ALGORITMOS E LÓGICA DE PROGRAMAÇÃO II

Prof. Wilson Lourenço

PESQUISA SEQUENCIAL

Pesquisa binária e sequencial

Busca

Busca é um método bastante utilizado em programação, pois frequentemente necessitamos de informações que podem ser adquiridas através dela.

Pode-se realizar busca em diversos lugares como: vetores (verificação da existência de um valor dentro dele), intervalos numéricos (qual o melhor ponto dentro de um intervalo que satisfaz o problema), strings (se um letra está contida em uma string) e etc.

Pesquisa binária e sequencial

Ordenação

A ordenação consiste em dispor elementos em uma certa sequência, seguindo algum critério. Por exemplo, a ordenação lexicográfica (alfabética) para dados literais, ou crescente e decrescente para dados numéricos.

Existem muitos algoritmos de ordenação conhecidos: InsertionSort, ShellSort, BubbleSort, HeapSort, MergeSort , QuickSort, dentro Outros.

Pesquisa binária e sequencial

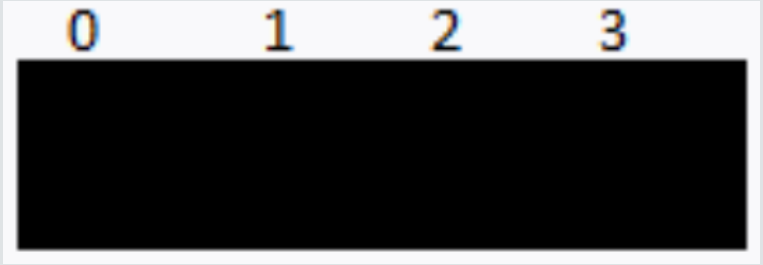
Busca Linear

Vamos supor que temos 4 portas fechadas, e atrás de cada porta contém 1 número, qual estratégia poderíamos tomar para encontrar o número 3?

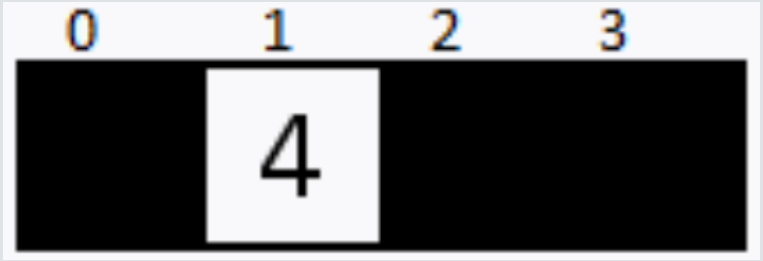
O método trivial seria abrir qualquer porta aleatoriamente enquanto não encontrar o número 3.

Pesquisa binária e sequencial

1º Passo:

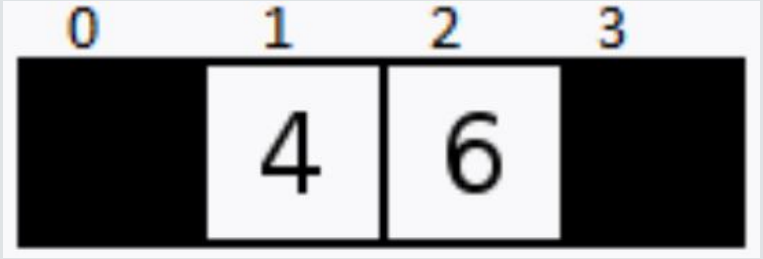


2º Passo:

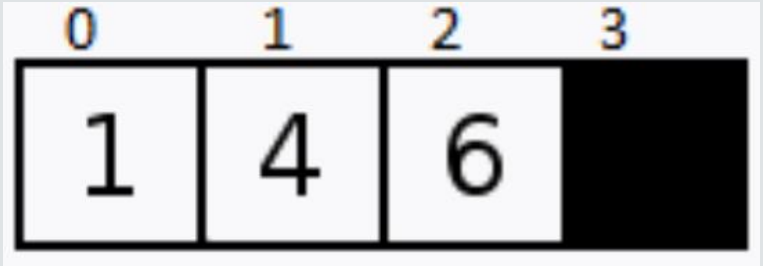


Pesquisa binária e sequencial

3º Passo:

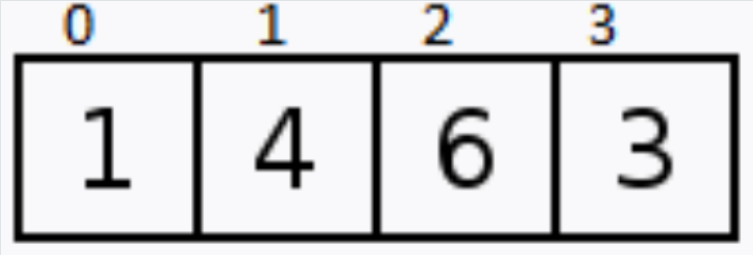


4º Passo:



