Estrutura de Dados

Métodos de Ordenação
Quicksort

Prof. Nilton Luiz Queiroz Jr.

Métodos de ordenação

- Dentro da categoria de métodos de ordenação em memória primária existem alguns métodos de ordenação que são mais eficientes que os métodos simples:
  - Mergesort;
  - **Quicksort**;
  - Heapsort;
  - Shellsort;

Quicksort

- O quicksort é um método de ordenação rápido e é considerado um dos método de ordenação mais eficientes;
- Método proposto por C. A. Hoare, em 1962, na Universidade de Moscou;
- Utiliza a estratégia de dividir para conquistar;
    - Dividir o problema em subproblemas menores;
    - Combinar as soluções para obtenção da solução final;

Quicksort

- O quicksort divide o vetor em duas partes quando possível;
- Ordena cada uma das partes usando o quicksort;
- Possui tempo computacional bem menor que os métodos simples;
- Recomendado para conjuntos grandes;

Quicksort

- O método consiste em:
  - Escolher um pivo x;
  - Colocar todos itens com chave menor que x à esquerda de x, formando uma sequência SE;
  - Colocar todos itens com chave maior que x à direita de x, formando uma sequência SD;
  - Após isso, aplica-se o quicksort em cada uma das sequências SE e SD;
  - Esse processo deve ser aplicado sucessivamente até que as sequências tenham apenas um elemento;
- O processo de dividir o vetor em elementos maiores que o pivo e elementos menores que o pivo deve ser feito em tempo linear;
  - Em geral o procedimento é chamado de partition;

Quicksort

- O algoritmo segue os seguintes passos:
  - Selecione o pivo;
  - Percorra o vetor, colocando todos os valores menores que o pivô da posição inicial do vetor mais 1 até a posição j do vetor;
  - Quando encontrar um valor menor que o pivo coloque-o na posição j;
  - Ao terminar de percorrer o vetor troque o pivo com o elemento da posição j-1;
  - Repita o quicksort da para os vetores da posição inicial do vetor até a posição antecessora à que o pivo doi colocado, e para a posição sucessora à que o pivo foi colocado até a posição final do vetor;
- Em geral tanto a posição inicial quanto a posição final são parâmetros do algoritmo;

Obs: A posição inicial e a posição final do vetor mudam conforme a chamada recursiva

Exemplo

- Em geral o pivô é o primeiro elemento;
  - Uma alternativa é selecionar o pivo aleatoriamente e o colocar na primeira posição;

3	5	2	6	1	7	4
pivo	i j					

Exemplo

3	5	2	6	1	7	4
pivo	j	i				

- pivo > v[i]
  - Trocar v[i] com v[j] e incrementar j

3	2	5	6	1	7	4
pivo		i j				

Exemplo

3	2	5	6	1	7	4
pivo		j	i			

Exemplo

3	2	5	6	1	7	4
pivo		j		i		

- Pivo maior que v[i]
  - Trocar v[i] com v[j] e incrementar j

3	2	1	6	5	7	4
pivo			j	i		

Exemplo

3	2	1	6	5	7	4
pivo			j		i	

Exemplo

3	2	1	6	5	7	4
pivo			j			i

Exemplo

- Percorreu o vetor a primeira vez
  - Troca pivo com elemento da posição j-1;

3	2	1	6	5	7	4
pivo			j			i

1	2	3	6	5	7	4
		pivo	j			i

Exemplo

- Aplicar o quicksort da posição inicial do vetor até a nova posição do pivo-1;
- Aplicar o quicksort da nova posição do pivo+1 até a posição final do vetor;

1	2	3	6	5	7	4
		pivo	j			i

Exemplo

- Trocar pivo com elemento da posição j-1;

1	2	3	6	5	7	4
pivo	i j					

Exemplo

- Somente um elemento;
  - Ordenado

1	2	3	6	5	7	4
	pivo					

Exemplo

- Aplicando quicksort para o outro lado
  - Note voltamos ao ponto da recursão em que o pivo era 3, e então fizemos a chamada do quicksort para o lado esquerdo do pivo (nova posição do pivo + 1 até posição final);

1	2	3	6	5	7	4
			pivo	i j		

- Pivo > v[i]
  - Trocar v[i] com v[j] e incrementar j;

