MÉTODOS DE BUSCA

Franciny Medeiros

Método de busca

- O problema da busca (ou pesquisa):

Dada uma coleção de **n** elementos, pretende-se saber se um determinado elemento **x** existe nessa coleção.

Como resolver?

Pesquisa sequencial

Pesquisa sequencial

- A solução é percorrer o vetor desde a primeira posição até a ultima, e comparar cada elemento com **x**.

```
1. i ← 0
2. Se i = n, escreve "não existe" e
     termina.
3. Se x = a[i], escreve "existe" e
     termina.
4. i ← i+1. Volta ao passo 2.
```

Pesquisa sequencial

- O algoritmo de pesquisa sequencial se chama assim, pois, varre os elementos da coleção sequencialmente um após o outro.

```c
int main()
{
   int v[6] = {25,10,2,2,17,24};
   int x = 0;
   printf("----- PESQUISA SEQUENCIAL -----\n\n");
   printf("Informe o valor da chave: ");
   scanf("%d", &x);

   int i, cont =0;
   bool existe = false;
   for(i = 0; i < 6; i++)
   {
       if(v[i] == x)
       {
            existe = true;
            cont++;
       }
   }
   if(existe)
   {
          printf("\nA chave %d existe no vetor, e apareceu %d vez(es)\n\n",x,cont);
   }else  printf("\nA chave %d não existe no vetor\n\n", x);

   printf("----- FIM PESQUISA SEQUENCIAL -----\n");
}
```

Quantas vezes é que a comparação v[i] == x é executada?

Caso x exista...

- 1 vez no melhor caso (x está na primeira posição);
- n vezes no pior caso (x está na última posição);
- n/2 vezes no caso médio.

E se o vetor estiver ordenado?

Pesquisa binária

- Se o vetor estiver ordenado, pode-se implementar um algoritmo de busca binária.

- Assume-se o "meio" da lista e escolhe se realiza a busca de **X** para a direita (elementos maiores que X) ou realiza a busca para a esquerda (elementos menos que X).

Pesquisa binária

1. esq <-- 0 , dir <-- n-1
2. Se esq > dir, escreve "não existe" e termina.
3. i <-- parte inteira de (esq+dir)/2
4. Se x = a[i], escreve "existe" e termina.
   Se x < a[i], ir <-- i-1. Volta ao passo 2.
   Se x > a[i], esq <-- i+1. Volta ao passo 2.

```
printf("----- PESQUISA BINARIA -----\n\n");

int esq = 0;
int dir = 6 - 1;

do
{
      if( esq > dir )
      {
          printf("A chave %d nao existe no vetor.\n",x);
          break;
      }
      i = (esq+dir)/2;
      if( x == v[i] )
      {
          printf("A chave %d existe no vetor e esta na posicao %d.\n",x,i);
          break;
      }
      else if( x < v[i])
      dir = i-1;
      else
          esq = i+1;
 }while( 1 );

printf("----- FIM PESQUISA BINARIA -----\n");
```

Quantas vezes a comparação x == v[i] é executada?

- no melhor caso, 1 vez.
- no pior caso, log2(n).

Métodos de pesquisa

- Se n=1000, qual será o número de comparações para:
  a) Algoritmo de busca sequencial no pior caso?
  b) Algoritmo de busca binária no pior caso?

- E se n=1000000000 (1 bilhão) ?

- Qual dos dois algoritmos é melhor?

Métodos de pesquisa

- Não se pode dizer que um é melhor do que o outro. Depende de alguns fatores...

1. Se n é pequeno (inferior a 100 ou coisa parecida) não vale a pena fazer pesquisas binárias, pois o computador é muito rápido para varrer um vetor de 100 posições.

2. O algoritmo de pesquisa binária assume que o vetor está ordenado. Ordenar um vetor também tem um custo (quantas comparações são necessárias para ordenar um vetor de dimensão utilizando o algoritmo da aula passada?

3. Se for para fazer uma só pesquisa, não vale a pena ordenar o vetor. Por outro lado, se são muitas pesquisas, o esforço da ordenação já poderá valer a pena.

Atividade

- Crie um sistema para gerenciar cadastro de produtos em uma loja.
  - Use um vetor para armazenar os produtos.
  - Cada produto tem:
    - Código: inteiro (chave de acesso)
    - Nome: string
    - Preço: real
  - Implemente uma função de busca para fazer:
    - Busca sequencial
    - Busca binária (nesta busca os dados devem estar ordenados. Escolha um algoritmo de ordenação adequado para realizar a ordenação da lista dos produtos)