Pesquisa binária e sequencial

5º Passo:



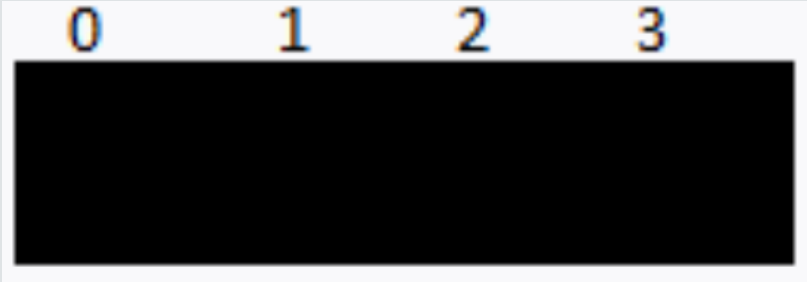
Pesquisa binária e sequencial

- Obviamente este método não é nem de perto o mais eficiente.

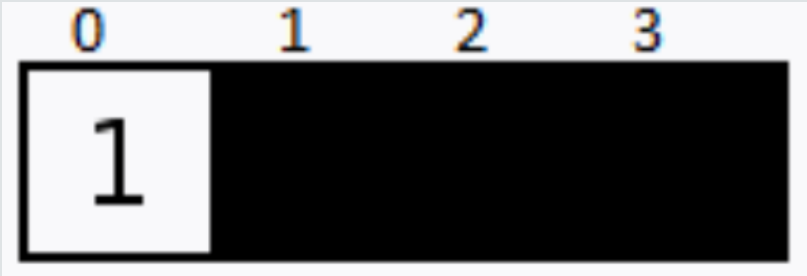
- Outra estratégia seria começando na primeira porta e ir abrindo as próximas na sequência. Este método se chama Busca Linear.

Pesquisa binária e sequencial

1º Passo:

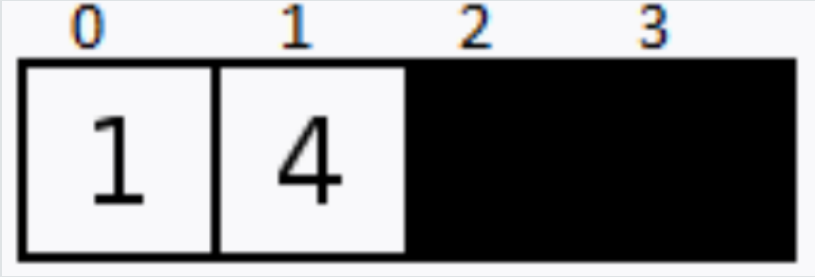


2º Passo:

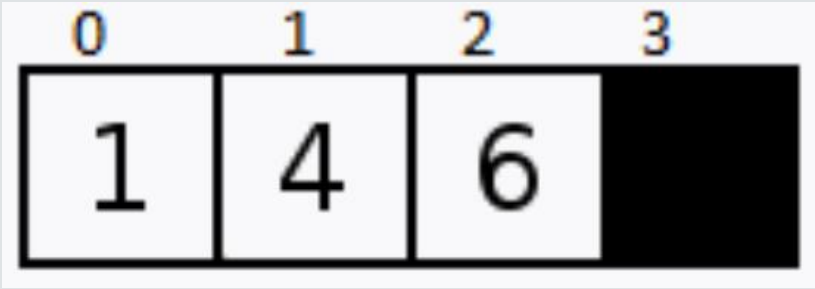


Pesquisa binária e sequencial

3º Passo:

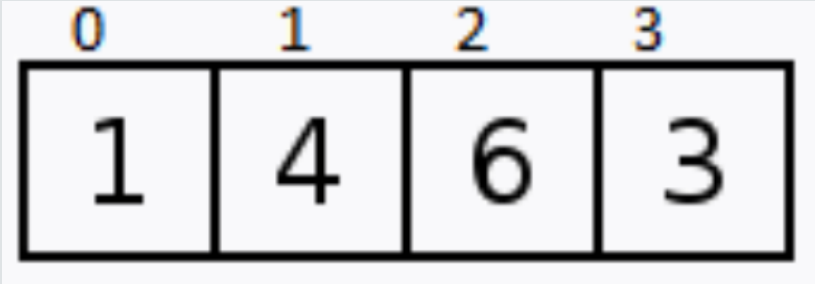


4º Passo:



Pesquisa binária e sequencial

5º Passo:



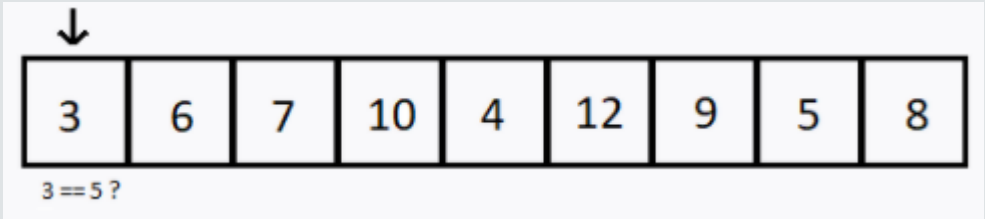
Pesquisa binária e sequencial

Verificamos sequencialmente (ou seja, um após o outro) cada elemento. Se encontramos o valor desejado, então a pesquisa foi bem sucedida.

