

- Quick Sort (Buckets)

↳ Ordenação por troca de partição

Ideia: dividir e conquistar

→ Um elemento é escolhido como pivô

→ "Particionar": os dados são rearranjados

① Valores menores que o pivô são colocados antes dele

② Valores maiores que o pivô são colocados depois dele

→ Recursivamente ordena os duas partções

- Performance

- Melhor caso: $O(N \log N)$

- Pior caso: $O(N^2)$ raro

- Estável: não altera a ordem dos dados iguais

- Desvantagens: Como escolher o pivô?

• Código (c)

```

1. void quickSort (int *V, int inicio, int fim)
2. {
3.     int pivo;
4.     if (fim > inicio) {
5.         pivo = particiona (V, inicio, fim);
6.         quickSort (V, inicio, pivo - 1);
7.         quickSort (V, pivo + 1, fim);
8.     }
9. }
  
```

separa os dados em duas partições

chama a função para os dois metados

9. int particiona (int *V, int inicio, int final) {

```

10. int esq, dir, pivo, aux;
11. esq = inicio;
12. dir = final;
13. pivo = V[inicio];
14. while (esq < dir) {
15.     while (V[esq] <= pivo) { } avança posição da
16.         esq++;
17.     }
18.     while (V[dir] > pivo) { } recua posição da
19.         dir--;
20.     }
  
```

esquerda

direita

```

21.     if( esq < dir ) {
22.         aux = V[esq];
23.         V[esq] = V[dir];
24.         V[dir] = aux;
25.     }
26. }
27. V[inicio] = V[dir];
28. V[dir] = pino;
29. return (dir);
30. }

```

• P seu dódigo

→ algoritmo usa duas funções

1. Quicksort: divide os dados em vetores cada vez menores
2. Particiona: elege um pinô e particiona de maneira que:
 - todos os elementos menores que o pinô estão antes dele
 - todos os elementos maiores que o pinô estão depois dele

* Quick Sort (V , Início, Fim)

- Se ($\text{Início} < \text{Fim}$) então:

- $\text{pivo} = \text{particiona}(V, \text{Início}, \text{Fim})$
- $\text{quickSort}(V, \text{Início}, \text{pivo}-1)$
- $\text{quickSort}(V, \text{pivo}+1, \text{Fim})$

* Particiona (V , Início, Fim)

- $\text{esq} = \text{Início}$
- $\text{dir} = \text{Fim}$
- $\text{pivo} = V[\text{Início}]$

Erga esq < dir foga:

Erga $V[\text{esq}] \leq \text{pivo}$ E $\text{esq} \leq \text{final}$, foga:
incrementa Esq

Erga $V[\text{dir}] > \text{pivo}$ E $\text{dir} \geq \text{início}$, foga:
decrementa Dir

Se esq < dir então:

troca $V[\text{esq}] \leftrightarrow V[\text{dir}]$

troca $V[\text{dir}]$ com $V[\text{início}]$
retorna dir

Observações:

desvantagens → não é algoritmo estável

→ Como escolher pivô?

- diferentes bordagens
- particionamento não é balanceado.

No pior caso, o pivô divide o vetor de N elementos em dois:

- * $N-1$ elementos
- * 0 elementos

• Quando isso acontece em toda recursão, temos $O(N^2)$

Vantagem

→ Costuma ser a melhor opção prática para ordenações de grandes conjuntos de dados

Exemplo:

0	1	2	3	4	5	6
23	4	67	-8	90	54	21

Vetor Inicial

* particiona($v, 0, 6$)

$p_{\text{Nov}}, \text{ESQ}$

23

4 67

-8

90

54

DIR

21

- $\text{ESQ} \leq \text{Pivo}$:
 $\text{ESQ}++$

Pivo
23

ESQ
4

67
ESQ

-8

90

54

DIR

21

- $\text{ESQ} \leq \text{Pivo}$?

