Quick Sort

Ciência da Computação
Laboratório de Ordenação e Pesquisa
Prof. M.Sc. Elias Gonçalves

História

- Proposto por Charles Antony Richard Hoare em 1960 e publicado em 1962.
- É o algoritmo de ordenação interna mais rápido que se conhece para uma ampla variedade de situações.
- É muito utilizado.

Ideia

Divisão e conquista

- Dividir o problema de ordenar um conjunto com n itens em dois problemas menores.
- Os problemas menores são ordenados independentemente.
- As partições são combinadas para produzir a solução final.

Algoritmo e código

Quicksort

➔A função quicksort recebe o vetor **v**, a posição inicial (**inicio**) e a posição final (**fim**). O caso base de parada da recursão é o início ser maior ou igual ao fim, por isso, se **inicio < fim** calcula o **pivo** chamando a função de particionamento e de forma recursiva aplica o quicksort na primeira e na ultima parte de **v**.

```
void quick_sort( int v[], int inicio, int fim ){
    if ( inicio < fim ){
        int pivo = particionar( v, inicio, fim );
        quick_sort( v, inicio, pivo-1 );
        quick_sort( v, pivo+1, fim );
    }
}
```

Ideia

Particionamento

- O vetor v é rearranjado por meio da escolha arbitrária de um pivo.
- O vetor v é particionado em dois:
  - Partição esquerda: chaves ≤ pivo;
  - Partição direita: chaves ≥ pivo.

Algoritmo

Particionamento

- O **pivo** pode ser escolhido arbitrariamente, porém, aqui optamos por escolher o último elemento do vetor **v**.

  pivo = v[fim];

- Crie duas variáveis, **i** e **j** para percorrer, respectivamente a parte inicial/esquerda de **v** e a parte final/direita de **v.** Inicialize ambas as variáveis com a posição inicial do vetor.

  i = j = inicio;

- Como o **pivo** é o último elemento de **v**, vamos percorrer **v** a partir do início/esquerda verificando se **v[j] <= pivo** e sempre incrementando o **j**. Para cada vez que essa condição for verdadeira, faz-se a troca do elemento em **v[j]** pelo elemento em **v[i]** e incrementa-se o **i**. Esse processo é repetido até o elemento na posição imediatamente antes do **pivo** (que está na ultima posição).

- Agora basta trocar o **v[fim]** (pivo) com o elemento na posição **v[i].** Assim, todos os elementos até **v[i]** são menores ou iguais ao **pivo** e todos os elementos de **v[i]** até **v[j]** são maiores ou iguais ao **pivo**.

Código

Particionamento

```
int particionar(int v[], int inicio, int fim){
    int pivo = v[fim];
    int i = inicio;
    int j;

    for(j=inicio; j<fim; j++){
        if(v[j] <= pivo){
            trocar( &v[j], &v[i] );
            i++;
        }
    }
    trocar( &v[i], &v[fim] );
    return i;
}
```

```
void trocar( int *x, int *y ){
    int aux;
    aux = *x;
    *x = *y;
    *y = aux;
}
```

Vantagens e desvantagens

- Vantagens:
  - Melhor opção para ordenar vetores grandes;
  - Muito rápido por que o laço interno é simples;
  - Memória auxiliar para a pilha de recursão é pequena;
  - Complexidade no caso médio é de ordem logarítmica.
- Desvantagens:
  - Não é estável (não se conhece forma eficiente para tornar o quicksort estável);
  - Pior caso é de ordem quadrática.

Biblioteca Virtual

CELES, Waldemar; CERQUEIRA, Renato; RANGEL, José Lucas. **Introdução a Estruturas de Dados com Técnicas de Programação em C** (Capítulo 11);

DROZDEK, Adam. **Estrutura de dados e algoritmos em c++** (Capítulo 9);

MARKENZON, Lilian; SZWARCFITER, Jorge Luiz. **Estruturas de Dados e seus Algoritmos** (Capítulos: 7, 11 e 12);

PINTO, Rafael Albuquerque. **Estrutura de Dados** (Páginas 155 a 177).