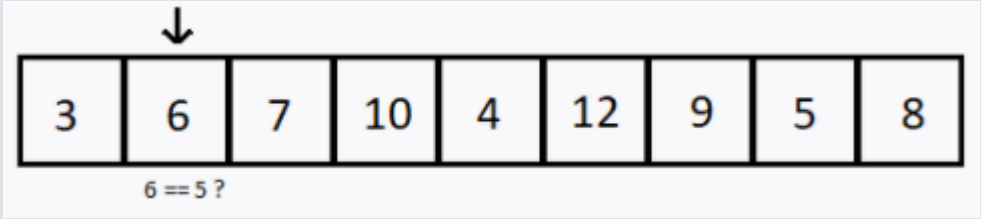
Caso todos os elementos do conjunto sejam verificados e o elemento desejado não esteja entre eles, dizemos que a pesquisa foi mal sucedida.

Pesquisa binária e sequencial

1º Passo:

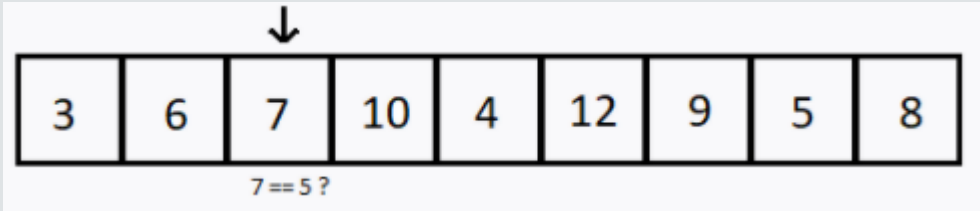


2º Passo:

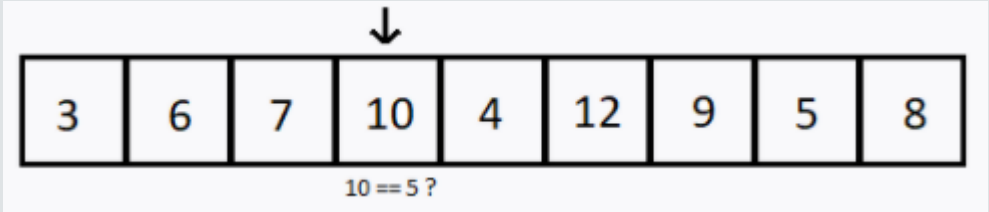


Pesquisa binária e sequencial

3º Passo:

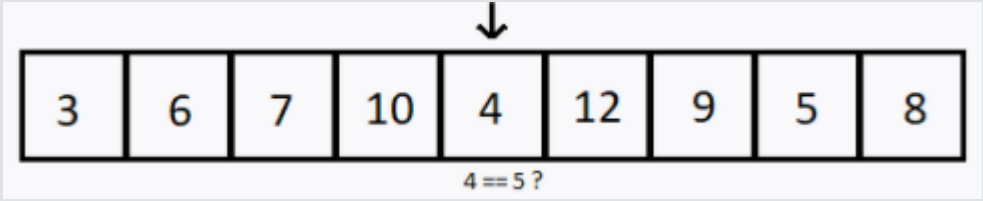


4º Passo:

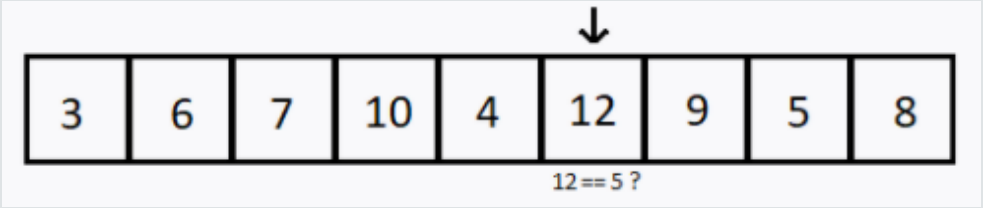


Pesquisa binária e sequencial

5º Passo:

