos ordenados com $m/2$

Saída: Arranjo D de tamanho m com os mesmos elementos de B e C mas ordenados

```

1  $i = 1; j = 1;$ 
2 para  $k$  de 1 até  $m$  faça
3   se  $B[i] < C[j]$  então
4      $D[k] = B[i];$ 
5      $i ++;$ 
6   senão
7      $D[k] = C[j];$ 
8      $j ++;$ 
9 devolva  $D;$ 

```

Algoritmo 5: MergeCountSplit

Entrada: B e C arranjos ordenados com $m/2$

Saída: Arranjo D de tamanho m com os elementos de B e C mas ordenados, e o número de inversões Splits

```

1  $i = 1; j = 1; t = 0;$ 
2 para  $k$  de 1 até  $m$  faça
3   se  $B[i] < C[j]$  então
4      $D[k] = B[i];$ 
5      $i ++;$ 
6   senão
7      $D[k] = C[j];$ 
8      $j ++; t = t + (m/2 - i + 1);$ 
9 devolva  $(D, t);$ 

```

15 / 17

Uma algoritmo de divisão e conquista (versão completa)

Algoritmo 6: SortCount

Entrada: Um arranjo A , o comprimento do arranjo n

Saída: Um arranjo com os mesmos elementos de A porém ordenados, O número de inversões

```

1 se  $n \leq 1$  então devolva  $(A, 0);$ 
2 senão
3    $(B, x) = \text{SortCount}(\text{Primeira metade de } A, n/2);$ 
4    $(C, y) = \text{SortCount}(\text{Segunda metade de } A, n/2);$ 
5    $(D, z) = \text{MergeCountSplit}(B, C, n);$ 
6 devolva  $(D, x + y + z);$ 

```

16 / 17

- Assim como no MergeSort, a árvore de recursão de SortCount tem $\log n$ níveis e cada nível executa $O(n)$ operações, então a complexidade total de SortCount é

$$O(n \log n)$$

- Portanto muito melhor que a versão força bruta!