Pivo
23

4

67
ESQ

-8

90

54

DIR

21

- $\text{ESQ} > \text{DIR}$

Comparar DIR

Pivo
23

4

67
ESQ

-8

90

54

DIR

21

- $\text{DIR} < \text{Pivo}$

Trocar ESQ e

DIR

Pivo
23

4

21
ESQ

-8

90

54

DIR

67

- $\text{ESQ} < \text{DIR}$

continua while

Pivo
23

4

21
ESQ

-8

90

54

DIR

67

$\text{ESQ} \leq \text{Pivo}$

$\text{ESQ}++$

Pivo
23

4

21
ESQ

-8

90

54

DIR

67

$\text{ESQ} \leq \text{Pivo}$

$\text{ESQ}++$

Pivo
23

4

21
ESQ

-8

90
ESQ

54

DIR

67

$\text{ESQ} > \text{DIR}$

Comparar dir

23

4

21
ESQ

-8

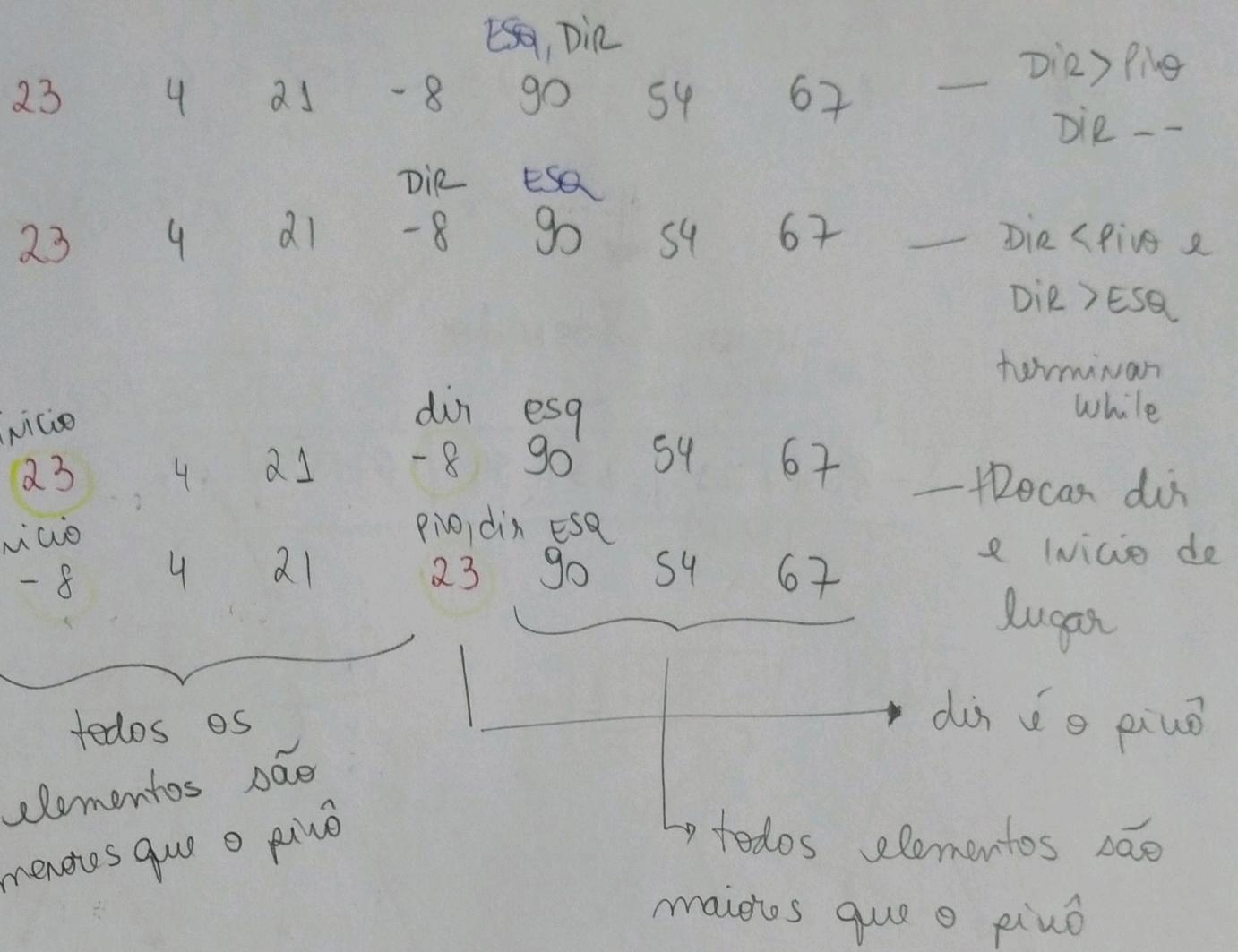
90
ESQ

54
DIR

67

$\text{Dir} > \text{Pivo}$

$\text{Dir}+-$



Particiona ($v_1, 0, 6$)

23	4	67	-8	90	54	21
-8	4	21	23	90	54	67

particiona ($v_1, 0, 2$)

-8	4	21
-8	4	21

pivô

particiona ($v_1, 4, 6$)

90	54	67
67	54	90

pivô

-8

particiona ($v_1, 1, 2$)

4	21
4	21

pivô

particiona ($v_1, 4, 5$)

67	54
54	67

90

pivô

-8 4 21 23 54 67 90



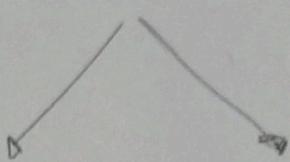
Vektor Oordenode

$[-8|4|21|23|54|67|90|]$

DEBVS

45	36	2	8	6	1	23	2	2	50
0	1	2	3	4	5	6	7	8	9

$$\text{meis} = (0+9)/2 = 4$$



45	36	2	8	6	1	23	2	2	50
0	1	2	3	4	5	6	7	8	9

1	23	2	2	50
5	6	7	8	9

$$(9+5)/2 = 7$$

45	36	2	8	0
0	1	2	3	4

$$(2+0)/2 = 1$$

$$4+3=7/2=3$$

45	36
0	1

$$1/2=0$$

45	36
0	1

2
2

8
3

0
4

5	23
5	6

2
7

2
8

10
9

$$17/2 = 8,5 = 8$$

$$12/2 = 6$$

1	23
5	6

1	23
5	6

1	23
5	6