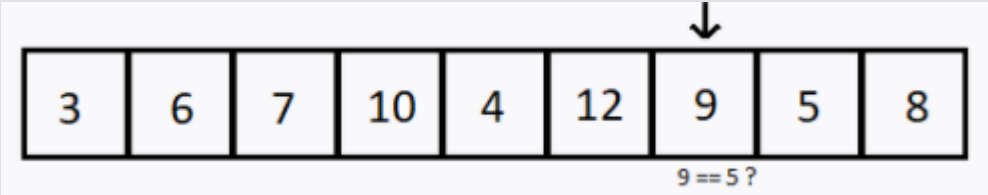


6º Passo:

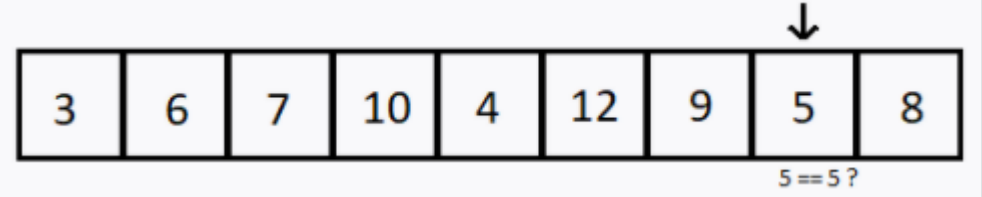


Pesquisa binária e sequencial

7º Passo:

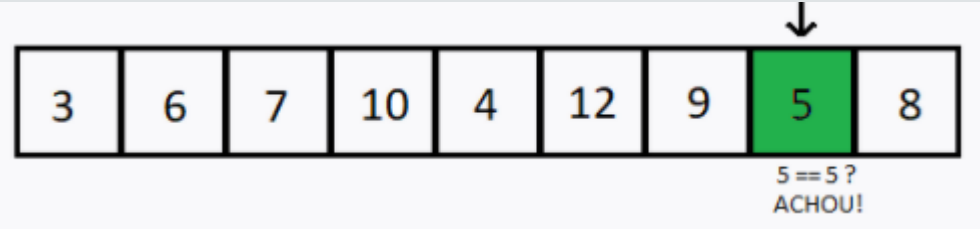


8º Passo:



Pesquisa binária e sequencial

8º Passo:



Pesquisa binária e sequencial

Busca binária

A pesquisa binária utiliza a técnica de “dividir e conquistar”.

Primeiro, testamos se o elemento procurado é menor que o elemento do meio do vetor. Se for o caso, então passamos a buscar apenas na primeira metade do vetor.

Pesquisa binária e sequencial

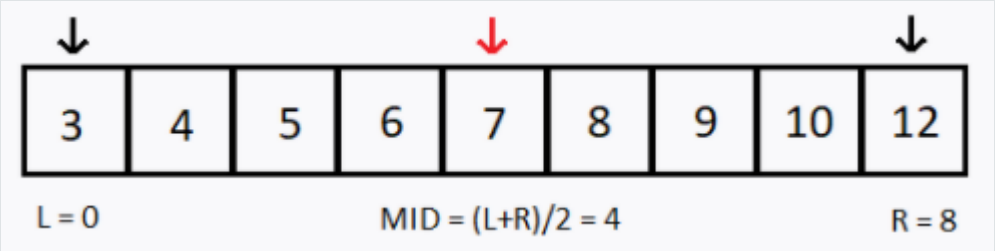
Se não, testamos se o elemento procurado é maior que o elemento do meio do vetor. Se for o caso, então passamos a buscar apenas na segunda metade do vetor.

Caso contrário o valor procurado é igual ao elemento que está no meio do vetor.

Pesquisa binária e sequencial

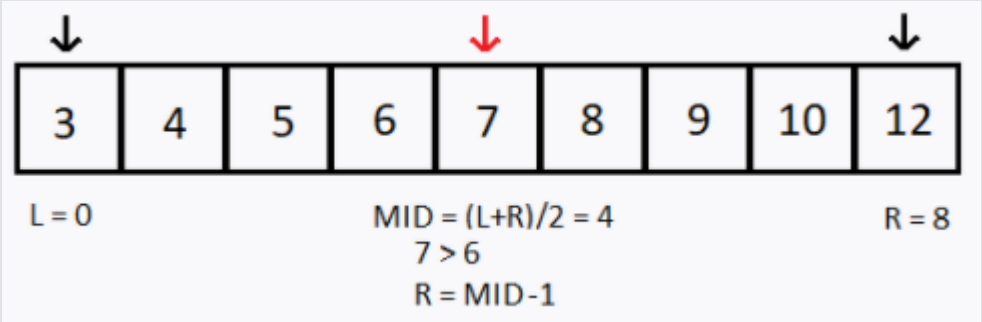
Esse procedimento é repetido até que o elemento seja encontrado ou não haja mais elementos a testar.
Suponha que desejamos buscar o número 6 no mesmo vetor anterior, porém agora foi informado que o vetor está ordenado em ordem crescente, como aplicar a busca binária ?

