Introdução aos Métodos de Busca.

Buscas Sequencial e Binária

Prof. Dr. Eleandro Maschio
Tecnologia em Sistemas para Internet
Câmpus Guarapuava
Universidade Tecnológica Federal do Paraná (UTFPR)

Buscas ou Pesquisas

- A principal finalidade de se trabalhar com **dados ordenados** é facilitar a **pesquisa** de informações.
- Contudo, há pouca eficiência em, mesmo tendo os dados ordenados, utilizarmos dos mesmos métodos de busca que nos valemos para dados desordenados.
- Portanto, é conveniente investirmos em métodos de busca que **tirem proveito** do fato de que os dados estejam ordenados.

Busca Sequencial
Em Dados Desordenados

Encontrar informações em uma matriz desordenada requer uma **busca sequencial**:

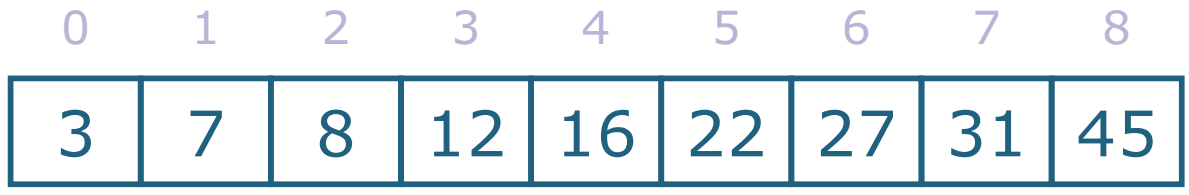
- Começa no **primeiro elemento**; e
- Interrompe quando o **elemento procurado** ou o **final da matriz** é encontrado.

Busca Sequencial
Em Dados Desordenados

- A pesquisa sequencial é fácil de ser implementada.
- O trecho de código procura por uma determinada chave nos dados e devolve o **índice** desta chave (se for encontrada) ou **-1** caso não seja.
- É fácil perceber que uma pesquisa sequencial testará em média **½ *n*** elementos.
- No melhor caso, testará somente **um elemento** e, no pior caso, ***n*** elementos.
- Se a informação estiver armazenada em disco, o tempo de procura pode ser muito longo. Mas se os dados estiverem desordenados, a pesquisa sequencial é o único método disponível.

Busca Sequencial
Em Dados Ordenados

Considere a seguinte matriz de dados ordenados e suponha que estamos procurando pelo **número 18**.



Perceba que após passarmos pelo **16** temos certeza que não encontraremos o **18**, pois os dados estão ordenados e o próximo número é o **22**.

Então, considerando dados ordenados, a busca:

- Começa no primeiro elemento; e
- Interrompe quando o elemento procurado, ou o final da matriz é encontrado, **ou um elemento maior é encontrado**.

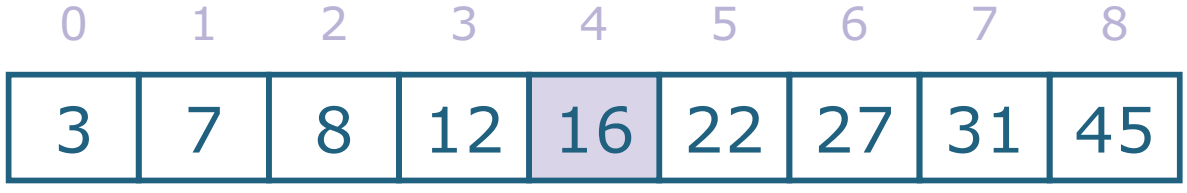
Busca Binária

- Método bastante superior à busca sequencial e utilizada somente para dados ordenados.
- Utiliza a técnica **dividir para conquistar**.
- Consiste em primeiro verificar o elemento central da matriz.
- Se esse elemento for maior do que a chave procurada, o método testa o **elemento central da primeira metade**; caso contrário, ele testa o **elemento central da segunda metade**.
- Este procedimento é repetido até que o elemento seja encontrado ou que não haja mais elementos a serem testados.

