

Algoritmos de Pesquisa

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados I (AE22CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

- Terminologia Básica
- Pesquisa Sequencial
- Pesquisa Sequencial Indexada

- Busca é uma tarefa muito comum em computação
- Recuperação de informações
- Exemplos
 - Agenda telefônica
 - Cadastro de cliente
 - Catálogo de uma biblioteca

- 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
- Várias abordagens podem ser empregadas para fazer busca
- Certos métodos de organização/ordenação de dados podem tornar o processo de busca mais eficiente
- Pesquisa em memória primária e secundária

Terminologia Básica

- Tabela ou Arquivo
 - Chave
 - Interna
 - Externa
 - Primária
 - Secundária

- Uma tabela pode ser:
 - Arranjo
 - Lista encadeada
 - Árvore
 - etc
- Uma tabela pode ser situada na:
 - Totalmente na memória primária
 - Totalmente na memória secundária
 - Dividida entre ambas

- Algumas das operações básicas em TAD:
 - Inserção
 - Remoção
 - Busca (ou pesquisa)
- Algoritmo de busca

- Tipos de busca:
 - Pesquisa sequencial
 - Pesquisa sequencial indexada
 - Pesquisa binária
 - Pesquisa por interpolação
 - Pesquisa em árvores*
 - *Hashing**
- O objetivo é encontrar um dado registro com o menor custo
- Cada técnica possui vantagens e desvantagens

Pesquisa Sequencial

Pesquisa Sequencial

- A busca sequencial é a forma mais simples de busca
- É aplicável a uma tabela organizada como um vetor ou uma lista encadeada
- Percorre-se registro por registro em busca da chave

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

- Exemplo: busca pelo item 48

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

- Exemplo: busca pelo item 48

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

Pesquisa Sequencial

- Exemplo: busca pelo item 48

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

Pesquisa Sequencial

- Exemplo: busca pelo item 48

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

- Exemplo: busca pelo item 48

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

Pesquisa Sequencial

- Exemplo: busca pelo item 48

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

- Algoritmo de busca sequencial:

```
int busca_sequencial(int x, int v[], int n){  
    int i;  
  
    for (i = 0; i < n; i++)  
        if (x == v[i])  
            return i;  
  
    return -1;  
}
```

- Uma maneira de tornar o algoritmo mais "eficiente" é usar um sentinela

```
int busca_sequencial(int x, int v[], int n){  
    int i;  
  
    for (i = 0; i < n && x != v[i]; i++);  
  
    if (i < n)  
        return i;  
    else;  
        return -1;  
}
```

- Limitação do uso de vetores: tamanho fixo
- Alternativa: listas encadeadas
- Complexidade:
 - Melhor caso: $O(1)$
 - Caso médio: $O(n)$
 - Pior caso: $O(n)$

- Para aumentar eficiência: reordenar continuamente a tabela de modo que os registros mais acessados sejam deslocados para o início (recuperação recorrente de registros)
 - ❶ Método mover-para-frente: sempre que uma pesquisa obtiver êxito, o registro recuperado é colocado no início da lista
 - ❷ Método da transposição: um registro recuperado com sucesso é trocado com o registro imediatamente anterior

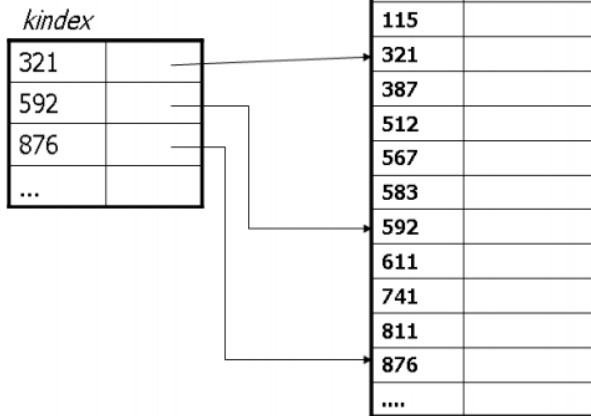
- Desvantagens do método mover-para-frente:
 - Uma única recuperação não implica que o registro será frequentemente recuperado
 - O método é mais custoso para arranjos em comparação com listas encadeadas
- Principal vantagem do método mover-para-frente: possui resultados melhores para quantidades pequena e média de buscas
- Para uma grande quantidade de buscas, o método transposição é mais vantajoso

- Busca sequencial em tabela ordenada
 - A eficiência da operação de busca melhora se as chaves dos registros estiverem ordenadas
 - Dificuldade do método?

Pesquisa Sequencial Indexada

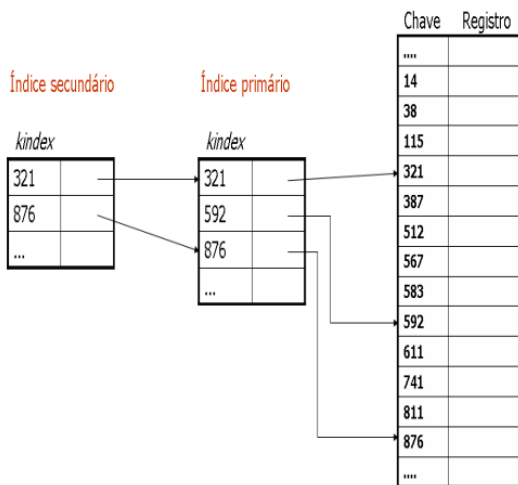
- Existe uma tabela auxiliar, chamada tabela de índices, além do próprio arquivo ordenado
- Cada elemento na tabela de índices contém uma chave (*kindex*) e um indicador do registro no arquivo que corresponde a esse índice
- A busca é feita a partir do ponto indicado na tabela
- Pode ser implementada como um vetor ou como uma lista encadeada

