Métodos de Busca

SCC172

Rosane Minghim
2012

Baseado no material dos Professores Rudinei Goularte e Thiago Pardo

Introdução

- Importância em estudar busca
  - Busca é uma tarefa muito comum em computação?
- Vários métodos e estruturas de dados podem ser empregados para se fazer busca
  - Quais estruturas de dados?
- Certos métodos de organização/ordenação de dados podem tornar o processo de busca mais eficiente

Introdução

- O problema da busca (ou pesquisa)

  "*Dado um conjunto de elementos, onde cada um é identificado por uma chave, o objetivo da busca é localizar, nesse conjunto, o elemento que corresponde a uma chave específica*"

Termos Relacionados

- Tabela: termo genérico, pode ser qualquer estrutura de dados usada para armazenamento interno e organização dos dados
- Uma tabela é um conjunto de elementos, chamados registros

Termos Relacionados

- Existe uma chave associada a cada registro, usada para diferenciar os registros entre si
  - Chave interna: chave está contida dentro do registro, em uma localização específica
  - Chave externa: essas chaves estão contidas em uma tabela de chaves separada que inclui ponteiros para os registros
  - Chave primária: para todo arquivo existe pelo menos um conjunto exclusivo de chaves
    - Dois registros não podem ter o mesmo valor de chave
  - Chave secundária: são as chaves não primárias
    - Chaves que não precisam ter seus valores exclusivos
    - Para que servem?

5

Termos Relacionados

- Algoritmo de busca
  - Formalmente, é o algoritmo que aceita um argumento **a** e tenta encontrar o registro cuja chave seja **a**

6

Termos Relacionados

- Operações na tabela
  - Inserção: adicionar um novo elemento à tabela
  - Algoritmo de busca e inserção: se não encontra o registro, insere um novo
  - Remoção: retirar um elemento da tabela
  - Recuperação: procurar um elemento na tabela e, se achá-lo, torná-lo disponível

7

Tipos de Busca

- A tabela pode ser:
  - Um vetor de registros
  - Uma lista encadeada
  - Uma árvore
  - Etc.
- A tabela pode ficar:
  - Totalmente na memória (busca interna)
  - Totalmente no armazenamento auxiliar (busca externa)
  - Dividida entre ambos

8

Tipos de Busca

- As técnicas de busca em memória interna que estudaremos serão:
  - Busca Seqüencial
  - Busca Binária
  - Busca por Interpolação
  - Busca em Árvores
  - Hashing
- O objetivo é encontrar um dado registro com o menor **custo**
  - Cada técnica possui vantagens e desvantagens

9

Busca Seqüencial

- A busca seqüencial é a forma mais simples de busca
  - É aplicável a uma tabela organizada como um vetor ou como uma lista

10

Busca Seqüencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

1							N=8
12	25	33	37	48	57	86	92

11

Busca Seqüencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

Procure por 48

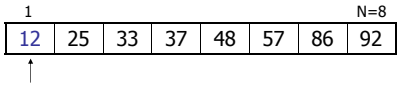
1							N=8
12	25	33	37	48	57	86	92

12

Busca Seqüencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

Procure por 48

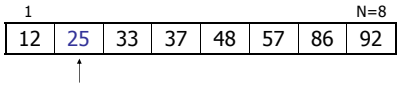


13

Busca Seqüencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

Procure por 48

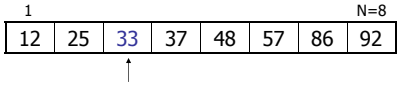


14

Busca Seqüencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

Procure por 48

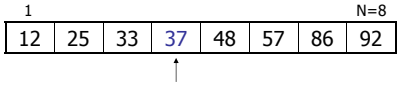


15

Busca Seqüencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

Procure por 48

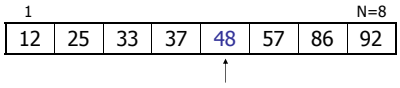


16

Busca Seqüencial

- Busca mais simples que há
  - Percorre-se registro por registro em busca da chave

Procure por 48



Busca Seqüencial

- Algoritmo de busca seqüencial em um vetor A, com N posições (0 até N-1), sendo x a chave procurada

```
i=0
Enquanto i< n e A[i] ≠ x faça
        i = i + 1
Se i == n retorne 0 /* não achou */
senão retorne 1 /* achou *
```


Busca Seqüencial

- Vetores desordenados: Uma maneira de tornar o algoritmo mais eficiente é usar um sentinela
  - Sentinela: consiste em adicionar um elemento de valor x no final da tabela
  - O sentinela garante que o elemento procurado será encontrado, o que elimina uma expressão condicional, melhorando a performance do algoritmo

```
i=0
A[n]=x
Enquanto A[i] ≠ x faça
        i = i + 1
Se i == n retorne 0 /* não achou */
senão retorne 1 /* achou *
```


Busca Seqüencial

- Complexidade
  - Se o registro for o primeiro: 1 comparação
  - Se o registro procurado for o último: N comparações
  - Se for igualmente provável que o argumento apareça em qualquer posição da tabela, em média: (n+1)/2 comparações
  - Se a busca for mal sucedida: N comparações
  - Logo, a busca seqüencial, no pior caso, é O(n)

