Pesquisa em Vetor

SSC300- Linguagens de Programação e Aplicações

Pesquisa

- O problema de procurar (pesquisar) alguma informação numa tabela ou num catálogo é muito comum
- Exemplo:
  - procurar o telefone de uma pessoa no catálogo
  - procurar o nº da conta de um certo cliente
  - consultar um determinado saldo em um terminal automático

19

Pesquisa

- A tarefa de "pesquisa", "procura" ou "busca" é, como se pode imaginar, *uma função muito utilizada*
  - A pesquisa é aplicada a elementos organizados em diferentes estruturas (vetores, matrizes, registros, arquivos...)
- As rotinas que executam a busca devem ser eficientes (menor tempo possível)

20

Pesquisa

EFICIÊNCIA

- O TEMPO GASTO pesquisando dados em tabelas depende do TAMANHO da tabela e do ALGORITMO utilizado na busca.

21

Algoritmos de Pesquisa

- Pesquisa Seqüencial
- Pesquisa Binária

Algoritmo de Pesquisa

- Para os algoritmos de pesquisa que se seguem vamos utilizar:
  - TAB - um vetor contendo N elementos inteiros distintos
  - DADO - elemento a ser procurado em TAB
  - ACHOU - indica o sucesso ou falha na pesquisa
  - POS - aponta para a posição do elemento encontrado

Pesquisa Seqüencial

Pesquisa Seqüencial

- Comparar o elemento procurado (DADO) com cada um dos elementos da tabela TAB na seqüência em que aparecem na tabela

Tabela(TAB)	Elemento Procurado(DADO)
3 6 1 2 0	2
2 2 2 2	POS ← 3 ACHOU ← V

Pesquisa Seqüencial

- A idéia básica da Pesquisa Seqüencial é localizar o elemento procurado através de comparações sucessivas e seqüenciais, a partir do primeiro elemento do vetor.
- Quando termina a busca?
  - A pesquisa termina quando o elemento é encontrado ou quando é atingido o fim do vetor.

Pesquisa Seqüencial - Algoritmo

```
int busca-sequencial(int TAB[], int DADO, int tam)
{
     int ACHOU, i, POS;
    ACHOU = 0; // valor falso para a variável ACHOU
    for(i=0; i < tam; i++) {
         if(TAB[i] = = DADO)
          {
                    ACHOU = 1;
                    POS = i;
          }
    }
    if (ACHOU == 1)
          return(POS);
    else
          return(-1);  // DADO não se encontra no vetor
}
```

INEFICIENTE: O processo de busca continua mesmo depois que o elemento foi encontrado

Pesquisa Seqüencial – Algoritmo melhorado

```
int busca-sequencial(int TAB[], int DADO, int tam)
{
     int ACHOU, i, POS;
    ACHOU = 0; // valor falso para a variável ACHOU
    for(i=0; ACHOU == 0 && i < tam ; i++) {
         if(TAB[i] = = DADO)
          {
                    ACHOU = 1;
                    POS = i;
          }
    }
    if (ACHOU == 1)
          return(POS);
    else
          return(-1);  // DADO não se encontra no vetor
}
```

A variável achou é usada também como condição do laço

Pesquisa Binária

Pesquisa Binária

- No caso dos elementos estarem ordenados, a busca pode ser melhorada
- Alternativa:
  - Busca seqüencial: a busca encerra quando TAB[i] > DADO → DADO não está presente em TAB
- Pesquisa binária é mais eficiente!
  - Os elementos precisam estar ordenados!!!

30

Tabela Elemento Procurado

(44 55 12 42 94 6 18 67) (67)

Ordenar o vetor

(6 12 18 42 44 55 67 94)

elemento cen

(67) > 42

Somente 3 comparações foram necessárias para encontrar o elemento!!!

44 55 67 94)

elemento central

(67) > 55

67 94)

elemento central

(67) = 67 => encontrou

31

Tabela Elemento Procurado

(44 55 12 42 94 6 18 67) (13)

Ordenar o vetor

(6 12 18 42 44 55 67 94)

elemento central

(13) < 42

(6 12 18)

elemento central

(13) > 12

(18)

elemento central

(13) não encontrado

32

Pesquisa Binária

- A idéia básica da Pesquisa Binária consiste em diminuir cada vez mais o intervalo de busca.
- Neste método, a tabela a ser pesquisada deve estar previamente ordenada (classificada).
- Encontra-se, inicialmente, o elemento central da tabela dividindo-a, assim, em duas metades.
- Verifica-se em que metade o elemento procurado se encontra e abandona-se a outra metade

33

Pesquisa Binária

- Conforme o resultado da operação efetuada, toma-se como novo intervalo de pesquisa uma das metades do intervalo anterior e o processo de busca é repetido.
- O término do processo se dá quando o elemento desejado é localizado ou quando o intervalo de busca torna-se vazio (significando que o elemento desejado não está presente na tabela).

