

[A2E2] - Busca e Ordenação

Total de pontos 2/3 ?

0 de 0 pontos

Objetivos

- escolher o melhor método de busca para seus algoritmos;
- escolher o melhor método de ordenação para seus algoritmos;
- diferenciar entre os diversos métodos de busca e classificação.

Nome Completo *

HARLLEM ALVES DO NASCIMENTO

Algoritmos de Busca

1 de 2 pontos

Busca Linear e Busca Binária

Busca Linear (sequencial)

Podemos procurar um valor desejado em um vetor por meio da pesquisa sequencial. Esta busca começa no primeiro elemento e percorre todo o vetor até o fim, localizando o valor escolhido. É a forma de busca mais simples, porém menos eficiente.

- melhor caso: o elemento é o primeiro do vetor - não precisa percorrer todo o vetor
- caso médio: o elemento está no vetor, mas não é o primeiro - o valor é encontrado após percorrer parte do vetor
- pior caso: o elemento não está no vetor - percorre todo o vetor, mas não encontra o valor.



código exemplo

```
#include <stdio.h>
#include <stdlib.h>
main( )
{
    printf("Busca Linear\n");

    int numeros[5]={50,20,10,70,15};
    int i, valor;

    printf("Qual eh o valor a procurar?\n");
    scanf("%d", &valor);

    for (i=0;i<5;i++){
        if (numeros[i]==valor){
            printf("Valor encontrado!");
            exit(1);
        }
    }

    printf("Valor nao encontrado");

    return(0);
}
```

Atenção para as linhas:

```
int numeros[5]={50,20,10,70,15}; //nosso vetor possui 5 elementos
printf("Valor encontrado!"); exit(1); //ao comparar e encontrar o valor, uma mensagem é exibida e a
execução para, ao ser chamada a função existe(1) da <stdlib.h>
```

Esse tipo de pesquisa pode ser ineficiente se tivermos um vetor de muitos elementos. Nesse casos, a busca binária pode ser utilizada.

Busca Binária

Ao contrário da busca linear, a busca binária procura otimizar o processo de procura por determinado elemento de um vetor ORDENADO. Essa busca considera que é possível sempre dividir o conjunto de elementos em duas partes (metade).



Analogia com a busca por uma palavra em um dicionário.



Exemplo (analogia)

Se buscarmos a palavra natureza em um dicionário, seria uma boa estratégia abrir o dicionário no meio e decidir se a palavra procurada está antes ou depois desse ponto. No caso desse exemplo já descartamos a parte inferior do livro. A seguir, tomamos a metade restante e repetimos o mesmo processo, dividimos no meio mais uma vez, e assim por diante. Ao final de algumas iterações, localizaremos o termo natureza.

Importante: o vetor tem que estar ordenado de forma crescente ou decrescente. Consideramos a busca por uma palavra em um dicionário justamente por ser um objeto organizado de forma ordenada (alfabética).

- melhor caso: o valor é encontrado logo na primeira divisão do vetor
- caso médio: o valor é encontrado após a partir da segunda iteração
- pior caso: o valor não é encontrado



código exemplo - função PesquisaBinaria

```
int PesquisaBinaria (int vet[], int chave, int tam)
{
    int inf = 0;    // limite inferior (o primeiro índice de vetor em C é zero    )
    int sup = tam-1; // limite superior (termina em um número a menos. 0 a 9 são 10 números)
    int meio;

    while (inf <= sup)
    {
        meio = (inf + sup)/2;

        if (chave == vet[meio]){

            return meio;

        }else
        if (chave < vet[meio]){

            sup = meio-1;
        }
        else{

            inf = meio+1;
        }
    }
    return -1; // não encontrado
}
```

Utilizando a função acima

```
main( )
{
    printf("Busca Linear\n");

    int numeros[10]={1,2,3,4,5,6,7,8,9,10};
    int i, valor;

    printf("Qual eh o valor a procurar?\n");
    scanf("%d", &valor);

    int e;
    e = PesquisaBinaria (numeros, valor, 10);

    if (e != -1){
        printf("Achou na posicao %d!", e);
    }

    return(0);
}
```



✓ O que a função pesquisa binária retorna, quando o valor buscado é encontrado? *

1/1

- ☒ a posição do valor no vetor
- ☐ o valor encontrado
- ☐ -1



✗ Considerando a entrada 6 para o código anterior, qual o retorno da função? *

0/1

- ☒ 6
- ☐ 5
- ☐ 7
- ☐ -1



Resposta correta

- ☒ 5

Algoritmos de Ordenação e Classificação

1 de 1 pontos

Existem vários métodos de ordenação e classificação. Um desses é o BubbleSort, que por ser o mais simples não é tão eficiente. Recomendado para vetores de 30 elementos.

