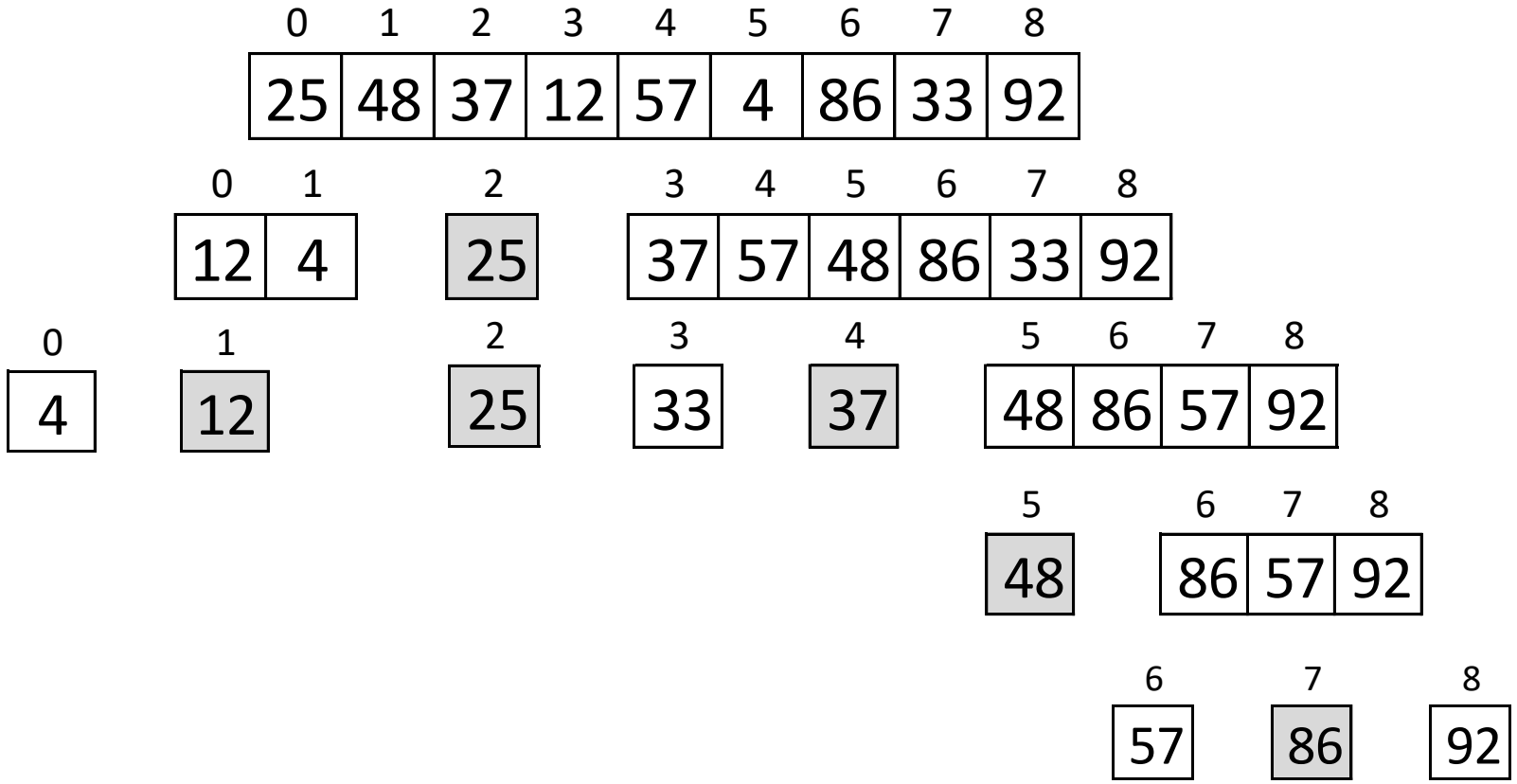
Quicksort

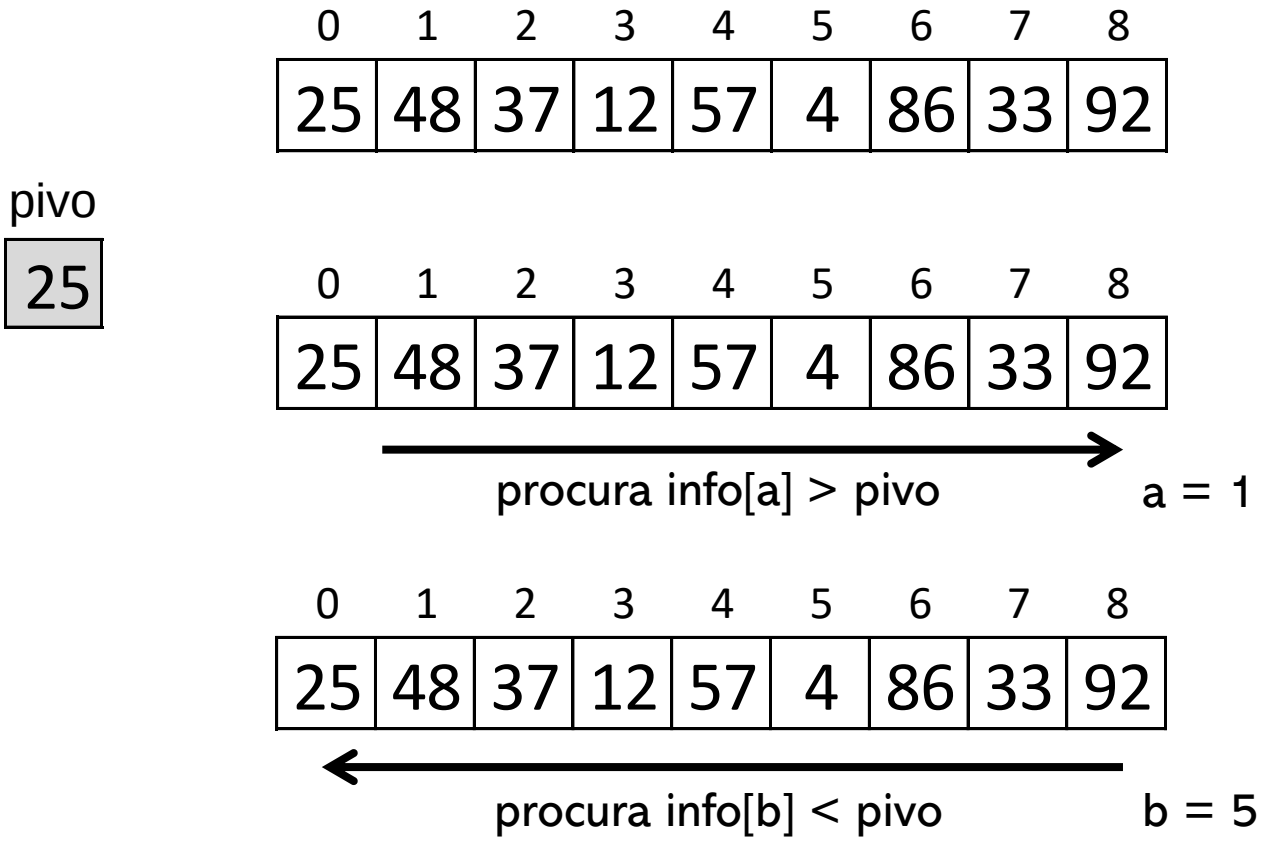
Definições

- Escolhe-se um elemento arbitrário, denominado de “pivô”
- Organiza-se o vetor de forma que:
  - Os elementos menores que o pivô fiquem à sua esquerda
  - Os elementos maiores que o pivô fiquem à sua direita
- Após este procedimento, o pivô estará ordenado
- Chamar recursivamente o algoritmo para organizar os sub-vetores à esquerda e à direita do pivô
- Prosseguir, até que os vetores tenham 0 ou 1 elemento.