34

Pesquisa Binária

- A cada passo divide-se a área de pesquisa pela metade
- Caso a tabela tenha 1500 elementos, em 11 pesquisas (pior caso) consegue-se saber se o elemento está na tabela

1500/2 → 750	24/2 → 12
750/2 → 375,5	12/2 → 6
376/2 → 188	6/2 → 3
188/2 → 94	3/2 → 1,5
94/2 → 47	2/2 → 1
47/2 → 23,5	

35

Pesquisa Binária - Algoritmo

```
Inteiro Busca-binaria(inteiro TAB[], inteiro DADO, inteiro tam)
Inicio.
    inteiro ACHOU, i, ini, fim, meio;
   ACHOU = 0; // valor falso para a variável ACHOU
   ini = 0;
   fim = tam-1;
   enquanto (ini <= fim ) && (ACHOU == 0) faça
         meio = (ini + fim) / 2;
         se DADO == TAB[meio] então
                  ACHOU = 1;
         senão se DADO < TAB[meio]
                  então  fim = meio – 1;
                  senão ini  = meio + 1;
              fim se;
        fim se;
   fim-enquanto;
    se ACHOU == 1
        então
                  retorne(meio);
        senão
                  retorne(-1);  // DADO não se encontra no vetor
   fim-se;
Fim.
```

36

Pesquisa Binária - Algoritmo

```
int busca_binaria(int TAB[], int DADO, int tam)
{
   int ACHOU, i, ini, fim, meio;
   ACHOU = 0; // valor falso para a variável ACHOU
   ini = 0;
   fim = tam -1;
   while ((ini <= fim ) && (ACHOU == 0))
   {
         meio = (ini + fim)/2;
         if (DADO == TAB[meio])
                  ACHOU = 1;
         else
         {
                  if (DADO < TAB[meio])
                           fim = meio - 1;
                  else
                           ini  = meio + 1;
         }
   }
   if (ACHOU == 1)
        return(meio); // DADO se encontra no vetor !
   else
      return (-1); // DADO não se encontra no vetor
}
```

37

Fixação

- Utilize o vetor abaixo para buscar os elementos ... Informe quantas comparações foram necessárias para cada busca (busca binária):
  - TAB = {-1, 0, 4, 7, 18, 29, 75}
    - DADO = 7
    - DADO = -1
    - DADO = 75
    - DADO = 9

38

Busca Binária em C

```
int busca_binaria(int TAB[], int DADO)
{
   int ACHOU, i, ini, fim, meio;
   ACHOU = 0;
   ini = 0;
   fim = MAX-1;
   while ((ini <= fim ) && (ACHOU == 0))
   {
            meio = (ini + fim)/2;
            if (DADO == TAB[meio])
                     ACHOU = 1;
            else
            {
                     if (DADO < TAB[meio])
                              fim = meio - 1;
                     else
                              ini  = meio + 1;
            }
   }
   if (ACHOU == 1)
      return(meio);
   else
      return (-1);  // DADO não se encontra no vetor
}
```

Busca Binária em C

```
int main(int argc, char *argv[])
{
  int TAB[MAX], i, num, result;
  printf("entre com os elementos do vetor:");
  for (i=0; i< MAX; i++)
  {
     printf("TAB[%d] = ", i);
     scanf("%d", &TAB[i]);
  }
  ordena(TAB);
  printf("vetor ordenado:\n");
  for (i=0; i< MAX; i++)
       printf("TAB[%d] = %d \n", i, TAB[i]);
  printf("\nentre com o numero a ser procurado:");
  scanf("%d", &num);
  result = busca_binaria(TAB, num);
  if (result == -1)
    printf("%d nao esta no vetor \n", num);
  else
     printf("%d esta na posicao %d do vetor \n", num, result);
  system("PAUSE");
  return 0;
}
```

40