



# Busca Sequencial e Binária



Professor Marcelo Módolo  
Universidade Metodista de São Paulo

# Busca Seqüencial e Binária

- Os algoritmos de busca seqüencial e binária serão explicados a seguir e servirão para complementar os conceitos de eficiência de algoritmos
  - Após explicar os algoritmos de busca retornaremos aos cálculos da função de eficiência
- 

## Busca Seqüencial

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
31	16	45	87	37	99	21	43	10	48	46	77	11	32	28	50

- O vetor acima é formado por 16 números inteiros
- Para saber se um determinado número está no vetor podemos fazer uma busca seqüencial
- Exemplo: buscar o número 46

## Busca Seqüencial: Exemplo 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
31	16	45	87	37	99	21	43	10	48	46	77	11	32	28	50
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑					

- Buscar o número 46
- Algoritmo: Iniciar pela posição 0 e seguir sequencialmente pelas posições subseqüentes até encontrar o número ou terminar o vetor
- O número foi encontrado na posição 10 em 11 iterações

## Busca Seqüencial: Exemplo 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
31	16	45	87	37	99	21	43	10	48	46	77	11	32	28	50
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑

- Buscar o número 55
- A busca inicia pela posição 0 e segue sequencialmente pelas posições subseqüentes até o final do vetor
- O número não foi encontrado no vetor após 16 iterações

## Busca Seqüencial: Exemplo 3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	16	21	28	31	32	37	43	45	46	48	50	77	87	99
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑				

- Buscar o número 46 em um vetor ordenado
- A busca inicia pela posição 0 e segue sequencialmente pelas posições subseqüentes até encontrar o número
- O número foi encontrado na posição 10 nas mesmas 11 iterações do vetor desordenado

## Busca Seqüencial: Exemplo 4

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	16	21	28	31	32	37	43	45	46	48	50	77	87	99
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑

- Buscar o número 55 em um vetor ordenado
- A busca inicia pela posição 0 e segue sequencialmente pelas posições subseqüentes até o final do vetor
- O número não foi encontrado no vetor após as mesmas 16 iterações do vetor desordenado

## Busca NÃO Seqüencial?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	16	21	28	31	32	37	43	45	46	48	50	77	87	99

- Existe uma maneira mais rápida de buscar o número 46 em um vetor ordenado?

## Busca Binária: Exemplo 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	16	21	28	31	32	37	43	45	46	48	50	77	87	99

- Buscar o número 46 em um vetor ordenado

### Algoritmo

1. Armazenar a posição inicial (ini) e a final (fin) do vetor

*No exemplo ini=0 e fin = 15*

## Busca Binária: Exemplo 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	16	21	28	31	32	37	43	45	46	48	50	77	87	99



2. Enquanto o número buscado não for encontrado ou não existirem mais números para comparar:

Encontrar posição central do vetor pela fórmula  $med = (ini + fin)/2$ . Caso o número buscado seja maior:  $ini = med+1$ , caso seja menor:  $fin = med-1$

*No exemplo  $med = (0 + 15)/2 = 7,5 \Rightarrow$  posição 7 e número = 37.*

## Busca Binária: Exemplo 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	16	21	28	31	32	37	43	45	46	48	50	77	87	99

↑
↑
↑
↑

Como  $46 > 37$  então  $ini = med + 1 = 7 + 1 = 8$ . Logo,

$med = (8 + 15)/2 = 11,5 \Rightarrow$  posição 11 e número = 48.

Como  $46 < 48$  então  $fin = med - 1 = 11 - 1 = 10$ . Logo,

$med = (8 + 10)/2 = 9 \Rightarrow$  posição 9 e número = 45.

Como  $46 > 45$  então  $ini = med + 1 = 9 + 1 = 10$ . Logo,

$med = (10 + 10)/2 = 10 \Rightarrow$  posição 10 e número = 46.

## Busca Binária: Exemplo 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	16	21	28	31	32	37	43	45	46	48	50	77	87	99





↑
↑
↑
↑

3. Caso o número buscado tenha sido encontrado retornar sua posição, caso contrário avisar que não foi encontrado

*No exemplo o número foi encontrado na posição 10 em apenas quatro iterações*

## Busca Binária: Exemplo 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	16	21	28	31	32	37	43	45	46	48	50	77	87	99

- Buscar o número 55 em um vetor ordenado
- A busca inicia pela posição 0 e segue sequencialmente pelas posições subseqüentes até o final do vetor
- O número não foi encontrado no vetor

## Busca Seqüencial: Algoritmo

1. Iniciar pela posição inicial do vetor
2. Enquanto o número buscado não for encontrado ou não existirem mais números para comparar, seguir buscando na posição subseqüente do vetor
3. Caso o número buscado tenha sido encontrado retornar sua posição, caso contrário avisar que não foi encontrado

