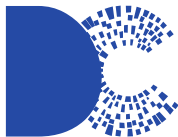




UNIVERSIDADE
FEDERAL DE
SERGIPE



DEPARTAMENTO
DE COMPUTAÇÃO

Busca binária e interpolada

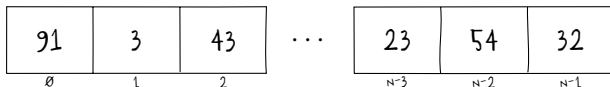
Estruturas de Dados

Bruno Prado

Departamento de Computação / UFS

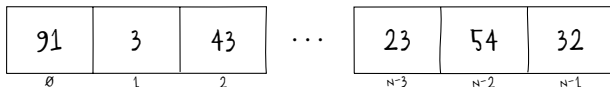
Introdução

- ▶ Problema de busca
 - ▶ Consiste em aplicar de um algoritmo de busca para encontrar um determinado elemento que pode estar armazenado em uma estrutura de dados



Introdução

- ▶ Problema de busca
 - ▶ Consiste em aplicar de um algoritmo de busca para encontrar um determinado elemento que pode estar armazenado em uma estrutura de dados



DADOS NÃO ORDENADOS $\rightarrow O(N)$

Introdução

- ▶ Dados armazenados com ordenação
 - ▶ Ata de presença
 - ▶ Lista de contatos
 - ▶ Palavras do dicionário
 - ▶ ...

Introdução

- ▶ Dados armazenados com ordenação
 - ▶ Ata de presença
 - ▶ Lista de contatos
 - ▶ Palavras do dicionário
 - ▶ ...

Como seria buscar estas
informações sem ordenação?

Introdução

- ▶ Custo da ordenação \times Eficiência de busca
 - ▶ A complexidade de algoritmos eficientes de ordenação pode variar entre $\Omega(n)$ e $O(n \log n)$, dependendo das características dos dados
 - ▶ O princípio de operação de uma busca mais eficiente é aproveitar a ordenação dos elementos para reduzir o número de comparações

Introdução

- ▶ Custo da ordenação \times Eficiência de busca
 - ▶ A complexidade de algoritmos eficientes de ordenação pode variar entre $\Omega(n)$ e $O(n \log n)$, dependendo das características dos dados
 - ▶ O princípio de operação de uma busca mais eficiente é aproveitar a ordenação dos elementos para reduzir o número de comparações
- ▶ Técnicas de busca em vetores ordenados
 - ▶ Binária
 - ▶ Interpolada

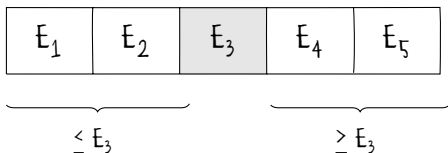
Busca binária

- ▶ Como funciona a busca binária?
 - ▶ Requisito: o vetor precisa estar ordenado
 - ▶ Propriedade: a ordenação permite dividir o vetor, reduzindo o espaço de busca a cada etapa

E_1	E_2	E_3	E_4	E_5
-------	-------	-------	-------	-------

Busca binária

- ▶ Como funciona a busca binária?
 - ▶ Requisito: o vetor precisa estar ordenado
 - ▶ Propriedade: a ordenação permite dividir o vetor, reduzindo o espaço de busca a cada etapa



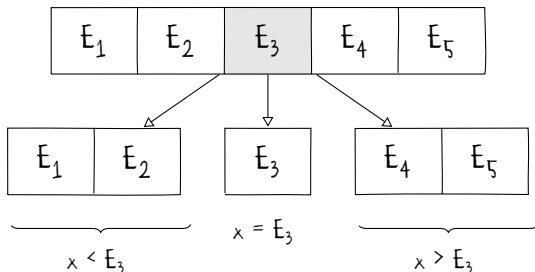
Busca binária

- ▶ Como funciona a busca binária?
 - ▶ Cada passo o vetor é dividido em duas partes
 - ▶ É verificado qual parte contém o elemento x

E_1	E_2	E_3	E_4	E_5
-------	-------	-------	-------	-------

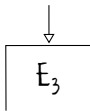
Busca binária

- ▶ Como funciona a busca binária?
 - ▶ Cada passo o vetor é dividido em duas partes
 - ▶ É verificado qual parte contém o elemento x



