Estrutura de Dados

Aula 04

Prof. Luiz Antonio Schalata Pacheco, Dr. Eng.

Instituto Federal de Santa Catarina
Câmpus Garopaba
Curso Superior de Tecnologia em Sistemas para Internet

schalata@ifsc.edu.br

25/04/2022

Vetores Ordenados

Conceito

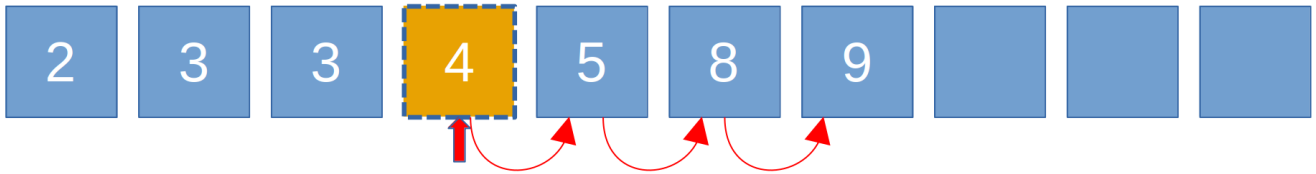
- Os dados estão organizados na ordem ascendente de valores-chave

- Cada célula tem um valor maior que a anterior
- O menor valor está no índice 0 do vetor
- Principal vantagem: agiliza o tempo de pesquisa

Vetores Ordenados - Inserção

- É necessário manter a ordem dos elementos
- Tem que pesquisar para verificar em qual posição deve ser inserido
- Os elementos posteriores devem ser remanejados
- Nos vetores não ordenados era de um passo

Inserir (4)



Vetores Ordenados - Inserção

- Pesquisar uma média de N/2 elementos
  - Pior caso: N, quando inserido na última posição
- Mover os elementos restantes leva, em média, N/2 passos
  - Pior caso: N, quando o elemento é inserido na primeira posição do vetor
- Big(O) = O(2n) = O(n)

Vetores Ordenados - Pesquisa Linear

- A pesquisa finaliza quando um item maior que o valor pesquisado é encontrado, pois o vetor está ordenado
- Pior caso: Quando está na última posição ou não está no vetor
- Big(O) = O(2n) = O(n)
- https://www.cs.usfca.edu/∼galles/visualization/Search.html

Vetores Ordenados - Exclusão

- Realiza primeiro uma pesquisa linear e depois um remanejamento
- Pesquisar uma média de N/2 elementos (pesquisa linear)
  - Pior caso: N, quando inserido na última posição
- Mover os elementos restantes leva, em média, N/2 passos
  - Pior caso: N, quando o elemento é inserido na primeira posição do vetor
- Big(O) = O(2n) = O(n)
- Mesma lógica utilizada na exclusão nos vetores não ordenados
- Pode terminar antes se não encontrar o item

Implementações

- Crie uma classe *VetorOrdenado*. Uso como modelo a classe *VetorNaoOrdenado*
- Crie um método para imprimir os valores do vetor ou que informe se ele está vazio
- Crie os métodos de inserção, pesquisa linear e remoção
- Teste o código realizando inserções, remoções e pesquisando valores

Classe Vetor Ordenado

```
1  import numpy as np
2
3  class VetorOrdenado:
4    def __init__(self, capacidade):
5      self.capacidade = capacidade
6      self.ultima_posicao = -1
7      self.valores = np.empty(self.capacidade, dtype=int)
8
9    def imprime(self):
10     if self.ultima_posicao == -1:
11       print("Vetor vazio!")
12     else:
13       for i in range(self.ultima_posicao + 1):
14         print(f'[{i}] - {self.valores[i]}')
```