Exemplo


0 1 2 3 4 5 6 7 8
25 48 37 12 57 4 86 33 92
0 1 2 3 4 5 6 7 8
12 4 25 37 57 48 86 33 92
0 1 2 3 4 5 6 7 8
4 12 25 33 37 48 86 57 92
5 6 7 8
48 86 57 92
6 7 8
57 86 92

Exemplo


0 1 2 3 4 5 6 7 8
25 48 37 12 57 4 86 33 92
pivo
25
0 1 2 3 4 5 6 7 8
25 48 37 12 57 4 86 33 92
procura info[a] > pivo
a = 1
0 1 2 3 4 5 6 7 8
25 48 37 12 57 4 86 33 92
procura info[b] < pivo
b = 5

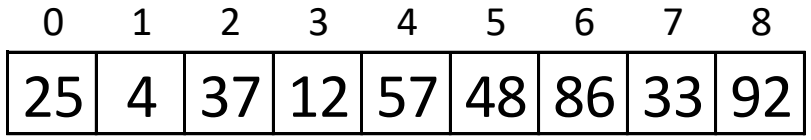
Exemplo

0	1	2	3	4	5	6	7	8
25	48	37	12	57	4	86	33	92

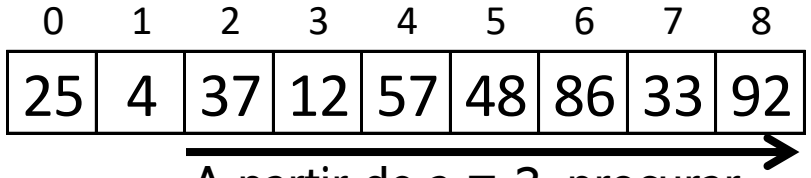
a = 1
b = 5

0	1	2	3	4	5	6	7	8
25	4	37	12	57	48	86	33	92

Exemplo

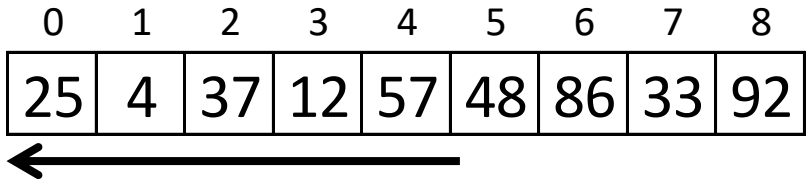


a = 1
b = 5



A partir de a = 2, procurar
info[a] > pivo

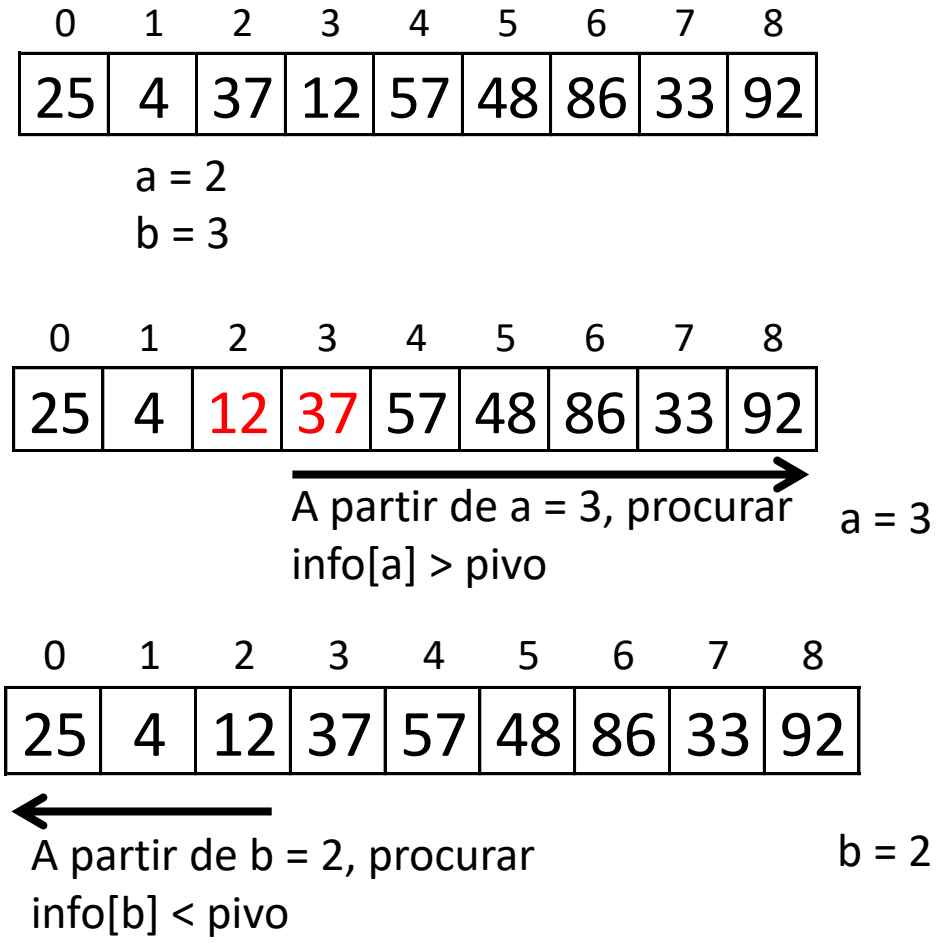
a = 2



A partir de b = 4, procurar
info[b] < pivo

b = 3

Exemplo


0 1 2 3 4 5 6 7 8
25 4 37 12 57 48 86 33 92
a = 2
b = 3
0 1 2 3 4 5 6 7 8
25 4 12 37 57 48 86 33 92
A partir de a = 3, procurar
info[a] > pivo
a = 3
0 1 2 3 4 5 6 7 8
25 4 12 37 57 48 86 33 92
A partir de b = 2, procurar
info[b] < pivo
b = 2

Exemplo

0	1	2	3	4	5	6	7	8
25	4	12	37	57	48	86	33	92

a = 3
b = 2

- Trocamos a posição do pivô, com o elemento que está na posição *b*

0	1	2	3	4	5	6	7	8
12	4	25	37	57	48	86	33	92