Busca binária

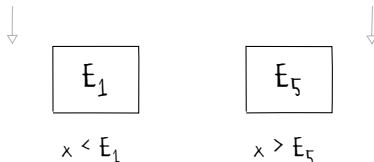
- ▶ Como funciona a busca binária?
 - ▶ Caso base 1: o elemento x está no meio do vetor
 - ▶ O índice do elemento é retornado



$$x = E_3$$

Busca binária

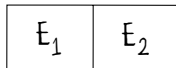
- ▶ Como funciona a busca binária?
 - ▶ Caso base 2: o elemento x não está no vetor
 - ▶ É retornado um índice negativo



Busca binária

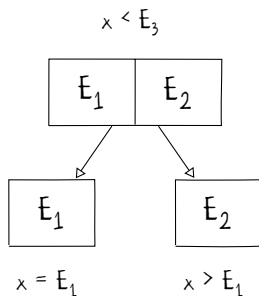
- ▶ Como funciona a busca binária?
 - ▶ Caso recursivo 1: o elemento x pode estar armazenado na metade inferior
 - ▶ O vetor é dividido novamente em duas partes para realização de uma nova busca considerando somente metade dos dados

$$x < E_3$$



Busca binária

- ▶ Como funciona a busca binária?
 - ▶ Caso recursivo 1: o elemento x pode estar armazenado na metade inferior
 - ▶ O vetor é dividido novamente em duas partes para realização de uma nova busca considerando somente metade dos dados



Busca binária

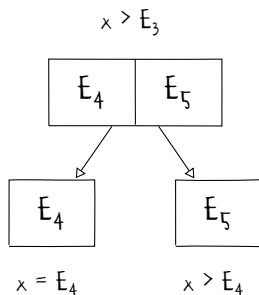
- ▶ Como funciona a busca binária?
 - ▶ Caso recursivo 2: o elemento x pode estar armazenado na metade superior
 - ▶ O vetor é dividido novamente em duas partes para realização de uma nova busca considerando somente metade dos dados

$$x > E_3$$

E_4	E_5
-------	-------

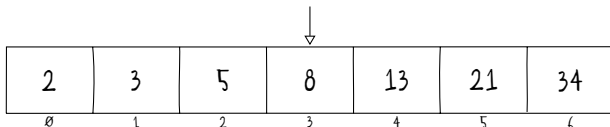
Busca binária

- ▶ Como funciona a busca binária?
 - ▶ Caso recursivo 2: o elemento x pode estar armazenado na metade superior
 - ▶ O vetor é dividido novamente em duas partes para realização de uma nova busca considerando somente metade dos dados



Busca binária

- ▶ Busca em vetor ordenado
 - ▶ Parâmetro de busca: 13
 - ▶ É feito o cálculo do índice do elemento pivô que divide o vetor em duas partes e a comparação do valor deste elemento com o valor procurado



2	3	5	8	13	21	34
0	1	2	3	4	5	6

$$p = \frac{0 + 6}{2} = 3$$

$$V[p] \leftrightarrow 13$$

Busca binária

- ▶ Busca em vetor ordenado
 - ▶ Parâmetro de busca: 13
 - ▶ É verificado que o elemento procurado pode estar armazenado na metade superior do vetor, uma vez que o elemento central é menor do que o pivô

$x > 8$
⏟

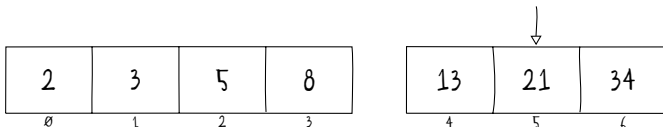
2	3	5	8	13	21	34
0	1	2	3	4	5	6

$$p = \frac{0 + 6}{2} = 3$$

$$V[p] < 13$$

Busca binária

- ▶ Busca em vetor ordenado
 - ▶ Parâmetro de busca: 13
 - ▶ É feito o cálculo do índice do elemento pivô que divide o vetor em duas partes e a comparação do valor deste elemento com o valor procurado

