Classificação e Pesquisa

Pesquisa de Dados
- Seqüencial
- Binária

Prof. Rodrigo Rocha
prof.rodrigorocha@yahoo.com
http://www.bolinhabolinha.com

Onde Estamos

- Ementa
  - Pesquisa de Dados
    - Seqüencial
    - Binária
  - Métodos de ordenação
    - seleção e troca
    - distribuição
    - inserção
    - Intercalação
  - Árvores
    - Pesquisa
    - Binária
    - AVL
    - Patrícia
  - B-Tree
  - Tabelas hash
    - Estática e Dinâmica

Introdução

- Uma operação fundamental nas tarefas computacionais é a BUSCA
  - "encontrar uma determinada informação em um grande conjunto de informações armazenadas"
- *Key (chave)*, *search key (chave procurada)*
  - Exemplo: Em um dicionário:
    - chave "*key*" – palavra
    - registro "*record*" – significado da palavra
- Operações fundamentais que devemos analisar
  - **Inicialização** da estrutura de dados
  - **Procura** por registros que contenham a chave
  - **Inserção** de novo registro
  - **Eliminação** de registro
  - **União** "*Join*" de dicionários
  - **Ordenação** do dicionário

Introdução

- Podemos ter duas situações
  - chaves únicas
    - um elemento aparece uma única vez na estrutura
  - chaves duplicadas
    - o mesmo elemento aparece várias vezes
      - ou, eliminamos a duplicação e colocamos um "apontador" para o conjunto de registros iguais
      - ou, trabalhamos com todos os registros, um de cada vez

Busca seqüêncial

- método mais simples de busca
- armazenamento simples: vetor ou lista
- procura por elemento em uma estrutura de dados, um a um, isto é: sequencialmente
- Pseudo-código
  - Para cada item da lista
  - Verifique se o elemento que você está procurando corresponde ao elemento atual
  - Se for, retorne a posição que achou
  - Senão, continue procurando até chegar no fim da lista
  - Se chegou ao final da lista e não encontrou o elemento, ele não existe, então retorne -1

Busca seqüencial em C

```
static int buscaElemento(int vet[ ], int elemento_a_procurar){

    int elemento_a_procurar = 5;
    int vet[5] = {5,2,8,7,11};

    int i;
    for(i=0;i<vet.length;i++)
    {
       if (vet[i]==elemento_a_procurar)
       {
          return i;
       }
    }
    return -1;
}
```

Analisando o desempenho

- Melhor caso
  - o item a ser procurado está na primeira posição
  - O(1)
- Pior caso
  - o item a ser procurado não está na lista, ou esta na última posição
  - varre o vetor inteiro
  - n/2 passos
  - O(n)
- Caso médio
  - procuro em metade dos elementos do vetor
  - O(n/2)

Exercícios

- a-) Crie uma classe em java chamada BuscaSeq e implemente os seguintes métodos (utilize vetor):
  - inicializaComNumerosRandomicos
  - buscaElemento
  - adicioneElemento
  - removaElemento
  - mostrarVetor
    - ➢Qual a complexidade das operações ?
- b-) E utilizando LISTA, como ficariam estas operações? A complexidade mudou ?

Exercícios

- c-) Quero implementar um novo método chamado inverteVetor, como ficaria o código em Java. Qual a complexidade ?

- d-) Qual a complexidade do método que remove o primeiro elemento de um vetor nos seguintes casos:
  - 1º caso – somente preenchendo com NULO
  - 2º caso – fazendo um deslocamento de todos os elementos restantes

43	2	4	65	32	33
2	4	65	32	33	

- e-) Qual a complexidade de uma função que desloca o vetor para a direita ou para a esquerda

43	2	4	65	32	33
2	4	65	32	33	43

- f-) Implemente os métodos dos exercícios “d” e “e”

Busca binária

- Paradigma “divisão-e-conquista”
  - dividir o número de registros em duas partes
  - determinar em que parte o item a ser procurado deve estar
  - concentrar a pesquisa nesta parte
- Restrição
  - a lista de elementos **deve estar ordenada**
- Exemplo
  - lista de telefones ordenados por número
    - abro a lista no meio,
      - se o telefone for menor, olho na primeira metade da lista
      - se maior, olho na segunda metade