Vetor Ordenado - Inserção

```
1   def insere(self, valor):
2     # Verifica se o vetor esta cheio
3     if self.ultima_posicao == self.capacidade - 1:
4       print('Capacidade maxima atingida')
5       return
6
7     # Encontra a posicao onde devera ser feita a insercao
8     posicao = 0
9     for i in range(self.ultima_posicao + 1):
10      posicao = i
11      # Compara cada posicao com o valor a ser inserido
12      if self.valores[i] > valor:
13        # Se o valor for maior, termina o laco
14        break
15      # Caso especial para a ultima posicao
16      if i == self.ultima_posicao:
17        posicao = i + 1
```

Vetor Ordenado - Inserção (continuação)

```
18      # Faz o remanejamento das posicoes
19      x = self.ultima_posicao
20      while x >= posicao:
21        self.valores[x + 1] = self.valores[x]
22        x -= 1
23
24      # Faz a insercao do novo valor na posicao correta
25      self.valores[posicao] = valor
26      self.ultima_posicao += 1
```


Vetor Ordenado - Pesquisa Linear

```
1   def pesquisa_linear(self, valor):
2     for i in range(self.ultima_posicao + 1):
3       # Se o valor for maior a pesquisa retorna -1
4       if self.valores[i] > valor:
5         return -1
6       # Situacao em que encontra o valor no vetor
7       if self.valores[i] == valor:
8         return i
9       # Caso especial para a ultima posicao
10      if i == self.ultima_posicao:
11        return -1
```


Vetor Ordenado - Exclusão

```
1   # Mesma logica da funcao excluir dos vetores nao
      ordenados.
2   # Nesse caso, a pesquisa e otimizada.
3   def excluir(self, valor):
4     posicao = self.pesquisar(valor)
5     if posicao == -1:
6       return -1
7     else:
8       # Faz o remanejamento dos valores, decrementando a
      ultima posicao
9       for i in range(posicao, self.ultima_posicao):
10        self.valores[i] = self.valores[i + 1]
11
12      self.ultima_posicao -= 1
```