1º Passo:

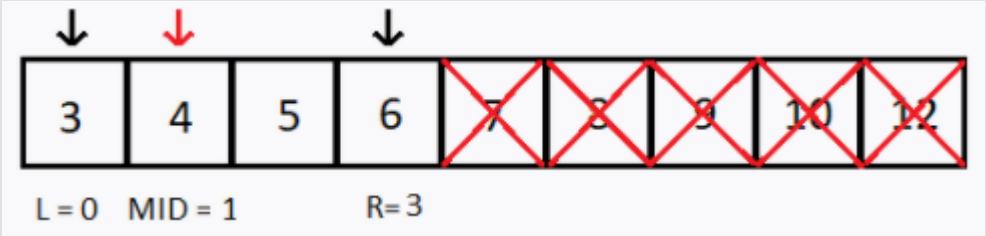


Pesquisa binária e sequencial

2º Passo:

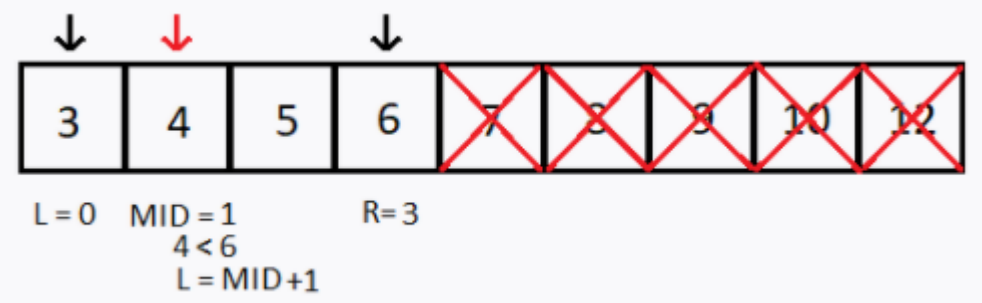


3º Passo:

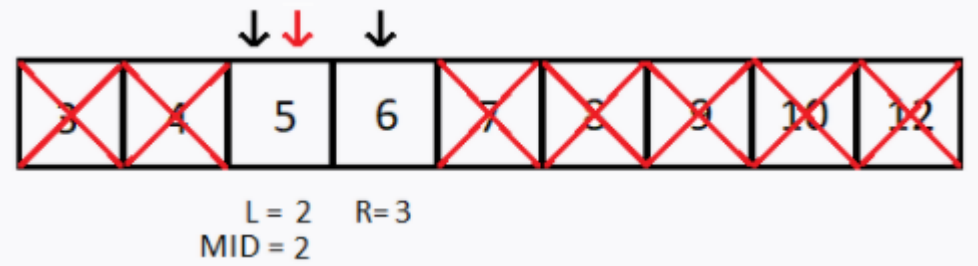


Pesquisa binária e sequencial

4º Passo:

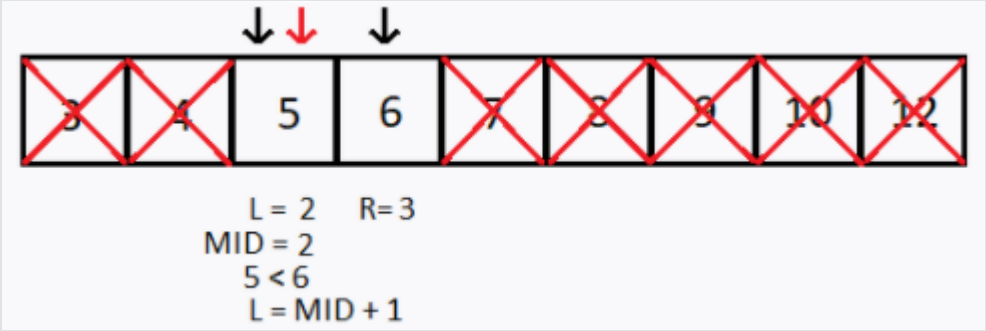


5º Passo:

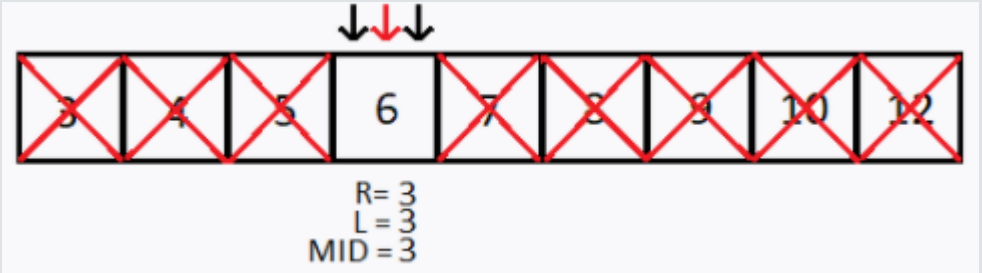


Pesquisa binária e sequencial

6º Passo:

