

QuickSort

*Algoritmos e Programação II
(slides baseados na apostila do Prof Fábio Viduani)*

Ordenação por separação (QuickSort)

- 0 Operação básica em Computação
- 0 Métodos mais eficientes de ordenação
- 0 Ordenação com recursão
- 0 Dividir para conquistar

Problema da separação

0 Problema:

0 Rearranjar o vetor $v[p..r]$ de modo que tenhamos

$$v[p..q] \leq v[q+1..r]$$

para algum q em $p..r-1$. A expressão $v[p..q] \leq v[q+1..r]$ significa que

$v[i] \leq v[j]$ para todo $i, p \leq i \leq q$, e todo $j, q < j \leq r$.

p=0

1

2

3

4

5

6

r=7

4	1	8	6	3	7	5	2
---	---	---	---	---	---	---	---

p=0

1

2

3

4

5

6

r=7

--	--	--	--	--	--	--	--

p=0

1

2

3

4

5

6

r=7

--	--	--	--	--	--	--	--

```
int separa(int p, int r, int v[MAX])
```

```
{
```

```
    int x, i, j;
```

p=0	1	2	3	4	5	6	r=7
4	1	8	6	3	7	5	2

```
    x = v[p];
```

```
    i = p - 1;
```

```
    j = r + 1;
```

```
    while (1) {
```

```
        do {
```

```
            j--;
```

```
        } while (v[j] > x);
```

```
        do {
```

```
            i++;
```

```
        } while (v[i] < x);
```

```
        if (i < j)
```

```
            troca(&v[i], &v[j]);
```

```
        else
```

```
            return j;
```

```
}
```

```
}
```

p=0	1	2	3	4	5	6	r=7

```
int separa(int p, int r, int v[MAX])
{
    int x, i, j;

    x = v[p];
    i = p - 1;
    j = r + 1;
    while (1) {
        do {
            j--;
        } while (v[j] > x);
        do {
            i++;
        } while (v[i] < x);
        if (i < j)
            troca(&v[i], &v[j]);
        else
            return j;
    }
}
```

Tempo de execução de pior caso é proporcional a $r - p + 1$, isto é, proporcional ao número de elementos do vetor.

Ordenação por separação

Princípio

Para ordenar um vetor $A[p..r]$:

- 0 **Divisão:** o vetor é particionado em dois subvetores não-vazios $A[p..q]$ e $A[q+1..r]$ tal que cada elemento de $A[p..q]$ é menor ou igual a cada elemento de $A[q+1..r]$.
 - 0 **Conquista:** os dois subvetores são ordenados por chamadas recursivas ao quicksort.
 - 0 **Combina:** nada a ser feito.
- Chave** Particionamento em tempo linear

```
/* Recebe um vetor v[p..r-1] e o rearranja em ordem crescente */
void quicksort(int p, int r, int v[MAX])
{
    int q;

    if (p < r) {
        q = separa(p, r, v);
        quicksort(p, q, v);
        quicksort(q+1, r, v);
    }
}
```

- Para ordenar um vetor $v[0..n - 1]$ basta chamar a função **quicksort** com os seguintes argumentos:

quicksort($0, n - 1, v$)

Ordenação por separação

- 0 Desempenho da função **quicksort** quando queremos ordenar um vetor $v[0..n - 1]$
 - 0 Proporcional ao número de comparações realizadas entre os elementos do vetor
 - 0 Se o índice devolvido pela função **separa** sempre tiver valor mais ou menos médio de p e r , então o número de comparações será aproximadamente $n \log_2 n$.
 - 0 Caso contrário, o número de comparações será da ordem de n^2
 - 0 Portanto, o consumo de tempo de pior caso da ordenação por separação não é melhor que o dos métodos elementares
 - 0 Felizmente, o pior caso para a ordenação por separação é **raro**. Dessa forma, o consumo de tempo *médio* da função **quicksort** é proporcional a $n \log_2 n$.

QUICKSORT

```
/* Recebe um vetor v[p..r-1] e o rearranja em ordem crescente */  
void quicksort(int p, int r, int v[MAX])  
{  
    int q;  
  
    if (p < r) {  
        q = separa(p, r, v);  
        quicksort(p, q, v);  
        quicksort(q+1, r, v);  
    }  
}
```

Melhor Caso

1 – O(n)

2 – T(n/2)

3 – T(n/2)

$$T(n) = \begin{cases} 1, & \text{se } n = 1 \\ 2T(n/2) + n, & \text{se } n > 1 \end{cases}$$

Recursão já resolvida → O(n. log n)

QUICKSORT

```
/* Recebe um vetor v[p..r-1] e o rearranja em ordem crescente */  
void quicksort(int p, int r, int v[MAX])  
{  
    int q;  
  
    if (p < r) {  
        q = separa(p, r, v);  
        quicksort(p, q, v);  
        quicksort(q+1, r, v);  
    }  
}
```

Pior Caso

1 – O(n)
2 – T(1) = 1
3 – T(n-1)

$$\begin{aligned} T(n) = & \quad 1, & \text{se } n = 1 \\ & T(n-1) + n & \text{, se } n > 1 \end{aligned}$$

Vamos resolver !!!! → O (??)

QUICKSORT

```
/* Recebe um vetor v[p..r-1] e o rearranja em ordem crescente */
void quicksort(int p, int r, int v[MAX])
{
    int q;                                Pior Caso

    if (p < r) {
        q = separa(p, r, v);                1 - O(n)
        quicksort(p, q, v);                  2 - T(1) = 1
        quicksort(q+1, r, v);               3 - T(n-1)
    }
}
```

$$\begin{aligned} T(n) = & \quad 1, & \text{se } n = 1 \\ & T(n-1) + n & \text{, se } n > 1 \end{aligned}$$

$$T(n) = O(n^2)$$

Árvore de Recursão no Pior Caso

