Pesquisa em Memória Primária

Ciência da Computação
Laboratório de Ordenação e Pesquisa
Prof. M.Sc. Elias Gonçalves

Pesquisa em Memória Primária

Objetivos:

→ Encontrar item cuja chave seja igual a do elemento dado na pesquisa.

→ Recuperar item com uma determinada chave.

→ Recuperar dados armazenados em uma base de dados.

Questões envolvidas na escolha do método de pesquisa:

→ Qual a quantidade de dados no conjunto?

→ Qual é a frequência com que operações de inserção e retirada de dados são realizadas?

→ Os dados estão estruturados?

→ Os dados estão ordenados?

→ Há valores repetidos?

Pesquisa em Memória Primária

Métodos de pesquisa:

→ Pesquisa sequencial;

→ Pesquisa binária;

→ Árvore de pesquisa;

→ Pesquisa digital;

Pesquisa Sequencial Simples

→ Pesquisa sequencialmente a partir do primeiro índice do vetor até encontrar a chave procurada.

→ O conjunto de dados não precisa estar ordenado.

→ Encontra uma única chave.

Quantas vezes a comparação é executada?

→ Melhor caso: O(1), encontra na 1ª posição.

→ Pior caso: O(n), encontra na última posição.

→ Custo sem sucesso: O(n+1), executa n+1 vezes.

Pesquisa Sequencial

```c
/*
 * Verifica se a chave está em um vetor.
 * Retorna a posicao em que ela se encontra caso exista.
 * Retorna -1 caso não exista.
*/
int pesquisaSequencial( int *v, int n, int chave ){
    for( int i=0; i<n; i++ )
        if( v[i] == chave )
            return i; // encontrou

    return -1; // nao encontrou
}
```

Pesquisa Sequencial Melhorada

→ Pesquisa sequencialmente a partir do primeiro índice do vetor;

→ Se o elemento na posição atual não for a chave procurada, verifica se o valor do elemento naquela posição é maior que a chave procurada.

→ Diminui o número de comparações, pois o elemento sendo maior que a chave encerra a busca.

→ O conjunto de dados precisa estar ordenado.

→ Encontra uma única chave.

Quantas vezes a comparação é executada?

→ Melhor caso: O(1), encontra na 1ª posição.

→ Pior caso: O(n), encontra na última posição.

→ Custo sem sucesso: O(n+1), executa n+1 vezes.

Pesquisa Sequencial Ordenada

```c
/*
 * Verifica se a chave está em um vetor.
 * Retorna a posicao em que ela se encontra caso exista.
 * Retorna -1 caso não exista.
 * Considera o vetor ordenado.
*/
int pesquisaOrdenada( int *v, int n, int chave ){
    for( int i=0; i<n; i++ ){
        if( v[i] == chave )
            return i; // encontrou
        else if( v[i] > chave )
            return -1; // não existe, para a pesquisa
    }

    return -1; // nao encontrou
}
```

Pesquisa Binária

→ Busca sempre o elemento do meio do conjunto de dados;

→ Em cada comparação o conjunto de dados é dividido em dois, reduzindo o tempo de busca.

→ O conjunto de dados precisa estar ordenado.

→ Encontra uma única chave.

Quantas vezes a comparação é executada?

→ Melhor caso: O(1), encontra no meio do vetor;

→ Pior caso: O(log2(n)) vezes;

→ Custo sem sucesso: O(log2(n) + 1) vezes.

Pesquisa Binária

```c
int pesquisaBinaria(int *v, int inicio, int fim, int chave ){
    // Calcula o meio do vetor
    if ( fim >= inicio ) {
        int meio = inicio + ( fim-inicio) / 2;

        // Encontrou a chave
        if ( v[meio] == chave )
            return meio;

        // Pesquisa na primeira parte do vetor
        if ( v[meio] > chave )
            return pesquisaBinaria( v, inicio, meio-1, chave );

        // Pesquisa na segunda parte do vetor
        return pesquisaBinaria( v, meio+1, fim, chave );
    }

    // nao encontrou a chave no vetor
    return -1;
}
```