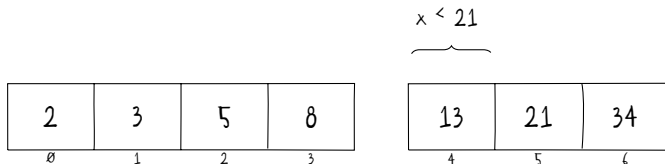


$$p = \frac{4 + 6}{2} = 5$$

$$V[p] \leftrightarrow 13$$

Busca binária

- ▶ Busca em vetor ordenado
 - ▶ Parâmetro de busca: 13
 - ▶ É verificado que o elemento procurado pode estar armazenado na metade inferior do vetor, uma vez que o elemento central é menor do que o pivô

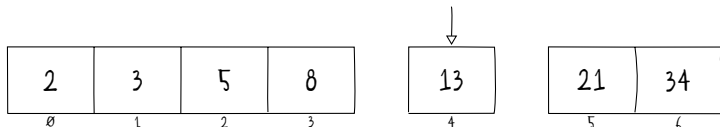


$$p = \frac{4 + 6}{2} = 5$$

$$V[p] > 13$$

Busca binária

- ▶ Busca em vetor ordenado
 - ▶ Parâmetro de busca: 13
 - ▶ É feito o cálculo do índice do elemento pivô que divide o vetor em duas partes e a comparação do valor deste elemento com o valor procurado



$$p = \frac{4 + 4}{2} = 4$$

$$V[p] \leftrightarrow 13$$

Busca binária

- ▶ Busca em vetor ordenado
 - ▶ Parâmetro de busca: 13
 - ▶ O elemento procurado é encontrado na posição 4 e o valor de seu índice p é retornado

2	3	5	8	13	21	34
0	1	2	3	4	5	6

$$p = \frac{4 + 4}{2} = 4$$

$$V[p] = 13$$

Busca binária

► Implementação recursiva em C

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Função de busca binária recursiva
4 int32_t bbr(int32_t* V, int32_t i, int32_t j, int32_t
    x) {
5     // Índice de partição
6     int32_t p = (i + j) / 2;
7     // Casos bases
8     if(j < i)
9         return -1;
10    else if(V[p] == x)
11        return p;
12    // Casos recursivos
13    else if(V[p] < x)
14        return bbr(V, p + 1, j, x);
15    else
16        return bbr(V, i, p - 1, x);
17 }
```


Busca binária

► Análise de complexidade

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

Busca binária

► Análise de complexidade

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

$$\begin{array}{llll} T(n) & = & T(\frac{n}{2}) + 1 & \\ & = & T(\frac{n}{4}) + 2 & \Rightarrow \frac{n}{2^k} = 1 \Rightarrow T(n) = T(\frac{n}{2^k}) + k \\ & = & T(\frac{n}{8}) + 3 & \quad 2^k = n \quad = 1 + k \\ & & k = \log_2 n & = 1 + \log_2 n \\ & \vdots & & \\ & = & T(\frac{n}{2^k}) + k & \end{array}$$

Busca binária

► Análise de complexidade

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\frac{n}{2}) + 1 & n > 1 \end{cases}$$

$$\begin{array}{llll} T(n) & = & T(\frac{n}{2}) + 1 & \\ & = & T(\frac{n}{4}) + 2 & \Rightarrow \frac{n}{2^k} = 1 \Rightarrow T(n) = T(\frac{n}{2^k}) + k \\ & = & T(\frac{n}{8}) + 3 & \quad 2^k = n \quad = 1 + k \\ & & k = \log_2 n & = 1 + \log_2 n \\ & \vdots & & \\ & = & T(\frac{n}{2^k}) + k & \end{array}$$

Espaço $\Theta(n)$, tempo: $\Omega(1)$ e $O(\log_2 n)$

Busca binária

► Implementação iterativa em C