Busca binária

- Procura em uma lista **A** de **n** elementos inteiros, pelo elemento **e**

  while not acabou and achou == false do

  begin

  compute meio of list

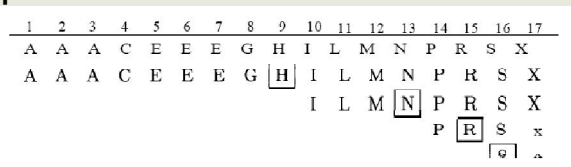
  if e == meio item then achou = true

  else if e < meio item procure metade abaixo

  else if e > meio item procure metade acima

  end

- Exemplo: Achando a letra “S” no vetor



Busca binária em C

```
int vet[5] = {7,3,4,9,1,11,43,23,77,98};
int chave=4;

   int limiteInf = 0;
   int numeroComparacoes = 0;
   int limiteSup = vet.length -1;
   int posicaoAtual;
    while(true)
    {
      numeroComparacoes++;
      posicaoAtual = (limiteInf + limiteSup ) / 2;
       if(vet[posicaoAtual]==chave)
       {
          System.out.println("Percorreu "+numeroComparacoes+" vezes");
          return posicaoAtual;
       }
       else
         if(limiteInf > limiteSup)
         {
             System.out.println("Percorreu "+numeroComparacoes+" vezes");
             return -1;
         }
         else
         {
          if(vet[posicaoAtual] < chave)
           limiteInf = posicaoAtual + 1;
          else
           limiteSup = posicaoAtual - 1;
         } // fim else
    } // fim while
```

Eficiência

- Qual a **eficiência** do Algoritmo de busca binária ?
  - Para n = 8 posições, o pior caso é quando precisando chegar na posição 1 ou 8.
  - A cada divisão, desprezamos toda a porção maior ou menor que a chave.

com n comparações	chegamos à posição
1	n/2
2	n/4
k	$n/2^k$
i	1

$n/2^i = 1 \Rightarrow n = 2^i \Rightarrow i = \mathbf{\log_2 n}$

Analisando o desempenho

- **Busca**
  - complexidade logarítmica $O(\log_2 n)$
- **Inserção** – demorada
  - devemos manter o conjunto de dados ordenado
  - elementos grandes, são inseridos na última posição.
  - leva “n” passos. O(n)
- **Eliminação**
  - os elementos devem ser movidos para completar o “buraco” vazio
  - buscar($\log_2 n$) + remover(1) + preencher espaço vazio (n)
  - = O(n)

Exercícios

- a-) Complete a tabela

Algoritmo	Complexidade (notação O)
Busca seqüencial	
Busca binária	
Inserção em vetor desordenado	
Inserção em vetor ordenado	
Remoção em vetor desordenado	
Remoção em vetor ordenado	

- b-) Você concorda com a frase “Vetores ordenados são extremamente úteis quando as buscas são mais freqüentes que as operações de remoção e inserção.”, justifique.

Exercícios

- c-) Que tipo de algortimo de busca você implementaria para cada caso abaixo, justifique:
  - procura de uma letra em um vetor que contenha o alfabeto
  - procura por um determinado número em um vetor
    - 3,7,6,4,3,2,3,5,6,78,8,9,54,3,23
  - procura por um determinado número em um vetor
    - 1,4,8,12,16,456,1024,5433,6546546
- d-) Dos algortimos abaixo, qual a sua complexidade, e qual deles você implementaria em um projeto que exigisse máxima eficiência?
  - busca sequencial
  - busca binária com ordenação O(n)
  - busca binária com ordenação $O(\log_2 n)$
  - busca binária com ordenação $O(n^2)$
- e-) Implemente o algortimo de busca binária em Java utilizando recursão. Sua complexidade mudou ?

Exercício

- f-) Um *cracker* quer invadir um conta bancária, que não possui mecanismo de bloqueio, isto é, não trava após *n* tentativas erradas de digitação da senha.
  - Se a senha tivesse apenas um algarismo númerico
    - implemente o algoritmo
    - qual a complexidade deste algoritmo ?
    - que tipo de busca é essa?
    - é possível melhorar esta busca? como ?
  - Se esta senha consiste em 4 dígitos numéricos,
    - implemente um algoritmo que ache esta senha
    - diga quala complexidade deste algoritmo
  - E se a senha fosse alfanúmerica ?
    - implemente um algoritmo que ache esta senha
    - diga qual a complexidade deste algoritmo
  - Se o computador leva 1ms para processar cada senha, quando demoraría para achar a senha ZZZZ
  - Existe alguma maneira melhor de achar esta senha ?

Bibliografia

- **Livro texto**
  - ZIVIANI, Nivio. **Projeto de Algoritmos : com implementação em Pascal e C**.. 2ª ed. São Paulo: Pioneira Thomson Learning, 2004.
  - VELOSO, Paulo A. S.. **Estrutura de Dados.** 1ª ed. São Paulo: Campus, 1983.
  - CORMEN, Thomas H. **Algoritmos : teoria e prática**. 1ª ed. Rio de Janeiro: CAMPUS, 2002.
- **Complementar**
  - SCHILDT, Herbert. **C Completo e Total**. 3ª ed. São Paulo: Pearson Education, 2005.
  - CELES, Waldemar; CERQUEIRA, Renato. **Introdução a Estruturas de Dados : com técnicas de programação em C**. 4ª ed. Rio de Janeiro: Elsevier, 2004
  - WIRTH, Niklaus. **Algoritmos e Estruturas de Dados**. 1ª ed. Rio de Janeiro: LTC, 1989
  - TENENBAUM, Aaron M; SOUZA, Tereza Cristina Félix de. **Estruturas de Dados usando C**. 1ª ed. São Paulo: Makron Books,1995.