INSTITUTO FEDERAL
RIO GRANDE DO SUL

Estruturas de Dados II

Busca Sequencial (Linear) x Busca Binária

Dados Não Ordenados X Dados Ordenados

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Agosto/2023

Busca Não Ordenado X Ordenado

O problema da busca (ou pesquisa)

"Dado um conjunto de elementos, onde cada um possui um conjunto de propriedades, identificado por uma chave, o objetivo da busca é localizar, nesse conjunto, o elemento que corresponde a uma chave específica"

O objetivo é encontrar "o elemento" (registro, nó, dado, etc) com o menor custo.

Busca Não Ordenado X Ordenado

Algumas técnicas usadas para busca de dados são:

- **Busca Seqüencial**
  - É a forma mais simples de busca, para dados ordenados ou não, onde cada elemento é comparado. Complexidade é O(n).

- **Busca Binária**
  - Divide o conjunto em duas partes e compara o elemento buscado ao meio do arranjo. Se igual, busca bem-sucedida.Caso o elemento esteja no lado inferior, busca-se na metade inferior ou caso esteja no lado superior,busca-se na metade superior. (log(n))

INSTITUTO FEDERAL
RIO GRANDE DO SUL

Busca Não Ordenado X Ordenado

- **Busca por Interpolação (Chaves Distribuídas)**
  - Semelhante a Busca Binária, esse método divide em subconjuntos de dados para aumentar e eficiência da busca binária. Complexidade (log(log(n)))

- **Busca em Árvores**
  - Arvores são estruturas hierárquicas que podem ser utilizadas para a localização de chaves em arvores estruturadas: a chave para cada nó deve ser maior do que quaisquer chaves presentes em subárvores da esquerda e menor a quaisquer chaves em subárvores à direita (o vice versa – depende da implementação). Complexidade (log(n))

- **Hashing**
  - Método de pesquisa direto ao elemento baseado em um calculo (hash) para o índice do elemento. Complexidade O(1) (melhor caso)

Busca Sequencial: Não Ordenado

Busca Sequencial – Dados não Ordenados

Dada uma coleção de elementos não ordenados, pretende-se saber se um determinado elemento valor está presente nessa coleção.

Imagine uma conjunto de números implementada como sendo um vetor de ***n*** elementos inteiros:

vetor[0]..vetor[n-1].

Uma solução possível é percorrer todo o vetor desde a primeira posição até a ultima e para cada posição do vetor, comparamos o valor do vetor com procurado:

- Se forem iguais dizemos que valor existe.
- Se chegarmos ao fim do vetor sem sucesso dizemos que valor não existe.

Busca Sequencial: Não Ordenado

```c
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO 20
main()
{
 int x, i=0, vetor[20] = {21,12,3,14,5,20,50,10,35,9,19,33,44,16,37,8,39,22,6,7};
printf("Qual Número deseja localizar?:");
scanf("%d",&x);
    while (i < TAMANHO && vetor[i]!=x)
    {
    i++;
    }

    if (vetor[i]==x)
        printf("Valor Localizado");
    else
        printf("Valor NAO Localizado");
}
```

Busca Sequencial: Dados Ordenados

Busca Sequencial (Linear) – Dados Ordenados

Considerando um conjunto de elementos ordenados, pretende-se saber se um determinado elemento valor está presente nessa coleção.

Suponha que esse comjunto é implementada como sendo um vetor de ***n*** elementos inteiros ordenados, onde :

```
vetor[0]..vetor[n-1]  e
vetor[0] < vetor[1].. vetor[i] < vetor[n-1]
```

Uma solução possível é percorrer o vetor desde a primeira posição até a ultima ou até o elemento maior ao valor procurado. Se o elemento procurado for menor do que o valor em uma determinada posição do array, temos a certeza de que ele não estará no restante do array.

Para cada posição i, comparamos vetor[i] com valor e comparamos se é menor:

- **Se forem iguais dizemos que valor existe.**
- Se chegarmos ao **fim do vetor ou um valor maior**, dizemos que valor não existe. Isso evita a necessidade de percorrer o array do seu início até o seu fim

Busca Sequencial: Dados Ordenados

```c
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO 20
main()
{
 int x, i=0, vetor[20] = {3,5,6,7,8,9,10,12,14,16,19,20,21,22,33,35,37,39,44,50};

printf("Qual Numero deseja localizar?:");
scanf("%d",&x);
    while (i < TAMANHO && vetor[i]!=x && vetor[i]<= x )
    {
        i++;
    }
    if (vetor[i]==x)
        printf("Valor Localizado");
    else
        printf("Valor NAO Localizado");
}
```

Busca Binária: Dados Ordenados

Busca Binária – Somente dados Ordenados.

Considere um conjunto de dados ordenados , onde ***i e j***, sendo ***i < j***, se, e somente se***, A[i] ≤ A[j].***

Neste método, comparando um determinado elemento com o elemento procurado,
- é possível identificar se o elemento procurado é o elemento comparado,
- se o elemento ele está antes do elemento comparado ou
- se está depois do elemento comparado.

Se fizermos isso sempre com o elemento do meio da lista, a cada comparação dividiremos o conjunto em duas partes, **reduzindo nosso tempo de pesquisa**.

Se em um determinado momento o vetor, após sucessivas divisões, tiver tamanho zero, então o elemento não está no vetor.

Busca Binária

```c
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO 20
main()
{
 int x, i=0, vetor[20] = {3,5,6,7,8,9,10,12,14,16,19,20,21,22,33,35,37,39,44,50};

    printf("Qual Numero deseja localizar?:");
    scanf("%d",&x);
   int meio=0, esquerda = -1, direita = TAMANHO;
   while (esquerda < direita-1) {
      int meio = (esquerda + direita)/2;
      if (vetor[meio] < x)
        esquerda = meio;
      else
        direita = meio;
   }
    if (vetor[direita]==x)
        printf("Valor Localizado");
    else
        printf("Valor NAO Localizado");
}
```