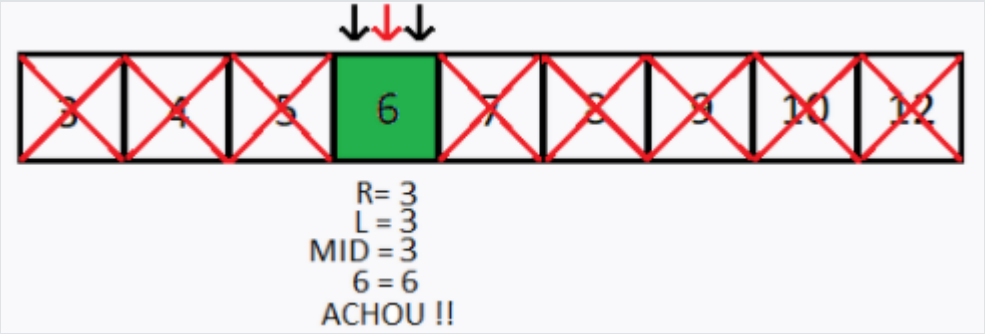


7º Passo:



Pesquisa binária e sequencial

8º Passo:



Observação 1: O vetor precisa estar **ordenado** para conseguir realizar a busca binária.

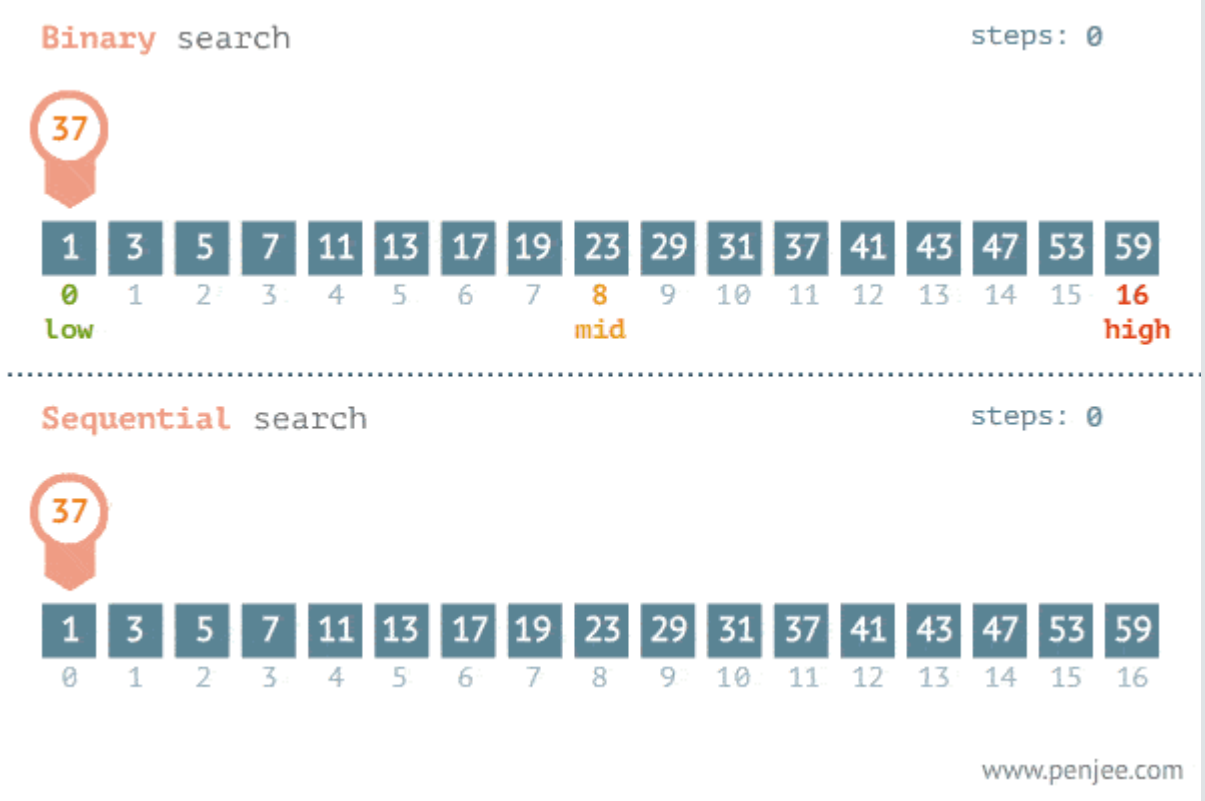
Observação 2: Note que, a cada teste, descartamos uma das metades do (sub)vetor pesquisado.