Algoritmo QuickSort

```
Algoritmo: quickSort()
  n ← size(info)-1;
  quickSort(0, n);
```

```
Algoritmo: quickSort(int inicio, int fim)
se (inicio < fim)  então
   idxPivo ← particionar(inicio, fim);
   quickSort(inicio, idxPivo-1);
   quickSort(idxPivo+1, fim);
fim-se
```

Algoritmo QuickSort

Algoritmo: particionar(int inicio, int fim)

```
a ← inicio;
b ←  fim+1;
pivo ← info[inicio];

enquanto (verdadeiro) faça
   faça
      a ← a+1;
   enquanto (a ≤ fim e info[a] < pivo);

   faça
      b ← b-1;
   enquanto (b ≥ inicio e info[b]) > pivo);

   se (a ≥ b) então
     break;
   fim-se;

   trocar(a,b);
fim-enquanto;

trocar(b, inicio);
retornar b;
```

O método **trocar(int p1, int p2)** troca o elemento que está na posição **p1** pelo elemento que está na posição **p2**, e vice versa.

Esforço computacional

- No melhor caso:
  - O pivô representa um valor mediano que divide o conjunto de dados em dois sub-vetores de igual tamanho.
  - Neste caso, a ordem de complexidade é O(n log(n))

- No pior caso.
  - O pivô é o maior elemento ou o menor (não divide em dois sub-vetores de tamanho igual)
    - O algoritmo recai em Algoritmo de ordenação por Bolha

- Caso médio:
  - O algoritmo é O(n log(n))