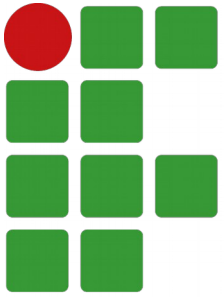
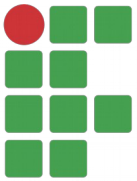


INSTITUTO FEDERAL
Ceará

QuickSort

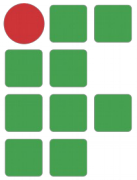
QuickSort



- Como o Mergesort, também é baseado no paradigma “dividir para conquistar”
- Entretanto, divisões das partições são dinâmicas
- Passos básicos:
  - Escolhe-se um pivô
  - O pivô é posicionado de forma que todos os elementos anteriores a ele sejam menores e todos os posteriores, maiores
  - Ordena recursivamente os subvetores à esquerda e à direita

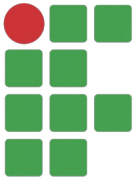
Elementos menores	**PIVÔ**	Elementos maiores

QuickSort



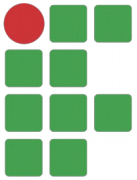
- Pivô encontra na posição correta
  - As chamadas recursivas desconsidera o pivô
- Exemplo
  - 25 57 48 37 12 92 86 33

QuickSort



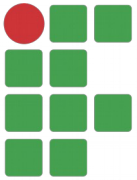
- 25 57 48 37 12 92 86 33
- Pivô = 25
- Após a primeira execução
  - (12) 25 (57 48 37 92 86 33)

QuickSort



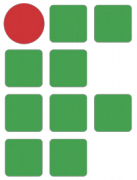
- 25 57 48 37 12 92 86 33
- Pivô = 25
- Após a primeira execução
  - (12) 25 (57 48 37 92 86 33)
  - (12) < 25 < (57 ... 33)

QuickSort



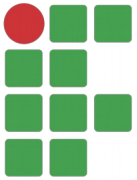
- 25 57 48 37 12 92 86 33
- Pivô = 25
- Após a primeira execução
  - (12) 25 (57 48 37 92 86 33)
  - (12) < 25 < (57 ... 33)
- Aplicamos o método recursivamente a cada um dos subvetores
  - (12) 25 (57 48 37 92 86 33)

QuickSort



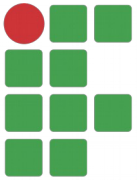
- (12) 25 (57 48 37 92 86 33)
- Ordenação do subvetor à esquerda de 25 (antigo pivô)
  - Pivô = 12

QuickSort



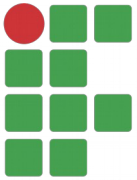
- 12 25 (57 48 37 92 86 33)
- Ordenação do subvetor à direita de 25 (antigo pivô)

QuickSort



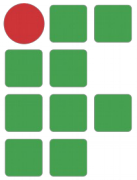
- 12 25 (57 48 37 92 86 33)
- Ordenação do subvetor à direita de 25 (antigo pivô)
  - 12 25 (48 37 33) 57 (92 86)

QuickSort



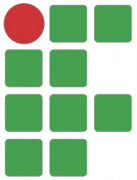
- 12 25 (48 37 33) 57 (92 86)
- Pivô = 48
- 12 25 (37 33) 48 57 (92 86)
- 12 25 (37 33) 48 57 (92 86) → pivô = 37
- 12 25 (33) 37 48 57 (92 86)
- 12 25 (33) 37 48 57 (92 86) → pivô = 37
- 12 25 33 37 48 57 (92 86)

QuickSort



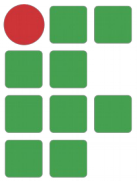
- 12 25 33 37 48 57 (92 86) → Pivô = 92
- 12 25 33 37 48 57 (86) 92
- 12 25 33 37 48 57 (86) 92
- 12 25 33 37 48 57 86 92

QuickSort



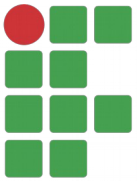
- Escolha do pivô
  - Elemento cuja posição será procurada
  - No exemplo: primeiro elemento do subvetor é o pivô
  - Outras abordagens:
    - Escolha aleatória
    - Meio
    - Fim
    - Mediana de n elementos