Pesquisa binária e sequencial

Resumindo, a eficiência da busca binária é muito superior a uma busca linear (no pior caso), pois a cada iteração metade do sub-vetor é descartada. Sua complexidade é de O(log(n)) enquanto a complexidade da busca linear é O(n) (Assunto sobre complexidade será tratado futuramente).

Segue um exemplo da comparação entre as duas buscas.

Pesquisa binária e sequencial



Pesquisa sequencial

A pesquisa sequencial trabalha com grupos de dados ordenados, varre a lista sequencialmente, parando se:

1. O dado foi encontrado ou;
2. O valor atual já é maior que o valor buscado ou ainda;
3. Se chegou ao fim da lista.

Aqui, então, se o valor foi encontrado, retorna o índice onde está o valor; se não foi encontrado, retorna -1.

Exemplo 1 Ex1PesqSequencial

```
import java.util.Scanner;

public class Ex1PesqSequencial {
    public static void main(String[] args) {

        int vetor_tst[]={8,4,0,8,3,4,6,5};
        int fun,x;

        Scanner in = new Scanner(System.in);
```

```
System.out.println("Informe o valor a ser encontrado: ");
        x = in.nextInt();
        fun=buscaSequencial(8,vetor_tst,x);

        if(fun==-1){
            System.out.println("O valor não foi encontrado!");
        }else{
            System.out.println("\n" + "Valor encontrado no índice "
+ fun);
        }
    }
```

```
 static int buscaSequencial(int tamanho, int vetor[],int p)
    {
     int i;
     for(i=0;i<tamanho;i++)
     {
      if(vetor[i]==p){
       return i;
       }
     }
     return -1;
    }
}
```

Exemplo 2 Ex2PesqBinaria

```
import java.util.Scanner;

public class Ex2PesqBinaria {

   static final int TAMANHO=5;
   static Scanner in = new Scanner(System.in);

   public static void main(String[] args) {

      int vet[] =  new int[TAMANHO];
      int opcao, pos, procura;

      //Lê a vetor
      LeVetor(vet);
      //Ordena o vetor
      OrdenaVetor(vet);
      do
      {
         opcao=MontaMenu();

         System.out.println("");
```

Exemplo 2 Ex2PesqBinaria

```
      //Monta o menu com as opcoes
      if (opcao==1){//imprime vetor
         ImprimeVetor(vet);
      }else if (opcao==2){//busca binaria
         System.out.println("\nDigite um valor para pesquisar: ");
         procura = in.nextInt();

         pos=PesquisaBinaria(vet,procura);

         if (pos==-1)
         {
            System.out.println("\n\nValor nao encontrado no vetor!");
         }
         else
         {
            System.out.println("\n\nValor encontrado na posicao " + (pos+1) + "\n");
         }
      }
   }while (opcao!=0);
}
/*-------------------------------------
Efetua a leitura do vetor
-------------------------------------*/
static void LeVetor(int vet[])
{
   int i;
   for (i=0;i<TAMANHO;i++)
   {
      System.out.println("Informe o valor " + (i+1) + ":");
      vet[i] = in.nextInt();
   }
}
```

Exemplo 2 Ex2PesqBinaria

```
/*---------------------------------
   Imprime o vetor na tela
   ---------------------------------*/
   static void ImprimeVetor(int vet[])
   {
      int i;
      for (i=0;i<TAMANHO;i++)
      {
         System.out.println(vet[i]);
      }
   }
   /*---------------------------------------------
   Ordena o vetor utilizando o métido Bubble Sort
   ---------------------------------------------*/
   static void OrdenaVetor(int v[])
   {
      int i,j,aux;
      for (i=0;i<TAMANHO-1;i++)
      {
         for (j=i+1;j<TAMANHO;j++)
         {
            if (v[i] > v[j])
            {
               aux=v[i];
               v[i]=v[j];
               v[j]=aux;
            }
         }
      }
   }
```

Exemplo 2 Ex2PesqBinaria

```
/*-------------------------------------------------------
   Algorítmo de Busca Binária como parâmetro recebe o
   vetor e o valor a ser procurado
   -------------------------------------------------------*/
   static int PesquisaBinaria(int v[], int pesq)
   {
      int intComeco = 0; //Limite inferior (em C o índice inicial é zero)
      int intFinal = TAMANHO-1;  //Limite superior (tamanho do vetor -1
                        // porque o índice inicial é zero )
      int meio;
      while (intComeco <= intFinal)
      {
      //meio = comeco + (final-comeco)/2;
      meio = (intComeco + intFinal)/2;
      if (pesq == v[meio])
         return meio;
      else if (pesq < v[meio])
         intFinal = meio-1;
      else
         intComeco = meio+1;
      }
      return -1;   // não encontrado
   }
      //Exibe o menu de opções na tela
   static int MontaMenu()
   {
      int opcao;
      System.out.println("\nBUSCA BINARIA");
      System.out.println("_____\n");
      System.out.println("Digite a opcao desejada:\n\n");
      System.out.println("[1] Ver o vetor ordenado\n");
      System.out.println("[2] Pesquisar um valor\n");
      System.out.println("[0] Sair\n\n");
      opcao=in.nextInt();
      return opcao;
   }
}
```

