

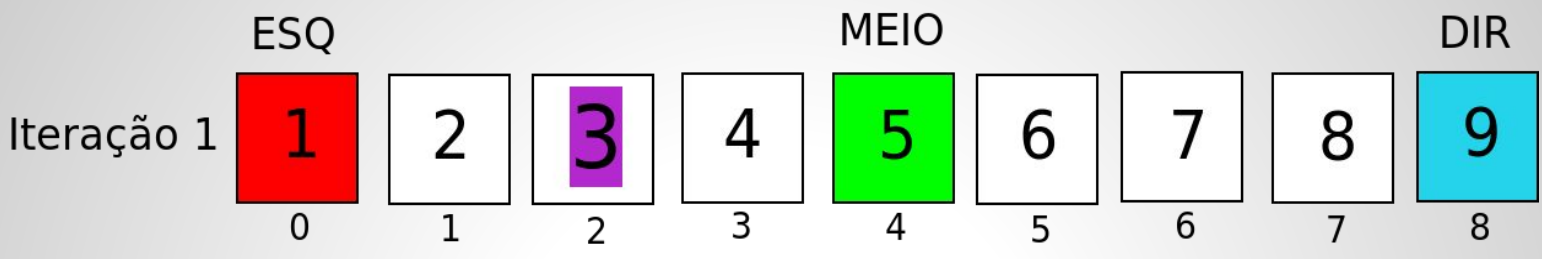


# Curso de Programação Nível Intermediário

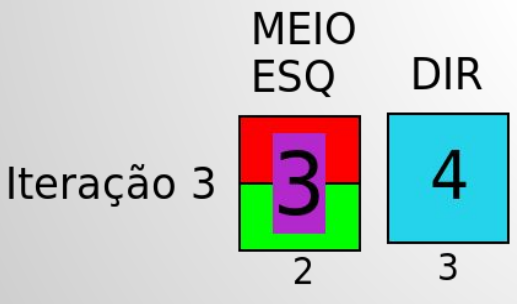


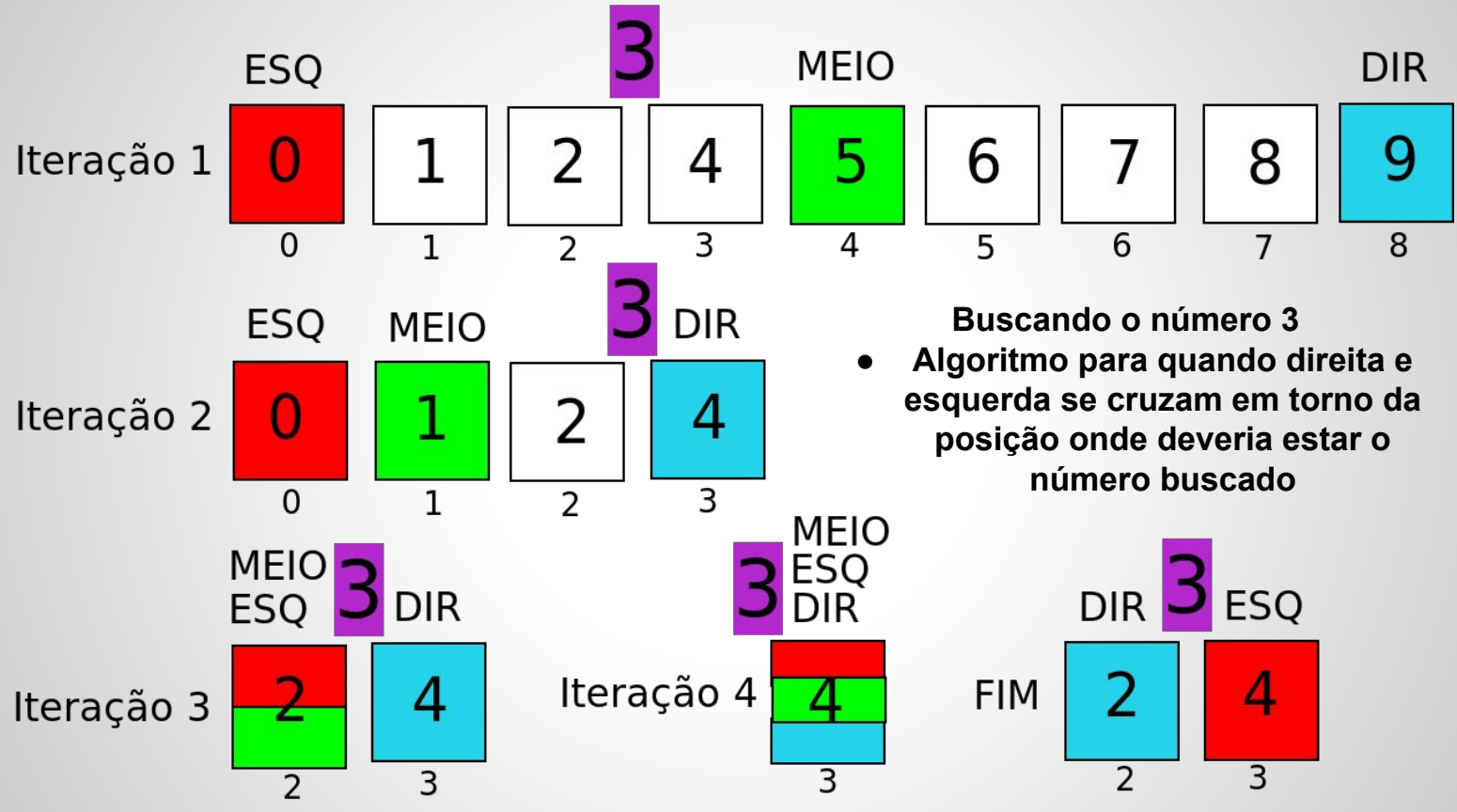
**Universidade Federal da Bahia  
Instituto de Computação  
Departamento de Ciência da Computação**