QuickSort

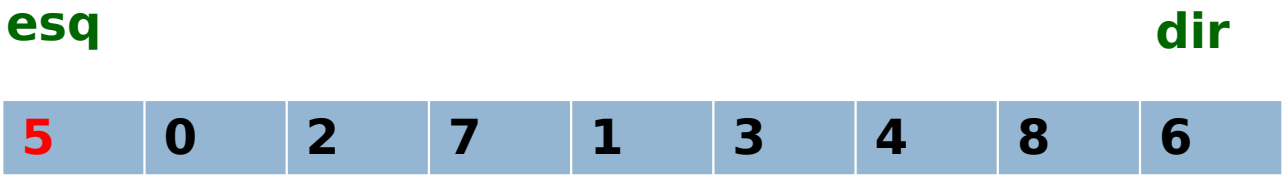


- Posicionamento do pivô
  - 2 ponteiros são definidos: 1 no início (esq), outro no fim (dir)
  - Enquanto esq <= dir
    - **Esq** será movido em direção a **dir** (e vice-versa)
      - Esq++ → enquanto pivo >= elemento em **esq**
      - Dir-- → enquanto pivo < elemento em **dir**

QuickSort

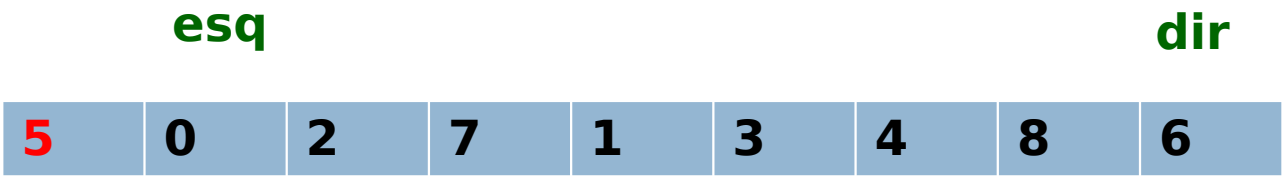


- Posicionamento do pivô
  - 2 ponteiros são definidos: 1 no início (esq), outro no fim (dir)
  - Enquanto esq <= dir
    - **Esq** será movido em direção a **dir** (e vice-versa)
      - Esq++ → enquanto pivo >= elemento em **esq**
      - Dir-- → enquanto pivo < elemento em **dir**

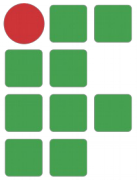


QuickSort

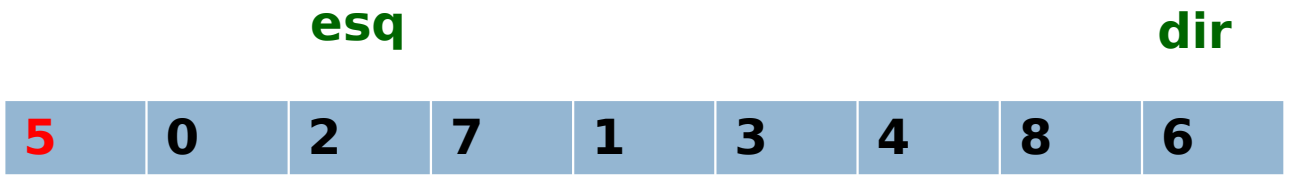
- Posicionamento do pivô
  - 2 ponteiros são definidos: 1 no início (esq), outro no fim (dir)
  - Enquanto esq <= dir
    - **Esq** será movido em direção a **dir** (e vice-versa)
      - Esq++ → enquanto pivo >= elemento em **esq**
      - Dir-- → enquanto pivo < elemento em **dir**