## Busca Binária: Algoritmo

1. Armazenar a posição inicial (ini) e a final (fin) do vetor
2. Enquanto o número buscado não for encontrado ou não existirem mais números para comparar:  
Encontrar posição central (med) do vetor pela fórmula  $med = (ini + fin)/2$ . Caso o número buscado seja maior:  $ini = med + 1$ , caso seja menor :  $fin = med - 1$
3. Caso o número buscado tenha sido encontrado retornar sua posição, caso contrário avisar que não foi encontrado

## Cenários

- **Melhor caso:** menor custo de execução do algoritmo sobre todas as entradas de tamanho n
- **Pior caso:** maior custo de execução do algoritmo sobre todas as entradas de tamanho n
- **Caso médio:** média dos custos de execução do algoritmo sobre todas as entradas de tamanho n



## Cenários da Busca Seqüencial

- Melhor caso:  $f(n) = 1$
- Pior caso:  $f(n) = n$
- Caso médio:  $f(n) = (n + 1)/2$



## Cenários da Busca Binária

- Melhor caso:  $f(n) = 1$
- Pior caso:  $f(n) = 1 + \log_2 n$
- Caso médio: (como calcular?)



## Busca Seqüencial x Binária

- A busca seqüencial tem  $O(n)$  enquanto a busca binária tem  $O(\log_2 n)$
- A busca binária é muito mais rápida que a busca seqüencial, mas ela exige que os dados estejam ordenados
- Para ordenar os dados existem vários métodos de ordenação que podem ser melhores em determinadas situações
- Alguns desses métodos de ordenação serão estudados

## Exercício

- Analisar o programa em Java para cada um dos algoritmos de busca: seqüencial e binária
- Encontrar a função de eficiência para cada um desses programas considerando os casos: pior, melhor e médio
- Transformar a função encontrada para a notação 'O'

## Busca Seqüencial: Algoritmo

1. Iniciar pela posição inicial do vetor
2. Enquanto o número buscado não for encontrado e existirem elementos no vetor, seguir buscando na posição subsequente do vetor
3. Caso o número buscado tenha sido encontrado retornar sua posição, caso contrário avisar que não foi encontrado

## Busca Binária: Algoritmo

1. Armazenar a posição inicial (ini) e a final (fin) do vetor
2. Enquanto o número buscado (num) não for encontrado e existirem elementos para comparação: encontrar posição central (med) do vetor pela fórmula  $med = (ini + fin)/2$  e comparar med com num:
  - a) Caso  $num > vetor[med]$ :  $ini = med + 1$ ;
  - b) Caso contrário:  $fin = med - 1$ ;
3. Caso o número buscado tenha sido encontrado retornar sua posição, caso contrário avisar que não foi encontrado

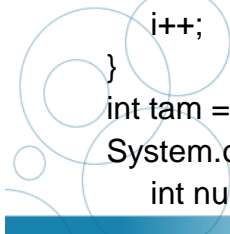
## Referências Bibliográficas

- TENEMBAUM, Aaron M. *Estrutura de dados usando C*. São Paulo: Makron Books, 1995.
- MORAES, Carlos Roberto. *Estrutura de Dados e Algoritmos: uma abordagem didática*. 2. ed. São Paulo: Futura, 2003.



## Busca Seqüencial

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    Arquivo entra = new Arquivo("r", "d:/entradaORD.txt");  
    int vetor[] = new int[100000];  
    int i = 0;  
    while (!entra.fda()) {  
        vetor[i] = entra.leiaInt();  
        System.out.println(vetor[i]);  
        i++;  
    }  
    int tam = i;  
    System.out.println("Digite o número buscado");  
    int num = sc.nextInt();
```



## Busca Seqüencial

```

int cont = 0;
i = 0;
while (i < tam && vetor[i] != num) {
    i++;
    cont++;
}
if (i < tam) {
    System.out.println("Número encontrado na posição "
        + i + " após " + cont + " iterações");
} else {
    System.out.println("Número NÃO encontrado após "
        + i + " iterações");
}
}

```

## Busca Binária

```

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Arquivo entra = new Arquivo("r", "d:/entradaORD.txt");
    int vetor[] = new int[100000];
    int i = 0;
    while (!entra.fda()) {
        vetor[i] = entra.leiaInt();
        System.out.println(vetor[i]);
        i++;
    }
    int tam = i;
    System.out.println("Digite o número buscado");
    int num = sc.nextInt();
}


```

## Busca Binária

```

int cont = 0,
int med, ini = 0, fin = tam - 1;
med = (ini + fin) / 2;
while (ini <= fin && vetor[med] != num) {
    if (num > vetor[med]) {
        ini = med + 1;
    } else {
        fin = med - 1;
    }
    med = (ini + fin) / 2;
    cont++;
}

```



## Busca Binária

```

if (ini <= fin) {
    System.out.println("Número encontrado na posição "
        + med + " após " + cont + " iterações");
} else {
    System.out.println("Número NÃO encontrado após "
        + cont + " iterações");
}

```

