Programação 2

Algoritmos de Busca

Instituto Federal do Espírito Santo - Ifes
Campus Colatina
Professor: Giovany Frossard Teixeira

Definição

- São algoritmos que tem como objetivo encontrar um elemento em uma estrutura de dados.

Busca Sequencial (ou Linear)

- Consiste em buscar sequencialmente elemento por elemento até encontrar, ou não, o elemento desejado.
- Sua complexidade é, no melhor caso, O(1) esse caso ocorre quando encontramos o elemento desejado já na comparação com o primeiro elemento.
- O pior caso é aquele que varremos o vetor todo e não encontramos o elemento, sendo portanto, O(n).
- Podemos considerar o caso médio como encontrando o elemento mais ou menos no meio do vetor, ou seja, no elemento n/2. Nesse caso, portanto, o algoritmo também é O(n) pela regra da constante.

Busca Sequencial (ou Linear)

```
/**
  * Retorna -1 caso não encontre ou a posição, caso encontre.
  */
int buscaLinear(int *vetor, int tamanho, char elementoProcurado) {
     int i;
     for (i = 0; i < tamanho; i++) {
         if (vetor[i] == elementoProcurado) {
             return i;
         }
     }
     return -1;
}


int _tmain(int argc, _TCHAR* argv[])
{
    int vet[10] = {23, 17, 8, 15, 9, 12, 19, 7, 14, 10};
    printf("A posicao do elemento eh: %d", buscaLinear(vet, 10, 12));
    return 0;
}
```

Busca Binária

- Parte do **pressuposto** de que o **vetor está ordenado** e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Busca Binária

```c
#include "stdafx.h"

int buscaBinaria(int *vetor, int inicio, int fim, int elementoProcurado) {
    int meio = (inicio + fim)/2;


    if (vetor[meio] == elementoProcurado)
        return meio;
    else if (inicio >= fim)
        return -1;
    else if (vetor[meio] < elementoProcurado)
        return buscaBinaria(vetor, meio+1, fim, elementoProcurado);
    else
        return buscaBinaria(vetor, inicio, meio-1, elementoProcurado);
}


int _tmain(int argc, _TCHAR* argv[])
{
    int i;
    int vet[10] = {1, 6, 9, 15, 55, 87, 99, 150, 800, 1000};
    printf("Imprimindo as posicoes de todos os elementos\n");
    for (i= 0; i<10; i++)
        printf("A posicao do elemento eh: %d \n", buscaBinaria(vet, 0, 9, vet[i]));
    printf("\n\nVerificando que nao sao encontrados antecessores dos elementos\n");
    for (i= 0; i<10; i++)
        printf("A posicao do elemento eh: %d \n", buscaBinaria(vet, 0, 9, vet[i] - 1));
    printf("\n\nVerificando que nao sao encontrados sucessores dos elementos\n");
    for (i= 0; i<10; i++)
        printf("A posicao do elemento eh: %d \n", buscaBinaria(vet, 0, 9, vet[i] + 1));
    return 0;
}
```


Instituto Federal do Espírito Santo - Ifes
Campus Colatina
Professor: Giovany Frossard Teixeira

Saída do Algoritmo – para o exemplo

```
Imprimindo as posicoes de todos os elementos
A posicao do elemento eh: 0
A posicao do elemento eh: 1
A posicao do elemento eh: 2
A posicao do elemento eh: 3
A posicao do elemento eh: 4
A posicao do elemento eh: 5
A posicao do elemento eh: 6
A posicao do elemento eh: 7
A posicao do elemento eh: 8
A posicao do elemento eh: 9

Verificando que nao sao encontrados antecessores dos elementos
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1

Verificando que nao sao encontrados sucessores dos elementos
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
A posicao do elemento eh: -1
```

Análise de Desempenho

- No melhor caso, o elemento buscado é encontrado na primeira chamada da função (sem qualquer chamada recursiva ainda). Nesse contexto o algoritmo é O(1).
- No pior caso, o elemento não é encontrado. Como a altura de uma árvore é dada por h = lg2 n e o algoritmo vai ser executado h vezes, podemos dizer que o algoritmo é O(log n).
- No caso médio, o algoritmo também é O(log n) pela regra da constante.

Bibliografia

- AZEREDO, Paulo A. **Métodos de Classificação de Dados**. Rio de Janeiro: Campus,1996.
- SANTOS, Henrique José. **Curso de Linguagem C da UFMG**, apostila.
- FORBELLONE, André Luiz. **Lógica de Programação – A Construção de Algoritmos e Estruturas de Dados.** São Paulo: MAKRON, 1993.