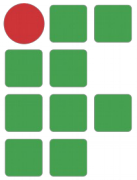
QuickSort



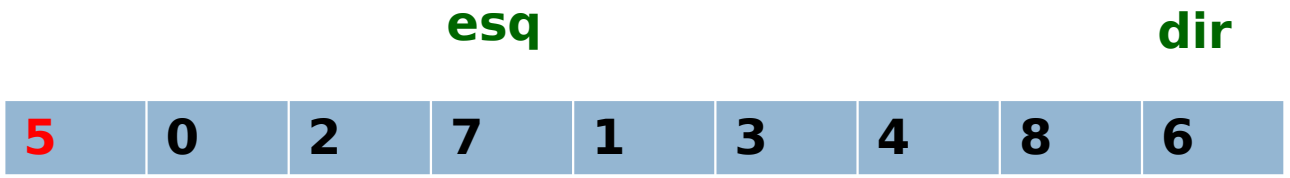
- Posicionamento do pivô
  - 2 ponteiros são definidos: 1 no início (esq), outro no fim (dir)
  - Enquanto esq <= dir
    - **Esq** será movido em direção a **dir** (e vice-versa)
      - Esq++ → enquanto pivo >= elemento em **esq**
      - Dir-- → enquanto pivo < elemento em **dir**



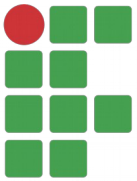
QuickSort



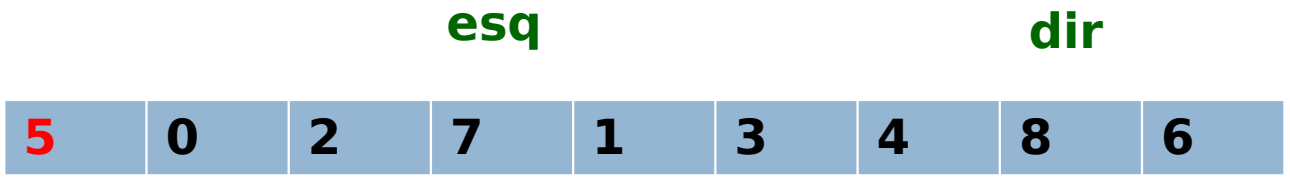
- Posicionamento do pivô
  - 2 ponteiros são definidos: 1 no início (esq), outro no fim (dir)
  - Enquanto esq <= dir
    - **Esq** será movido em direção a **dir** (e vice-versa)
      - Esq++ → enquanto pivo >= elemento em **esq**
      - Dir-- → enquanto pivo < elemento em **dir**



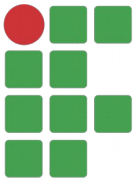
QuickSort



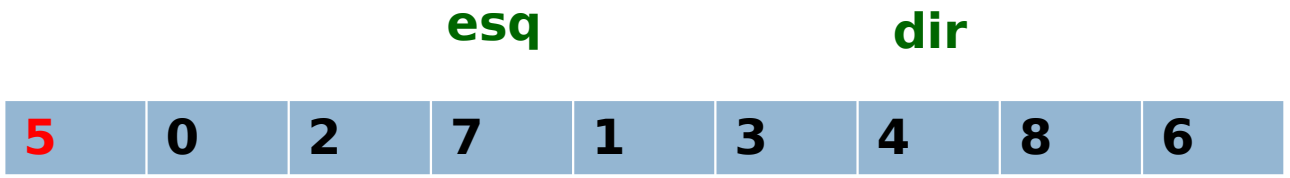
- Posicionamento do pivô
  - 2 ponteiros são definidos: 1 no início (esq), outro no fim (dir)
  - Enquanto esq <= dir
    - **Esq** será movido em direção a **dir** (e vice-versa)
      - Esq++ → enquanto pivo >= elemento em **esq**
      - Dir-- → enquanto pivo < elemento em **dir**



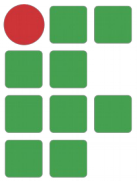
QuickSort



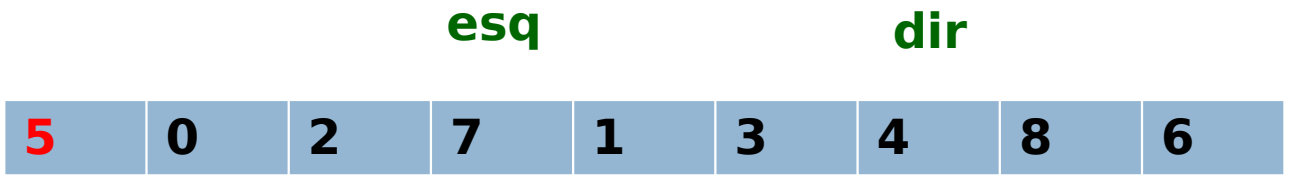
- Posicionamento do pivô
  - 2 ponteiros são definidos: 1 no início (esq), outro no fim (dir)
  - Enquanto esq <= dir
    - **Esq** será movido em direção a **dir** (e vice-versa)
      - Esq++ → enquanto pivo >= elemento em **esq**
      - Dir-- → enquanto pivo < elemento em **dir**



