COMPUTAÇÃO
UFBA

Universidade Federal
da Bahia

Laboratório de Programação I (MATA57)

Prof.: Claudio Junior N. da Silva (claudiojns@ufba.br)

Busca Binária

2023.1

Problema I – *binary_search()*

- Dado um vector int de tamanho T, escreva um programa em C++ que verifique se um determinado número N se encontra nesse vector;
- Quais problemas?

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
  vector<int> v;
  int i, j;
  for(i=0; i < 5; i++) {
    cin >> j;
    v.push_back(j);
  }
  stable_sort (v.begin(), v.end());
  cin >> j;
  if( binary_search(v.begin(), v.end(), j) )
        cout << "Numero encontrado";
  return 0;
}
```

Problema II – *lower_bound / upper_bound*

- Dado um vector int de tamanho T, escreva um programa em C++ retorne:
  - 1° elemento que seja maior ou igual a um valor N qualquer;
  - 1° elemento que seja maior do que um valor N qualquer;
- Quais problemas?

```
#include <iostream>
#include <algorithm>    // lower_bound, upper_bound, sort
#include <vector>       // vector

using namespace std;

int main () {
  vector<int> meuVector;
  meuVector = {80,10,60,50,40,30,70,20};
  sort (meuVector.begin(), meuVector.end()); // 10,20,30,40,50,60,70,80
  vector<int>::iterator low,up;

  // retorna o iterator para o primeiro elemento que seja maior ou igual a
  low=lower_bound (meuVector.begin(), meuVector.end(), 10);

  // retorna o iterator para o primeiro elemento que seja maior  a
  up= upper_bound (meuVector.begin(), meuVector.end(), 10);

  cout << "lower_bound na posição " << (low- meuVector.begin()) << endl;
  cout << "upper_bound na posição " << (up - meuVector.begin()) << endl;
  return 0;
}
```

Problema III – *busca binária*

- Dado um determinado vector de inteiros com tamanho T, escreva um programa em C++ que contenha uma função de busca binária a qual retorna:
  - -1 quando o elemento não for encontrado no vector, ou;
  - Caso o elemento exista no vector, retorne sua posição.

Problema III – *busca binária*

1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	5	8	12	33	21	22	25	31	44	46	49	51	53	55

2

8	9	10	11	12	13	14
31	44	46	49	51	53	55

3

12	13	14
51	53	55

	esquerda	direita	meio (esquerda + (direita – esquerda)/2)	N	vetor(meio)
1	0	14	0 + (14 -0) / 2 = 7	53	25
2	meio + 1 = 8	14	8 + (14-8) / 2 = 11	53	49
3	Meio + 1 = 12	14	12 + (14-12) / 2 = 13	53	53

Problema III – *busca binária*

1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	5	8	12	33	21	22	25	31	44	46	49	51	53	55

2

8	9	10	11	12	13	14
31	44	46	49	51	53	55

3

8	9	10
21	44	46

	esquerda	direita	meio (esquerda + (direita – esquerda)/2)	N	vetor(meio)
1	0	14	0 + (14 -0) / 2 = 7	44	25
2	meio + 1 = 8	14	8 + (14-8) / 2 = 11	44	49
3	8	meio – 1 = 10	8 + (10 – 8)/2 = 9	44	44

Busca Binária

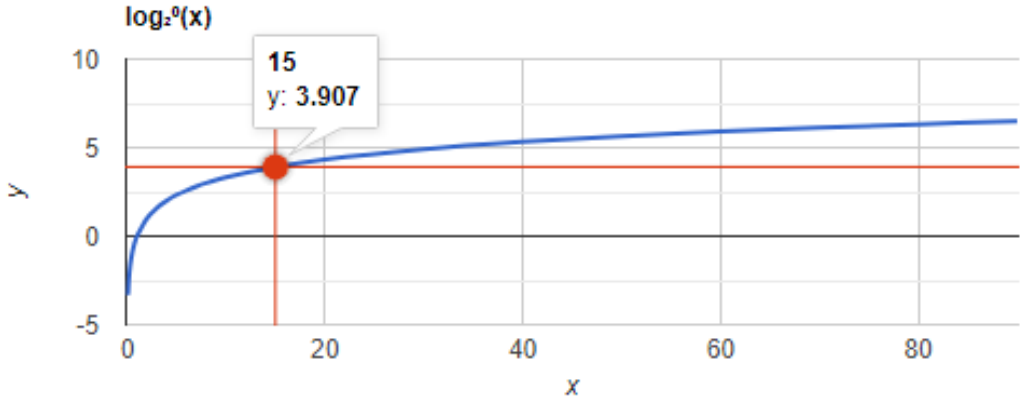
```cpp
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;

int buscaBinaria(vector<int>& vetor, int valorProcurado) {
    int esquerda = 0;
    int direita = vetor.size() - 1;
    while (esquerda <= direita) {
        int meio = esquerda + (direita - esquerda) / 2;
        if (vetor[meio] == valorProcurado) {
            return meio;
        }
        if (vetor[meio] < valorProcurado) {
            // O valor está à direita do meio
            esquerda = meio + 1;
        } else {
            // O valor está à esquerda do meio
            direita = meio - 1;
        }
    }
    return -1; // O valor não foi encontrado
}
```

```cpp
int main(void){
    int n =5, a;
    vector<int> vTeste;
    cout << "Informe os elementos do Vector..: " ;
    int numero;
    for(int i=0;i<n;i++){
       cin >> numero;
       vTeste.push_back(numero);
    }
    sort(vTeste.begin(), vTeste.end());
    cout << endl;
    cout << "Informe o número a ser procurado: " ;
    cin >> a;
    cout << endl;
    int indice = buscaBinaria(vTeste, a);
    if (indice ==-1){
        cout << "O elemento não foi encontrado.";  }
    else {
        cout << "O elemento foi encontrado"     }
    return 0;
}
```

Busca Binária

- Quais as vantagens?
- Complexidade $\Theta(\log_2 n)$



	Busca Linar	Busca Binária
Melhor Caso	$O(1)$	$O(1)$
Médio Caso	$O(n+1)/2$	$O(\log n)$
Pior Caso	$O(n)$	$O(\log n)$