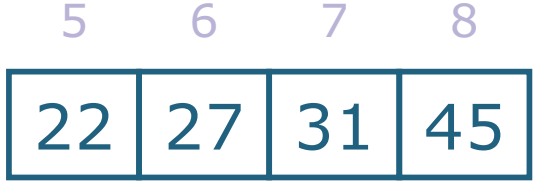
Dinâmica de Funcionamento

Suponha que estamos procurando a chave **31**.

Primeira tentativa: `(0+8) div 2 = 4`



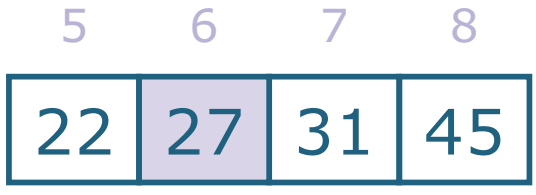
O número procurado é maior do que 16. Então a busca se concentra nos elementos da segunda metade.



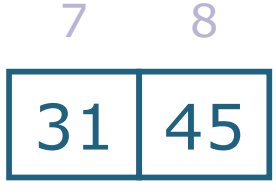
Dinâmica de Funcionamento

Suponha que estamos procurando a chave **31**.

Segunda tentativa: `(5+8) div 2 = 6`



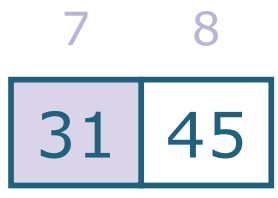
O número procurado é maior do que 27. Então a busca se concentra nos elementos da segunda metade.



Dinâmica de Funcionamento

Suponha que estamos procurando a chave **31**.

Terceira tentativa: `(7+8) div 2 = 7`



É **exatamente** o número que procuramos. Basta retornar o índice, que é 7.

Busca Binária

- No exemplo dado, fizemos 3 comparações ao invés de 8. Portanto, numa sequência típica de exemplo, com 9 elementos, para o caso utilizado, tivemos uma melhora de 62,5%.
- Em uma pesquisa binária, o número de comparações, no pior caso é **$\log_2 n$**.
- No caso médio, o número é um **pouco menor** e;
- No melhor caso, o número de comparações é **um**.

Implementação

(**1**) **Inicialize** as variáveis:

```
int inicio = 0;
int fim = TAM - 1;
```

(**2**) Para cada iteração:

```
int meio = (inicio + fim) / 2;
```

- se for o elemento do meio, **retorne** a variável `meio`;
- caso contrário, **atualize** as variáveis `inicio` ou `fim`.

(**3**) **Interrompa** se `inicio > fim`.

Exercícios

(**1**) Calcule, em uma planilha, o número de comparações para o pior caso de uma busca binária em se tratando de conjuntos de 100, 1.000, 10.000 e 100.000 elementos. Relacione com o ganho percentual com relação à busca sequencial.

(**2**) Implemente a Busca Sequencial para dados ordenados.

(**3**) Implemente a Busca Binária para conjuntos de 100 elementos.

(**4**) Com os dois algoritmos em mãos, procure uma chave utilizando ambos e apresente o número de comparações feitas nos dois casos.

Busca Binária

Na Busca Binária, considere a substituição da seguinte linha:

```
int meio = (inicio + fim) / 2;
```

pela esta outra:

```
int meio = inicio + ((fim-inicio) *
           (chave - v[inicio])) / (v[fim]-v[inicio]);
```

Qual seria o impacto no funcionamento do algoritmo?

Busca por Interpolação

- Se as chaves estiverem **uniformemente** distribuídas, esse método pode ser ainda mais eficiente do que a Busca Binária.
- A fórmula **estima** a posição baseada na proporcionalidade de onde a chave deveria ser encontrada.

Busca por Interpolação

- A complexidade, para o pior caso, pode ser reduzida para $\mathbf{\log_2(\log_2 n)}$.
- Entretanto, se as chaves não estiverem uniformemente distribuídas, a Busca por Interpolação pode ser **tão ruim** quanto a Busca Sequencial.
- Em situações práticas, as chaves tendem a se aglomerar em torno de determinados valores e não sendo uniformemente distribuídas.**cial**:
  - Há maior quantidade de nomes começando com a letra **S** do que com a letra **Q**.