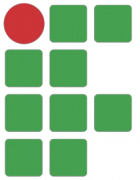
QuickSort



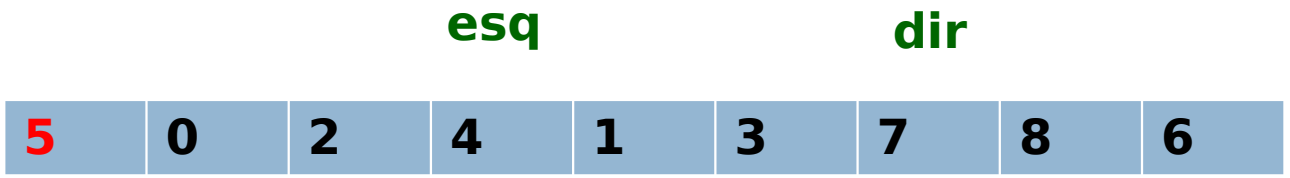
- Posicionamento do pivô
  - Se **esq** < **dir**
    - Troco os elementos



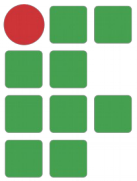
QuickSort



- Posicionamento do pivô
  - Se **esq** < **dir**
    - Troco os elementos



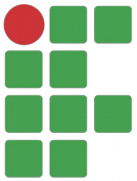
QuickSort



- Posicionamento do pivô
  - Continuo enquanto esq < dir
    - Os 2 ponteiros vão se cruzar

Esq → pivo >= v[esq]
Dir → pivo < v[dir]

QuickSort



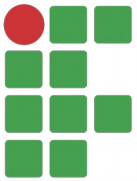
- Posicionamento do pivô
  - Continuo enquanto esq < dir
    - Os 2 ponteiros vão se cruzar

Esq → pivo >= v[esq]
Dir → pivo < v[dir]

esq **dir**

5	0	2	4	1	3	7	8	6

QuickSort



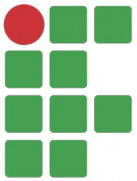
- Posicionamento do pivô
  - Continuo enquanto esq < dir
    - Os 2 ponteiros vão se cruzar

Esq → pivo >= v[esq]
Dir → pivo < v[dir]

esq
dir

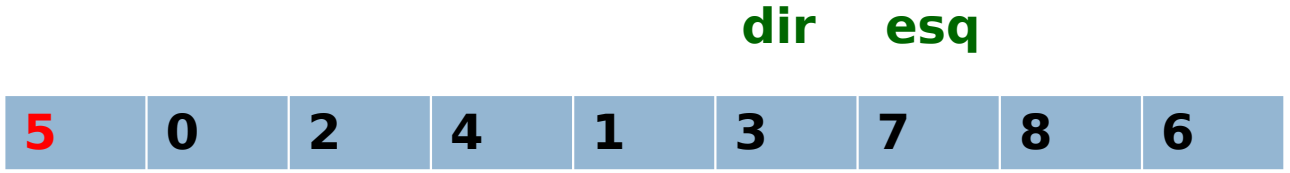
5	0	2	4	1	3	7	8	6

QuickSort

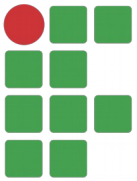


- Posicionamento do pivô
  - Continuo enquanto esq < dir
    - Os 2 ponteiros vão se cruzar

Esq → pivo >= v[esq]
Dir → pivo < v[dir]