```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 // Função de busca binária iterativa
4 int32_t bbi(int32_t* V, uint32_t n, int32_t x) {
5     // Índices de partições
6     int32_t i = 0, j = n - 1;
7     int32_t p = (i + j) / 2;
8     // Iterações de 1 -> k
9     while(j >= i && V[p] != x) {
10         if(V[p] > x)
11             j = p - 1;
12         else
13             i = p + 1;
14         p = (i + j) / 2;
15     }
16     return (V[p] == x) ? (p) : (-1);
17 }
```

Busca binária

- ▶ Análise de complexidade

$$\sum_{i=1}^k \frac{n}{2^i} = \frac{n}{2^1} + \frac{n}{2^2} + \dots + \frac{n}{2^k}$$

Busca binária

► Análise de complexidade

$$\sum_{i=1}^k \frac{n}{2^i} = \frac{n}{2^1} + \frac{n}{2^2} + \dots + \frac{n}{2^k}$$

$$\frac{n}{2^k} = 1$$

$$2^k = n$$

$$k = \log_2 n$$

Busca binária

► Análise de complexidade

$$\sum_{i=1}^k \frac{n}{2^i} = \frac{n}{2^1} + \frac{n}{2^2} + \dots + \frac{n}{2^k}$$

$$\frac{n}{2^k} = 1$$

$$2^k = n$$

$$k = \log_2 n$$

Espaço: $\Theta(n)$, tempo: $\Omega(1)$ e $O(\log_2 n)$

Busca interpolada

- ▶ O que é uma busca interpolada?
 - ▶ É um algoritmo de busca que calcula a posição do elemento procurado baseando-se em informações sobre os dados
 - ▶ Esta técnica é geralmente utilizada por pessoas na busca por informações com ordenação, como em agendas, dicionários ou listas de contatos

Busca interpolada

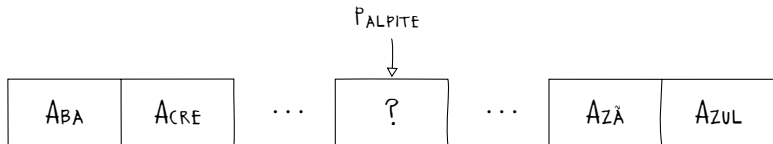
- ▶ Como buscar de forma interpolada?
 - ▶ É preciso que a estrutura de dados esteja ordenada e que a distribuição dos elementos seja conhecida

1	9	17	25	33	41	49	57
0	1	2	3	4	5	6	7

$$\begin{aligned}bi(x) &= \left\lfloor 0 + \left(\frac{7-0}{57-1} \right) \times (x-1) \right\rfloor \\&= \left\lfloor \frac{7x}{56} \right\rfloor \\&= \left\lfloor \frac{x}{8} \right\rfloor\end{aligned}$$

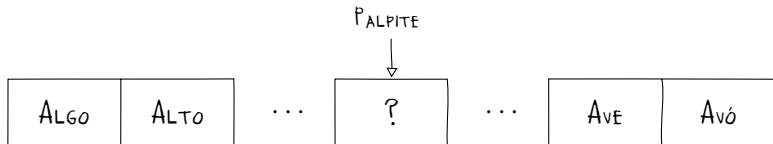
Busca interpolada

- ▶ Como buscar de forma interpolada?
 - ▶ Termo: auto
 - ▶ A distribuição das palavras no dicionário pode ser "calculada" através da espessura das páginas



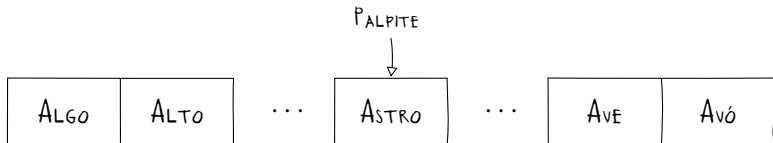
Busca interpolada

- ▶ Como buscar de forma interpolada?
 - ▶ Termo: auto
 - ▶ Baseando-se na ordenação alfabética e na quantidade de nomes, é feito o cálculo probabilístico do índice para o termo procurado



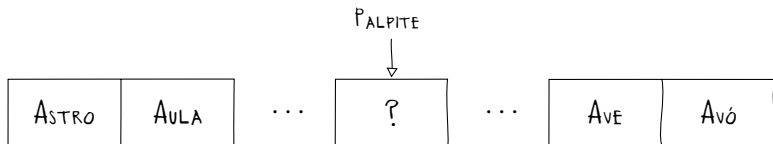
Busca interpolada

- ▶ Como buscar de forma interpolada?
 - ▶ Termo: auto
 - ▶ O índice calculado não corresponde ao termo procurado, sendo feito um novo cálculo baseado no resultado encontrado



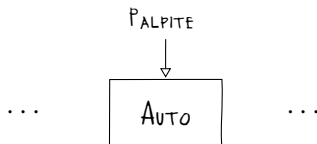
Busca interpolada

- ▶ Como buscar de forma interpolada?
 - ▶ Termo: auto
 - ▶ Por estar após o termo encontrado, é feito um novo cálculo probabilístico do índice considerando elementos da parte superior



Busca interpolada

- ▶ Como buscar de forma interpolada?
 - ▶ Termo: auto
 - ▶ O próximo índice calculado pode resultar diretamente no termo procurado ou pode ser necessária a realização de uma busca sequencial para encontrar o elemento

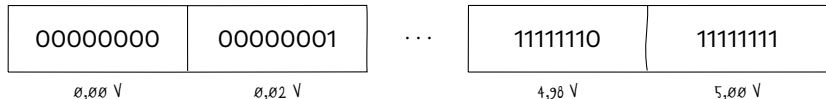


Busca interpolada

- ▶ Análise de complexidade
 - ▶ O índice de particionamento é calculado em tempo constante, de acordo com uma distribuição de probabilidade
 - ▶ Utilizando uma entrada com distribuição uniforme, é obtido um tempo de execução $\Omega(1)$
 - ▶ No pior caso é obtida a complexidade $O(n)$ quando uma distribuição uniforme não é aplicada

Exemplo

- ▶ Considere o problema de quantização de dados, onde uma informação em formato analógico precisa ser representada em um formato digital
 - ▶ O sinal analógico possui amplitude de 0 a 5V
 - ▶ A codificação digital possui 8 bits para representação
 - ▶ Aplicando os conceitos de busca binária e interpolada, realize a conversão do sinal analógico 3,78V para sua representação binária de 8 bits



Exercício

- ▶ A empresa de tecnologia Poxim Tech está realizando um estudo comparativo entre a busca binária e interpolada para um sistema de biblioteca, para determinar qual das abordagens é mais eficiente
 - ▶ Os livros são identificados unicamente pelo *International Standard Book Number* (ISBN) que é composto de 13 dígitos numéricos
 - ▶ Para realização de consulta dos livros é utilizado o ISBN, contabilizando o número total de chamadas realizadas para realização da busca binária e interpolada e retornando o nome do autor (até 50 caracteres) e do título do livro (até 100 caracteres)
 - ▶ A busca interpolada é feita da função de heurística $h(i, j) = \lfloor i + (ISBN_j - ISBN_i) \bmod (j - i + 1) \rfloor$ para determinar o provável índice do livro procurado

Exercício

- ▶ Formato do arquivo de entrada
 - ▶ [#*Livros*]
 - ▶ [#*ISBN*₁] [*Autor*₁] & [*Titulo*₁]
 - ▶ ...
 - ▶ [#*ISBN*_{*n*}] [*Autor*_{*n*}] & [*Titulo*_{*n*}]
 - ▶ [#*Consultas*]
 - ▶ [#*ISBN*₁]
 - ▶ ...
 - ▶ [#*ISBN*_{*m*}]

Exercício

► Formato do arquivo de entrada