# **AULA 3 - BUSCA BINÁRIA**



- Buscando o número 3
- Algoritmo para quando o número buscado coincide com o meio





# Busca Binária Simples

- Descrição
  - Checar se um conjunto de números está no vetor.
- Entrada
  - Um inteiro N representando o tamanho do vetor, N inteiros do vetor em ordem crescente, um inteiro M indicando a quantidade de casos de teste, M inteiros como casos de teste.
- Saída
  - "SIM" quando um caso de teste está no vetor, "NAO" caso contrário.

# Busca Binária Simples

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

int main() {
    int i, j, N, M, p, esq, dir, meio;

    cin >> N;
    int v[N];
    for(i=0; i < N; i++) //Lê vetor ordenado
        cin >> v[i];

    cin >> M;
```

```
    for(i=0; i < M; i++) {
        cin >> p; // Número buscado
        esq=0; dir=N-1;
        while(esq <= dir) { // Busca binária
            meio = (esq + dir)/2;
            if(p == v[meio]) // achou
                break;
            if(p < v[meio]) // Joga fora metade direita
                dir = meio-1;
            else // Joga fora metade esquerda
                esq = meio+1;
        }
        if(v[meio] == p) cout << "SIM" << endl;
        else cout << "NAO" << endl;
    }
    return 0;
}
```

# Busca Binária Simples

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

int main() {
    int i, N, M, num;
    cin >> N;
    vector<int> v(N);
    for(i=0; i < N; i++) //Lê vetor ordenado
        cin >> v[i];

    cin >> M;

    for(i=0; i < M; i++) {
        cin >> num;        // Número buscado
        if( binary_search(v.begin(), v.end(), num) == true)
            cout << "SIM" << endl;
        else
            cout << "NAO" << endl;
    }
}
```

# Ordenação - binary\_search

```
struct pessoa {
    int id;
    string nome;
};

bool cmp(pessoa i, pessoa j) {
    return (i.id < j.id || i.id == j.id && i.nome < j.nome);
}

int main() {
    vector<pessoa> v;
    ...
    stable_sort (v.begin(), v.end(), cmp);
    pessoa j;
    ...
    bool r = binary_search(v.begin(), v.end(), j, cmp); // pode usar uma função de comparação, assim como o sort
}
```



# Vetores - binary\_search

Saiba mais em:

[http://www.cplusplus.com/reference/algorithm/binary\\_search/](http://www.cplusplus.com/reference/algorithm/binary_search/)

# Ordenação - lower\_bound

```
#include <iostream>
#include <vector>
#include <utility>
#include <algorithm>
using namespace std; // lower_bound -> std
int main() {
    vector<int> v;
    int i, j;
    for(i=0; i < 1000; i++) {
        cin >> j;
        v.push_back(j);
    }

    stable_sort (v.begin(), v.end());
    cin >> j;
    vector<int>::iterator it;
    it = lower_bound(v.begin(), v.end(), j);
}

// lower_bound - retorna iterator para o primeiro
// elemento que seja maior ou igual a j

// para saber a posição, faça it-v.begin()
```

# Ordenação - upper\_bound

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std; // upper_bound -> std
int main() {
    vector<int> v;
    int i, j;
    for(i=0; i < 1000; i++) {
        cin >> j;
        v.push_back(j);
    }
```

```
    stable_sort (v.begin(), v.end());
    cin >> j;
    vector<int>::iterator it;
    it = upper_bound(v.begin(), v.end(), j);
}
// upper_bound - retorna iterator para o primeiro
// elemento que seja maior que j

// para saber a posição, faça it-v.begin()
```

# Vetores - {lower,upper}\_bound

Saiba mais em:

[http://www.cplusplus.com/reference/algorithm/lower\\_search/](http://www.cplusplus.com/reference/algorithm/lower_search/)

[http://www.cplusplus.com/reference/algorithm/upper\\_search/](http://www.cplusplus.com/reference/algorithm/upper_search/)

# Ordenação - binary\_search

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

int main() {
    vector<int> v;
    int i, j;
    for(i=0; i < 1000; i++) {
        cin >> j;
        v.push_back(j);
    }

    stable_sort (v.begin(), v.end());
    cin >> j;
    if( binary_search(v.begin(), v.end(), j) )
        cout << "Tá lá!\n";
    }

    // binary_search
    // retorna true caso o elemento exista no vetor
    // retorna false caso contrário
```