Tipos de Busca

Busca Sequencial

Busca Binária

Arvore de Busca Binária

Hash

Busca Sequencial

Compara a chave com cada item na array ou lista, até encontrar um item de dado cujo valor é igual o valor da chave.

Busca Sequencial - Code

```
for (i=0; i<n; i++)
      if (A[i]==x)
          return(i); /chave encontrada/
return(-1); /chave não encontrada/
```

Algoritmo de busca seqüencial em um vetor A, com N posições (0 até N-1), sendo x a chave procurada

Busca Sequencial - Code c/ Sentinela

```
A[N]=x;
for(i=0; x!=A[i]; i++);
if (i<n) return(i); /chave encontrada/
else return(-1); /sentinela encontrado/
```

Busca Binária - Code

```
Bin-Search(collection c, low, high, k)
int mid;
if low > high
     then return NIL;
mid = (high+low)/2;
if k = key[mid]
     then return key[mid];
     else if k < key[mid]
        then return Bin_search(c, low, mid-1, k);
        else return Bin_search(c, mid+1, high, k);
```

Busca Binária - Complexidade

O(log(n)), pois cada comparação reduz o número de possíveis candidatos por um fator de 2.

Árvore Binária de Busca - Busca Geral Recursivo

```
Tree-Search(x, k)
if x = NIL or k = key[x]
    then return x
if k < key[x]
    then return Tree-Search(left[x], k)
    else return Tree-Search(right[x], k)
```

Árvore Binária de Busca - Busca Geral Iterativo

```
Iterative-Tree-Search(x, k)
while x ≠ NIL and k ≠ key[x]
    do if k < key[x]
        then x ← left[x]
        else x ← right[x]
return x
```

Árvore Binária de Busca - Busca do Valor Minimo

```
Tree-Minimum(x)
while left[x] ≠ NIL
    do x ← left[x]
return x
```

Árvore Binária de Busca - Busca do Valor Maximo

```
Tree-Maximum(x)
while right[x] ≠ NIL
    do x ← right[x]
return x
```

Algoritmo de Busca do Valor Sucessor

O sucessor do nó x é o nó com o menor chave maior que key[x].
Case 1: Se a subarvore direita do nó x não for vazio, então, o sucessor do x é o nó mais esquerdo na subarvore direita;
Case 2: Se a subarvore direita do nó x for vazio, o sucessor do x (se x é um filho esquerdo) é o antecessor de nível mais baixa ou é o antecessor de nível mais baixa , cujo filho esquerdo também é antecessor do x (se x é um filho
direito) .

Algoritmo de Busca do Valor Sucessor

```
Tree-Successor(x)
if right[x] ≠ NIL
    then return Tree-Minimum(right[x])
y ← p[x]
while y ≠ NIL and x = right[p[x]]
    do x ← y
        y ← p[y]
return y
```

Algoritmo de Inserção

```
Tree-Insert(T, z)
y ← NIL
x ← root[T]
while x ≠ NIL
    do y ← x
        if key[z] < key[x]
            then x ← left[x]
            else x ← right[x]
p[z] ← y
if y = NIL
    then root[T] ← z
    else if key[z] < key[x]
        then left[y] ← z
        else right[y] ← z
```

Algoritmo de Remoção

```
Tree-Delete(T, z)
if left[z] = NIL ou right[z] = NIL
    then y ← z
    else y ← Tree-Successor(z)
if left[y] ≠ NIL
    then x ← left[y]
    else x ← right[y]
if x ≠ NIL
    then p[y] ← p[x]
if p[y] = NIL
    then root[T] ← x
```

Algoritmo de Remoção (cont)

```
    else if y =left[p[y]]
        then left[p[y]] ← x
        else right[p[y]] ← x
if y ≠ z
    then key[z] ← key[y]
return y
```

Ver Note

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int
conj1,conj2,conj3,n1,n2,n3,i,k1,k2,k3,soma1,soma2,som
a3,mediageral;
    float media1,media2,media3;
    printf("Insira o numero de valores para serem
somados no conjunto 1: \n");
    scanf("%d", &n1);
    soma1=0;
    for(i=0;i<n1;i++){
       printf("insira os valores: \n");
       scanf("%d",&k1);
       soma1=k1+soma1;
    }
   media1= (float) soma1/n1;
        printf("A média do conjunto 1 é: %f\n",
media1);
   printf("Insira o numero de valores para serem
somados no conjunto 2: \n");
    scanf("%d", &n2);
    soma2=0;
    for(i=0;i<n2;i++){
       printf("insira os valores: \n");
       scanf("%d",&k2);
       soma2=k2+soma2;
    }
        media2= (float)soma2/n2;
        printf("A média do conjunto 1 é: %f\n",
media2);
  printf("Insira o numero de valores para serem
somados no conjunto 3: \n");
    scanf("%d", &n3);
    soma3=0;
    for(i=0;i<n3;i++){
       printf("insira os valores: \n");
       scanf("%d",&k3);
       soma3=k3+soma3;
    }
        media3= (float)soma3/n3;
        printf("A média do conjunto 3 é: %f\n",
media3);
```

Ver Note (cont)

```
   mediageral= (float)(media1+media2+media3)/3;
    printf ("A média final gerada dos três conjuntos
é: %f\n",mediageral);
```

Exercício 2.7.1. Escrever um programa C, sem utilizar funções, que
a) Leia três conjuntos de n números reais digitados pelo usuário (n pode ser diferente para cada conjunto)

Ver Note 2

```
#include <stdio.h>
#include <stdlib.h>
float conjunto(void){
    int i,n,k;
    float soma,media;
    printf("Insira o numero de valores para serem
somados no conjunto: \n");
    scanf("%d", &n);
    soma=0;
    for(i=0;i<n;i++){
       printf("insira os valores: \n");
       scanf("%d",&k);
       soma=k+soma;
    }
        media= (float) soma/n;
        return (media)
}
int main(){
    float media1,media2,media3, mediageral;
    media1=conjunto ();
     media2=conjunto ();
      media3=conjunto ();
    printf("A média do conjunto 1 é: %f\n", media1);
    printf("A média do conjunto 2 é: %f\n", media2);
    printf("A média do conjunto 3 é: %f\n", media3);
     mediageral= (media1+media2+media3)/3.0;
    printf ("A média final gerada dos três conjuntos
é: %f\n",mediageral);
```

b) Imprima a média e o desvio padrão de cada um dos três conjuntos;

Questao 4

```
int soma (int valor){
    int aux;
    if (valor == -1 || valor == 1){
        valor = -1;
    }
    else{
        valor = ((-2*valor) +1) + soma(valor-1);
     }
     return valor;
}
```

Escreve uma função recursiva para calcular a seuinte soma: -1-3-5-7-...-(2N-1)

By **malandro123**
cheatography.com/malandro123/

Published 27th October, 2015.
Last updated 27th October, 2015.