Tabela de índices



Pesquisa Sequencial Indexada

- Se a tabela for muito grande, pode-se ainda usar a tabela de índices secundária:



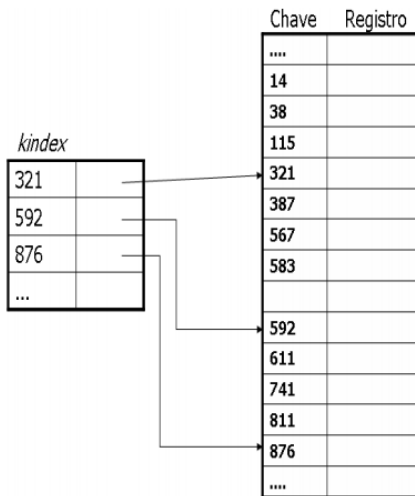
Pesquisa Sequencial Indexada

- Vantagem: os itens na tabela poderão ser examinados sequencialmente sem que todos os registros precisem ser acessados
- Desvantagens:
 - A tabela tem que estar ordenada
 - Exige espaço adicional para armazenar a(s) tabela(s) de índices

- Remoção
 - Remove-se o elemento e rearranja-se os elementos localmente
 - Pode haver atualização da tabela de índice
 - Marca-se a posição do elemento removido, indicando que ela pode ser ocupada por um outro elemento futuramente
- Inserção
 - Se houver espaço vago na tabela, rearranjam-se os elementos localmente
 - Caso contrário, rearranjar a tabela a partir do ponto apropriado e reconstruir o(s) índice(s)

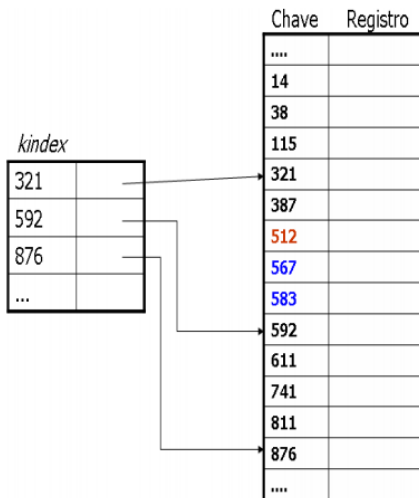
Pesquisa Sequencial Indexada

- Inserção do elemento 512 com espaço vago:



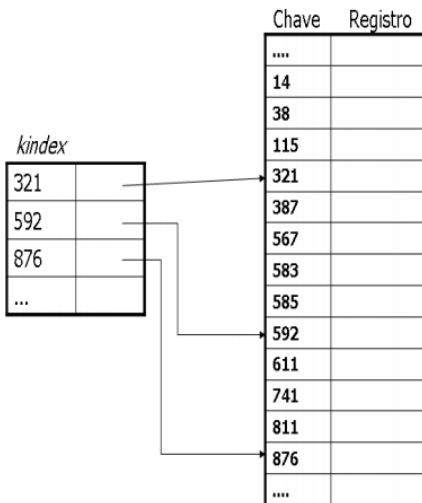
Pesquisa Sequencial Indexada

- Inserção do elemento 512 com espaço vago:



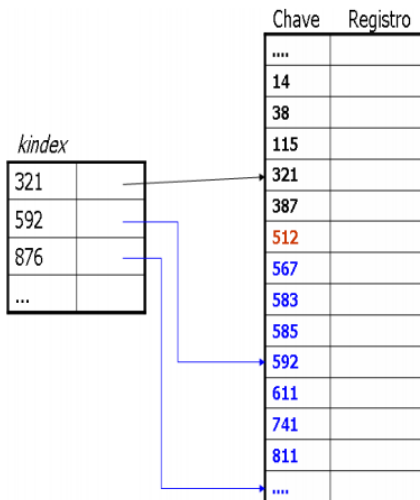
Pesquisa Sequencial Indexada

- Inserção do elemento 512 sem espaço vago:



Pesquisa Sequencial Indexada

- Inserção do elemento 512 sem espaço vago:



Pesquisa Sequencial Indexada

- Passos para montar a tabela de índice:
 - Se a tabela não estiver ordenada, ordene-a
 - Divida o número de elementos da tabela pelo tamanho do índice desejado: n/m , onde n é o tamanho da tabela e m o tamanho do índice
 - Para montar o índice, recuperam-se da tabela os elementos 0 , $0 + n/m$, $0 + 2 * (n/m)$, ...
 - Cada elemento do índice representa n/m elementos da tabela
- Para montar um índice secundário, aplica-se raciocínio similar sobre o índice primário
- Em geral, não são necessários mais do que 2 índices



Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.
Introduction to Algorithms.
Third edition, The MIT Press, 2009.



Horowitz, E., Sahni, S. Rajasekaran, S.
Computer Algorithms.
Computer Science Press, 1998.



Rosa, J. L. G.
Métodos de Busca. SCE-181 – Introdução à Ciência da
Computação II.
Slides. Ciência de Computação. ICMC/USP, 2018.



Ziviani, N.
Projeto de Algoritmos - com implementações em Java e C++.
Thomson, 2007.