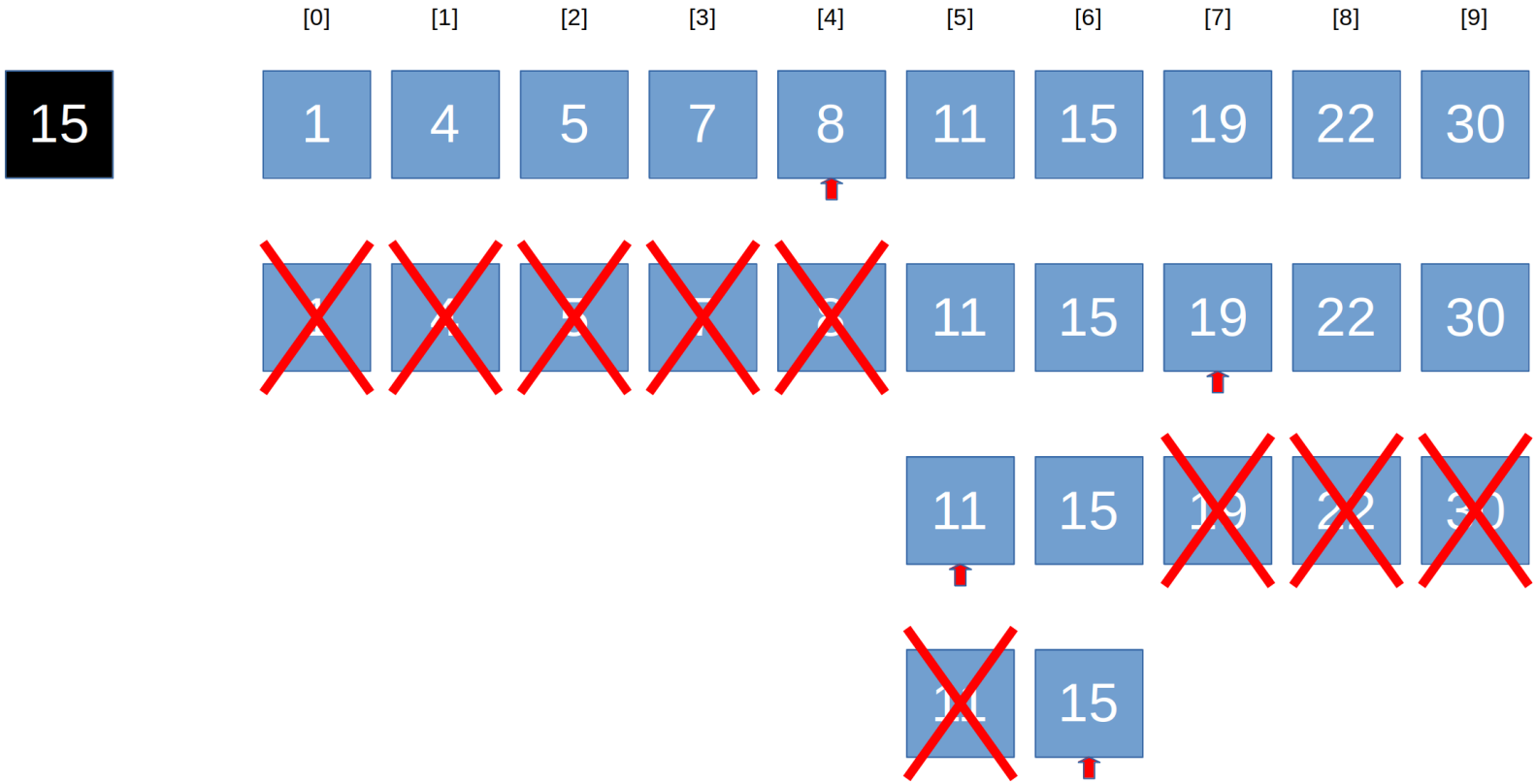

Pesquisa Binária

- É um algoritmo eficiente para encontrar um item em uma lista ordenada de itens
- É aplicável a vetores ordenados
- Consiste da divisão sucessiva pela metade da parte da lista que deve conter o item, até reduzir as localizações possíveis a apenas uma

Pesquisa Binária - Exemplo

- Pesquisar o número 37 nos valores ordenados de 1 a 100
  - 1 a $100/2 = 50$
    - 50 é o número pesquisado? Não
    - 37 é menor ou maior que 50? Menor
  - 1 a $49/2 = 25$
    - 25 é o número pesquisado? Não
    - 37 é menor ou maior que 25? Maior
  - 26 a $49/2 = 37$
    - 37 é o número pesquisado? Sim

Pesquisa Binária em Vetor Ordenado



- https://www.cs.usfca.edu/~galles/visualization/Search.html

Pesquisa Binária x Pesquisa Linear

Faixa	Comparações Binária	Comp. Linear (N/2)
10	4	5
100	7	50
1000	10	500
10000	14	5000
100000	17	50000
1000000	20	500000
10000000	24	5000000
100000000	27	50000000
1000000000	30	500000000

Pesquisa Binária - Implementação

1. Encontre a média de *max* e *min*, arredondando para baixo para que seja um inteiro
2. Se encontrou o valor, pare o algoritmo
3. Se o valor foi muito baixo, defina o *min* como 1 a mais do que o média
4. Se o valor foi muito alto, defina o *max* como 1 a menos do que o média
5. Volte ao passo 2

Pesquisa Binária - Implementação

```
1   def pesquisa_binaria(self, valor):
2     limite_inferior = 0
3     limite_superior = self.ultima_posicao
4
5     while True:
6       posicao_atual = int((limite_inferior +
      limite_superior) / 2)
7       if self.valores[posicao_atual] == valor:
8         return posicao_atual
9       elif limite_inferior > limite_superior:
10        return -1
11      else:
12        if self.valores[posicao_atual] < valor:
13          limite_inferior = posicao_atual + 1
14        else:
15          limite_superior = posicao_atual - 1
```

Vetores Ordenados - Análises

- Vetores ordenados tem tempo de pesquisa muito menores que os vetores não ordenados
- A inserção é mais demorada nos vetores ordenados
- As remoções são lentas nos dois tipos de vetores
- É necessário verificar o problema para escolher a estrutura adequada