```
1 5
2 9780130224187_Niklaus_Wirth&Algorithms_+_Data_
   Structures_=_Programs
3 9780201416077_Gaston_Gonnet&Handbook_of_Algorithms_and_
   Data_Structures
4 9780262033848_Thomas_Cormen&Introduction_to_Algorithms
5 9780321751041_Donald_Knuth&The_Art_of_Computer_
   Programming
6 9781584884354_Dinesh_Mehta&Handbook_of_Data_Structures_
   and_Applications
7 3
8 9780130224187
9 9781584884354
10 1234567890123
```

Exercício

- ▶ Formato do arquivo de saída
 - ▶ Para cada consulta realizada é exibida a quantidade de chamadas realizadas pela busca binária e interpolada, com informações sobre o livro
 - ▶ Após a realização das consultas é exibido a quantidade de vitórias e a média truncada de chamadas de cada algoritmo, onde em caso de empate a busca interpolada é vencedora

```
1 [9780130224187] B=2 | I=2 | Author:Niklaus_Wirth, Title:  
   Algorithms+_Data_Structures=_Programs  
2 [9781584884354] B=3 | I=2 | Author:Dinesh_Mehta, Title:  
   Handbook_of_Data_Structures_and_Applications  
3 [1234567890123] B=3 | I=2 | ISBN_NOT_FOUND  
4 BINARY=0:2  
5 INTERPOLATION=3:2
```