Observem que no caso da busca binária é necessário que o vetor esteja ordenado, faz-se necessário portanto métodos de ordenação de vetores. Veja um exemplo da execução do algoritmo Bolha.



O algoritmo compara o valor da posição atual do vetor, como o valor da próxima posição, levando sempre o maior valor para a posição da direita. Veja esta animação.

6 5 3 1 8 7 2 4

exemplo - vídeo

[A2V3] - Ordenação



código do vídeo

```
#include <stdio.h>
```

```
main( )  
{
```

```
    int vet[5]={5,3,1,2,4};  
    int i,j, aux;
```

```
    for (i=0;i<5;i++){  
        printf("%d, ",vet[i]);  
    }
```

```
    printf("\n");
```

```
    for (j=0;j<4;j++){  
        for (i=0;i<5;i++){  
            printf("%d, ",vet[i]);  
        }
```

```
        printf("O valor de j eh %d:\n",j);  
        printf("\n");  
        for (i=0;i<4;i++){  
            printf("O valor de i eh %d:\n",i);  
            printf("O valor atual eh %d, e o prox eh %d\n",vet[i], vet[i+1]);
```

```
            if(vet[i]>vet[i+1]){  
                aux=vet[i];  
                vet[i]=vet[i+1];  
                vet[i+1]=aux;  
                printf("trocou\n");  
            }
```

```
            printf("O valor atualizado é eh %d, e o prox eh %d\n",vet[i], vet[i+1]);  
            printf("\n");
```

```
        }  
    }
```

```
    for (i=0;i<5;i++){  
        printf("%d, ",vet[i]);  
    }
```

```
    return(0);  
}
```



✓ Conforme o vídeo , quando passamos um vetor em uma função acontece a:

1/1

- ☐ chamada por valor
- ☒ chamada por referência
- ☐ não é possível



Juntando as peças

0 de 0 pontos

Atividade



Crie um programa que receba como entrada 10 valores que serão armazenados em um vetor, depois a entrada de mais um valor. Após isso o programa deverá buscar se o último valor inserido está no vetor. Para resolver esta questão utilize funções e procedimentos, bem como os algoritmos BubbleSort, para ordenar o vetor, e a busca binária apresentados neste material, para buscar o valor. *

```
#include <stdlib.h>
#include <locale.h>
#define len 10

int BubbleSort(int arr[], int lenVetor){
    int temp;
    for(int i = 0; i < lenVetor; i++){
        for(int j = 0; j < lenVetor; j++){
            if(arr[j] > arr[j+1]){
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
    return 0;
}

int buscaBinaria (int key, int lenVet, int v[]) {
    //O left começa como -1 para regularizar o tamanho do vetor
    int left = -1, right = lenVet;
    //Vai ficar rodando enquanto a posição de left, right e mid não forem iguais
    while (left < right-1) {
        //Calcula o valor do meio do vetor
        int mid = (left + right)/2;
        //Verifica se o valor do meio é menor que a chave
        if (v[mid] < key){
            //Se for menor, então a primeira posição se torna a do meio
            left = mid;
        }
        //Se não for menor, então a posição do meio se torna a ultima
        else{
            right = mid;
        }
    }
    //Quando left, mid e right são iguais, o laço é quebrado e retorna a posição
    return right;
}

int main(void){
    system("cls");
    setlocale(LC_ALL, "Portuguese");

    int values[len], verify, pos;
```



```
printf("**Insira 10 Valores: \n\n");
for(int i = 0; i < len; i++){
    printf("Valor[%d]: ", i + 1);
    scanf("%d", &values[i]);
}
system("cls");
printf("**Procure um valor no Array Informado!!(Busca Binária)\n\n");
printf("Digite o valor para Buscar: ");
scanf("%d", &verify);

//Organiza vetor do maior para menor:
BubbleSort(values, len);

//Realiza Busca Binaria
pos = buscaBinaria(verify, len, values);

//Resultado
printf("\n**Resultado: ");
printf("O valor %d foi encontrado na posição %d ", verify, pos + 1);

printf("\n");
return 0;
}
```

Este formulário foi criado fora de seu domínio. - [Termos de Serviço](#) - [Política de Privacidade](#)

Google Formulários