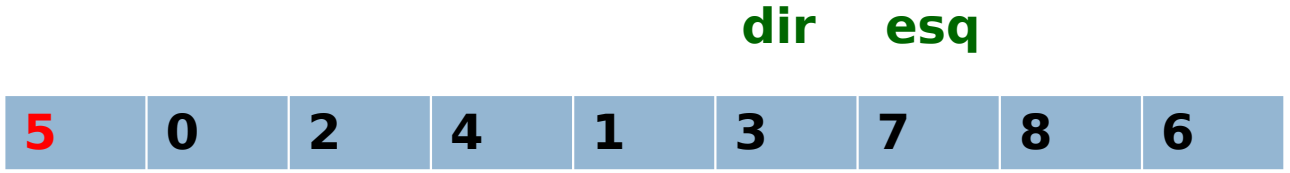


QuickSort

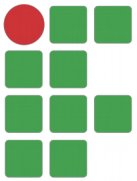


- Posicionamento do pivô
  - Elemento entre o pivô e dir
    - **Menores** que o pivô
  - Elementos entre o esq e o fim
    - **Maiores** que o pivô
  - Troco pivô e dir

Esq → pivo >= v[esq]
Dir → pivo < v[dir]

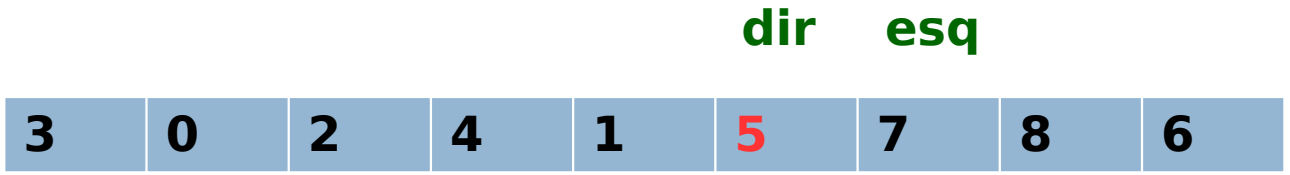


QuickSort

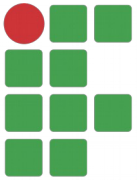


- Posicionamento do pivô
  - Elemento entre o pivô e dir
    - **Menores** que o pivô
  - Elementos entre o esq e o fim
    - **Maiores** que o pivô
  - Troco pivô e dir
  - Retorno dir → posição do pivô

Esq → pivo >= v[esq]
Dir → pivo < v[dir]

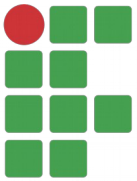


QuickSort



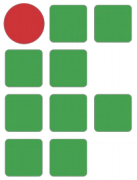
```
void quickSort(int *v, int inicio, int fim) {
    if (inicio < fim) {
        int pivo = particiona(v, inicio, fim);
        quickSort(v, inicio, pivo-1);
        quickSort(v, pivo+1, fim);
    }
}
```

QuickSort



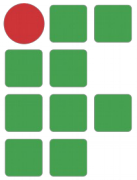
```
int particiona(int *v, int inicio, int fim) {
    int esq = inicio;
    int dir = fim;
    int pivo = v[inicio];

    while (esq < dir) {
        while (pivo >= v[esq]) {
            esq++;
        }
        while (pivo < v[dir]) {
            dir--;
        }
        if (esq < dir) {
            troca (&v[esq], &v[dir]);
        }
    }
    v[inicio] = v[dir]; //troco o lugar do elemento em dir e pivô
    v[dir] = pivo;
    return dir;
}
```

QuickSort 2



```c
/**
 * Algoritmo de partição proposta por Nico Lomuto
 * Descrito nos livros Programming Pearls (Bentley) e
 * Introduction to Algorithms (Cormen)
 * */
int particionaLomuto(int *v, int inicio, int fim) {
    int pivo = v[fim];
    int i=inicio;
    for (int j=inicio; j<fim-1; j++) {
        if (v[j] < pivo) {
            troca(&v[i], &v[j]);
            i++;
        }
    }
    troca(&v[i], &v[fim]);
    return i;
}
```

Exercício



- Dada a sequência de números: 3 4 9 2 5 1 8.
  - Ordene em ordem crescente utilizando os algoritmos QuickSort, apresentando a sequência dos números a cada passo (Teste de Mesa).