**ICMC** - Instituto de Ciências Matemáticas e de Computação

**BCC** - Bacharelado em Ciências da Computação

**Disciplina**: Laboratório de Introdução a Ciência da Computação

**Docente**: Leonardo Pereira

**Aluno**: Bernardo Marques Costa (**Número USP** 11795551)

**ANÁLISE DE ALGORITMO E COMPLEXIDADE: BUSCAS**

**INTRODUÇÃO**

Neste trabalho de Laboratório de Introdução a Ciência da Computação, deverá ser implementado os algoritmos de busca: busca sequencial e busca binária, sendo esta realizada em 2 modelos, iterativo e recursivo.

O programa receberá o total de buscas que serão realizadas e o tamanho de um registro que armazenará chaves identificadoras de filmes e séries no sistema da Netflix. Em seguida, será dado na mesma entrada cada uma destas chaves em ordem crescente. Por fim, será requisitado um modo de busca e uma chave.

O programa receberá a seguinte entrada:

|  |
| --- |
| < número de registros > <número de buscas > < registro[0] >  .... < registro[n] > < identificador da busca > < chave buscada > |

**ALGORITMO BUSCA SEQUENCIAL**

|  |
| --- |
| **int sequentialSearch(int \*array, int key, int size){  for(int i = 0; i < size; ++i)  if(key == array[i]) return i;  return -1; }** |

Podemos observar que se trata de um algoritmo simples, que realiza uma busca de um vetor indice por índice, sendo o formato da função de crescimento de operações linear. Desta forma, denotamos sua complexidade como O(n).

**Melhor caso:** temos para o melhor caso a situação quando o primeiro índice do vetor é a própria chave buscada.

**Pior caso:** chave não está no vetor, ou seja, quando o programa deverá realizar uma comparação a mais que o total de índices do vetor, não encontrando a chave buscada.

**ALGORITMO BUSCA BINÁRIA**

**> Iterativa**

|  |
| --- |
| **int binarySearchIterative(int \*array, int key, int start, int end){  int middle;**  **while(start <= end){  middle = start + (end - start) / 2;  if(array[middle] == key) return middle;  else if(array[middle] > key) end = middle - 1;  else if(array[middle] < key) start = middle + 1;**  **}  return -1; }** |

**> Recursiva**

|  |
| --- |
| **int binarySearchRecursive(int \*array, int key, int start, int end){**  **if(start <= end){  int middle = start + (end - start) / 2;**  **if(array[middle] == key)**  **return middle;**  **else if(array[middle] > key)**  **return binarySearchRecursive(array, key, start, middle - 1);    else if(array[middle] < key)**  **return binarySearchRecursive(array, key, middle + 1, end);  }    return -1; }** |

Sendo um algoritmo de busca em conquista, temos a tendência de otimizar a busca de uma chave em uma lista, **uma vez que esta lista esteja ordenada**, eliminando a cada passo e chamada da função uma fração do vetor.

Desta forma, após analisar o algoritmo verificamos uma complexidade O(log n), sendo muito mais eficiente que uma busca sequencial.

**Melhor caso:** ao contrário da busca sequencial, o melhor caso para a busca binária, tanto recursiva quanto iterativa, é a situação em que a chave está localizada no meio do vetor.

**Pior caso:** chave não encontrada no vetor, rodando todas as vezes e retornando o valor de -1, identificador de chave inexistente.