Busca Binária

- Se os dados estiverem ordenados em um arranjo, pode-se tirar vantagens dessa ordenação
  1. Solução similar à anterior, porém parando antes (quando?)
  2. Busca binária

     A[i]<=A[i+1], se ordem crescente

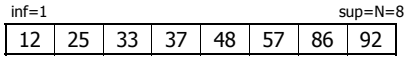
     A[i]>=A[i+1], se ordem decrescente

Busca Binária

- O elemento buscado é comparado ao elemento do meio do arranjo
  - Se igual, busca bem-sucedida
  - Se menor, busca-se na metade inferior do arranjo
  - Se maior, busca-se na metade superior do arranjo

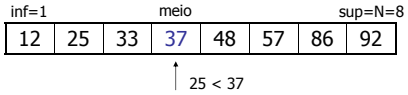
Busca Binária

- Busca-se por 25



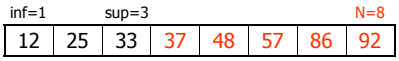
Busca Binária

- Busca-se por 25



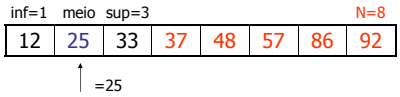
Busca Binária

- Busca-se por 25



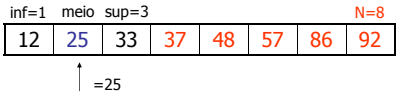
Busca Binária

- Busca-se por 25



Busca Binária

- Busca-se por 25



Em cada passo, o tamanho do arranjo em que se busca é dividido por 2

Busca Binária

- Escrever em Python uma função para a busca binária por um elemento em um arranjo ordenado

Busca Binária – Versão em C

```
int busca_binaria(int A[], int x) {

    int inf=0, sup=tam_arranjo-1, meio;

    while (inf<=sup) {
        meio=(inf+sup)/2;
        if (x==A[meio])
                return(meio);
        else if (x<A[meio])
                sup=meio-1;
        else inf=meio+1;
    }

    return(-1);
}
```


Busca Binária

- Complexidade?

Busca Binária

- Complexidade?
  - O(log(n)), pois cada comparação reduz o número de possíveis candidatos por um fator de 2

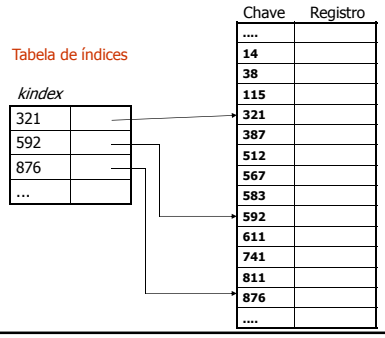
Busca Binária

- Vantagens
  - Eficiência da busca
  - Simplicidade da implementação
- Desvantagens
  - Nem todo arranjo está ordenado
  - Exige o uso de um arranjo para armazenar os dados
    - Faz uso do fato de que os índices do vetor são inteiros consecutivos
  - Inserção e remoção de elementos são ineficientes
    - Realocação de elementos

Busca Seqüencial Indexada

- Existe uma tabela auxiliar, chamada tabela de índices, além do próprio arquivo ordenado
- Cada elemento na tabela de índices contém uma chave (*kindex*) e um indicador do registro no arquivo que corresponde a *kindex*
  - Faz-se a busca a partir do ponto indicado na tabela, sendo que a busca não precisa ser feita desde o começo
- Pode ser implementada como um vetor ou como uma lista encadeada
  - O indicador da posição na tabela pode ser um ponteiro ou uma variável inteira

Busca Seqüencial Indexada



Busca Seqüencial Indexada

- Também pode indexar uma segunda chave, na qual o vetor não está ordenado.
- Ex. Clientes, visto na prova 2.

```
from collections import namedtuple
cliente = namedtuple('cliente','nome id val')
```

- Criação da lista de clientes:

```
client_list=[]
for i in range(n):
    nome_i = raw_input('nome do cliente: ')
    num_i = input('identificacao 999999999 : ')
    valores = []
    for j in range(n_meses):
        print 'valor do mes ',j+1
        valores.append(input(''))
    client = cliente(nome_i,num_i,valores)
    client_list.append(client)
```


Busca Seqüencial Indexada

- Passo 1: ordenação:
  - Pelo primeiro campo:

```
client_list.sort()
```

  - Por um outro campo (na ordem):

```
from operator    import itemgetter
client_list.sort(key=itemgetter(1), reverse=False)
```

- Tarefas:
  - Ordenar por um campo, indexar por outro.
  - Fazer a busca sequencial por um e outro
  - Fazer a busca binária pelos dois campos.

Busca Sequencial Indexada

- Vantagem
  - Os itens na tabela poderão ser examinados seqüencialmente sem que todos os registros precisem ser acessados
    - O tempo de busca diminui consideravelmente
- Desvantagens
  - Exige espaço adicional para armazenar a(s) tabela(s) de índices
- Algo mais?

37