Exemplo 3 Ex3PesqBinaria

```
public class Ex3PesqBinaria {

   public static void main(String[] args) {

      Ex3PesqBinaria pesquisarNomes = new Ex3PesqBinaria();

      Pessoa ana = new Pessoa("Ana", 18);
      Pessoa carla = new Pessoa("Carla", 20);
      Pessoa felipe = new Pessoa("Felipe", 24);
      Pessoa patricia = new Pessoa("Patricia", 23);
      Pessoa rafael = new Pessoa("Rafael", 20);

      Pessoa[] pessoas = {ana, carla, felipe, patricia, rafael};

      int inicio = pesquisarNomes.pesquisar('A', pessoas);
      int fim = pesquisarNomes.pesquisar('F', pessoas);

      /* Imprime os nomes encontrados. */
      while(inicio >= 0 && inicio <= fim) {
        System.out.println(pessoas[inicio].getNome());
        inicio++;
      }
    }
```

Exemplo 3 Ex3PesqBinaria

```
public int pesquisar(char letra, Pessoa[] pessoas) {
   int inicio = 0;        //Posição inicial do vetor.
   int meio = 0;          //Posição do meio do vetor.
   int fim = pessoas.length - 1;  //Posição final do vetor.

   /* Enquanto a posição do inicio for menor ou igual a posição
   do fim, procura o valor de x dentro do vetor. */
   while(inicio <= fim) {
    meio = (fim + inicio) / 2; //Encontra o meio do vetor.

    /* Se a primeira letra do nome da pessoa que está no meio
    do vetor for igual a letra procurada, retorna o valor da
    posição do meio do vetor e para de pesquisar. */
    if(pessoas[meio].getNome().charAt(0) == letra) {
     return meio;
    }
```

Exemplo 3 Ex3PesqBinaria

```
        /* Este if serve para diminuir o tamanho do vetor pela métade. */
        /* Se a primeira letra do nome da pessoa que está no meio
        do vetor for menor que o valor da letra procurada, então o
        inicio do vetor será igual a posição do meio + 1. */
        if(pessoas[meio].getNome().charAt(0) < letra) {
          inicio = meio + 1;
        } else {
          /* Se a primeira letra do nome da pessoa que está no meio
            do vetor for maior que o valor da letra procurada, então
            o fim do vetor será igual a posição do meio - 1. */
          fim = meio - 1;
        }
      }

      /* Se não encontrou nenhuma pessoa que tenha a primeira letra
      do nome igual a letra que está sendo procurada, então retorna a
      posição do vetor que possui a letra mais proxima. */
      return fim;
    }
}
```

Exemplo 3 Ex3PesqBinaria

```
public class Pessoa {

   private String nome;
   private int idade;

   public Pessoa(String nome, int idade) {
      this.nome = nome;
      this.idade = idade;
   }

   public String getNome() {
      return nome;
   }

   public void setNome(String nome) {
      this.nome = nome;
   }

   public int getIdade() {
      return idade;
   }

   public void setIdade(int idade) {
      this.idade = idade;
   }
}
```

Vídeo:

Busca Sequencial e binária:

https://www.youtube.com/watch?v=wDDX1npOkyY

Exercício:

1. **Considere o seguinte vetor:**

v = <6, 3, 4, 2, 5, 1>

Responda à seguinte pergunta, quantos passos serão necessários para encontrar o número 5, na ordenação e na busca binária?

2. **Dada a seguinte sequência de nomes (chaves):**

0	1	2	3	4	5	6	7	8
JAIR	VALDIR	CARLOS	JORGE	BIA	ANA	ZÉLIA	MANOEL	CARLA

Ordene esse vetor:

- **por seleção**
- **por inserção e;**
- **por troca**

Depois de ordenado, demonstre e compare as etapas percorridas pela pesquisa binária e para a pesquisa sequencial para encontrar a posição do item cuja chave é: JORGE.

Dicas para Estudo

Seja “CURIOSO”:

- Procure revisar o que foi estudado.
- Pesquise as referências bibliográficas.

Seja “ANTENADO”:

- Leia a próxima aula.

Seja “COLABORATIVO”:

- Traga assuntos relevantes para a sala de aula.
- Participe da aula.
- Proponha discussões relevantes sobre o conteúdo.

Prof. Me. Wilson Lourenço


Dúvidas?
Não mais..