1	2	3	6	5	7	4
			pivo		i j	

Exemplo

1	2	3	6	5	7	4
			pivo		j	i

- Pivo > v[i]
  - trocar v[i] com v[j] e incrementar j

1	2	3	6	5	4	7
			pivo			i j

Exemplo

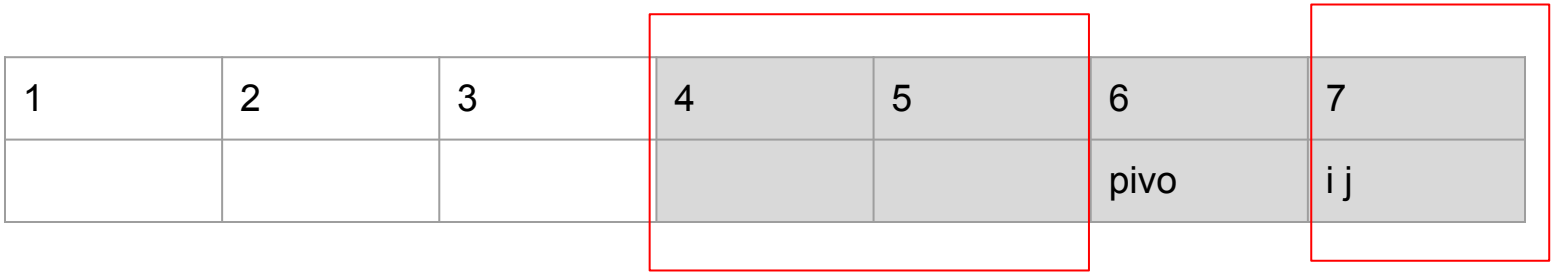
1	2	3	6	5	4	7
			pivo			i j

- Fim do vetor
  - Trocar pivo com v[j-1]

1	2	3	4	5	6	7
					pivo	i j

Exemplo

- Aplicar quicksort nos dois subconjuntos



Exemplo

- Terminou o vetor
  - Trocar pivo com v[j-1]

1	2	3	4	5	6	7
			pivo	i j		

Exemplo

- Somente um elemento

1	2	3	4	5	6	7
				pivo		

Exemplo

- Aplicando quicksort do outro lado do pivo;
- Somente um elemento;

1	2	3	4	5	6	7
						pivo

Exemplo

1	2	3	4	5	6	7

Implementação

- Pode-se utilizar um procedimento auxiliar para trocar dois valores na implementação do quicksort;

```
void trocar(int *a,int *b){
    int aux;
    aux=*a;
    *a=*b;
    *b=aux;
}
```

Implementação

- Para implementar o quicksort, podemos implementar um subprograma para particionar um sub-vetor;
  - Delimitando o **índice** de posição inicial por um parâmetro ini e o **índice** de sua posição final por um parâmetro fim;
  - É interessante retornar a posição em que o ficou pivo, pois o algoritmo quicksort irá utiliza-la como base;

Implementação

```
int partition(int v[],int ini,int fim){
    /*algoritmo*/
}
```

Implementação

```
int partition(int v[],int ini,int fim){
    int i,j,pivo;
    pivo=v[ini];
    j=ini+1;
    for(i=ini+1;i<=fim;i++){
        if(v[i]<pivo){
            trocar(&v[i],&v[j]);
            j++;
        }
    }
    trocar(&v[j-1],&v[ini]);
    return j-1;
}
```

Implementação

- Considerando a função partition, a implementação do quicksort pode ser feita de maneira recursiva;
  - Vale lembrar que após a execução da função partition, o pivo está em seu lugar, e o valor retornado é a posição em que o pivo está;

```
void quicksort(int v[], int ini, int fim){
    int pos_pivo;
    if(ini < fim){
        pos_pivo=partition(v,ini,fim);
        quicksort(v,ini,pos_pivo-1);
        quicksort(v,pos_pivo+1,fim);
    }
}
```

Quicksort

- Vantagem:
  - O melhor caso e o caso médio do quicksort tem tempo de execução na ordem de $O(n \log_2 n)$;
  - O algoritmo ordena os elementos *in situ*;
    - Não usa estruturas auxiliares;
- Desvantagem:
  - Seu pior caso tem tempo de execução $O(n^2)$;
    - É possível utilizar algumas estratégias para evitar o pior caso:
      - Mediana de 3;
      - Trocar o algoritmo de ordenação quando o pior caso for detectado;

Exercícios

1. Mostre o passo a passo da ordenação dos seguintes valores com o algoritmo quicksort:
   a. 1, 6, 5, 25, 15, 3, 9, 7
   b. 6, 3, 2, 4, 1, 9, 0