INSTITUTO FEDERAL DE ALAGOAS – IFAL / CAMPUS MACEIÓ
Curso Bacharelado em Sistemas de Informação
Disciplina de Estruturas de Dados

Prof. MSc. Ricardo Nunes
Ricardo (arroba) ifal.edu.br

Pesquisa

Objetivos

1. Ser capaz de explicar e implementar busca sequencial e binária.

Conteúdo

1. Conceito de busca
2. Busca sequencial
3. Análise de desempenho busca sequencial
4. Busca binária
5. Análise de desempenho busca binária

Pesquisa (Busca)

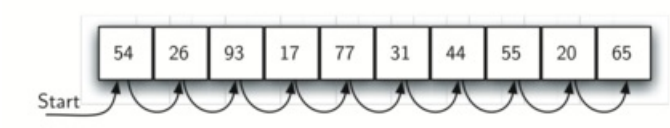
- Problema frequentemente encontrado ao se trabalhar com coleções de dados
- Geralmente retorna True ou False
- Python oferece o operador `in` para realizar pesquisa em `list`

```
>>> 15 in [3,5,2,4,1]
False
>>> 3 in [3,5,2,4,1]
True
```

- O interesse da disciplina é saber como o algoritmo funciona!

Busca Sequencial

- Os dados em uma lista são armazenados em uma posição relativa aos demais
- Nos vetores (e lista `list` em python) essas posições são os valoers dos índices de cada item
- Como os índices são ordenados, é possível percorrer toda a lista com vistando todos os itens de forma sequencial.



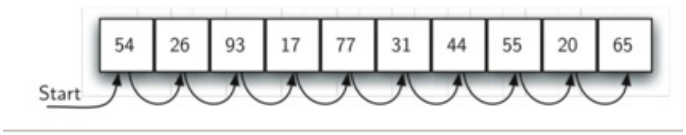
Busca Sequencial - Código em Python

```python
def busca_sequencial(uma_lista, item_pesquisado):
    pos = 0
    encontrou = False
    tamanho_lista = len(uma_lista)

    while pos < tamanho_lista and not encontrou:
        if uma_lista[pos] == item_pesquisado:
            encontrou = True
        else:
            pos = pos+1

    return encontrou

lista_teste = [1, 2, 32, 8, 17, 19, 42, 13, 0]
print('Procurando 3 na lista',lista_teste)
print('resultado:', busca_sequencial(lista_teste, 3),'\n')
print('Procurando 13 na lista',lista_teste)
print('resultado:', busca_sequencial(lista_teste, 13))
```



Análise da Busca Sequencial

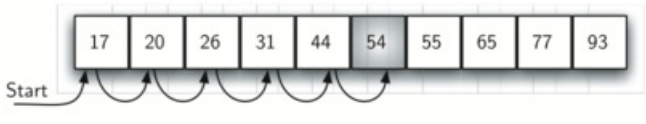
- Será utilizado a quantidade de comparações realizadas para fazer a busca
- Suposições
- lista **não ordenada**
- a probabilidade que um item esteja em uma posição da lista é a mesma para todas as posições
- a lista tem **n** itens

Caso	Melhor caso	Pior Caso	Caso Médio
item está presente na lista	1	n	n/2
item **não** está presente na lista	n	n	n

- Essa técnica é O(n)

Busca Sequencial em uma lista Ordenada

- Como seria a pesquisa se os itens estivessem ordenados. Poderia ter algum ganho em relação a isso?
- Procurar `50` na seguinte lista:



Busca Sequencial em uma lista Ordenada - Código em Python

```
def busca_sequencial_ordenada(uma_lista, item_pesquisado):
    pos = 0
    encontrou = False
    parar = False

    tamanho_lista = len(uma_lista)

    while pos < tamanho_lista and not encontrou and not parar:
        if uma_lista[pos] == item_pesquisado:
            encontrou = True
        else:
            if uma_lista[pos] > item_pesquisado:
                parar = True
            else:
              pos = pos+1

    return encontrou

lista_teste = [0, 1, 2, 8, 13, 17, 19, 32, 42]
print('Procurando 3 na lista',lista_teste)
print('resultado:', busca_sequencial_ordenada(lista_teste, 3),'\n')
print('Procurando 13 na lista',lista_teste)
print('resultado:', busca_sequencial_ordenada(lista_teste, 13))
```

Análise da Busca Sequencial em lista ordenada

Caso	Melhor caso	Pior Caso	Caso Médio
item está presente na lista	1	n	n/2
item **não** está presente na lista	1	n	n/2

- Essa técnica ainda é O(n)

Questões

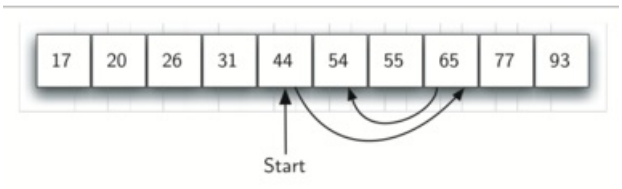
Q-1. Suponha que você está realizando uma busca sequencia na lista [15, 18, 2, 19, 18, 0, 8, 14, 19, 14]. Quantas comparações serão necessárias para encontrar 18?

Q-2. Suponha que você está realizando uma busca sequencia na lista ordenada [3, 5, 6, 8, 11, 12, 14, 15, 17, 18]. Quantas comparações serão necessárias para encontrar 13?

Busca Binária

- Se uma lista estiver ordenada, é possível tirar proveito disso!

- Passos:
- Verifica o meio da lista. Se achou elemento, retorna
- Caso contrário, aproveite-se do fato que a lista está ordenada e elimine metade dos elementos restantes.
- Repita os passos anteriores para os elementos não eliminados



Busca Binária - Código em Python

```
def busca_binaria(uma_lista, item_pesquisado):
    inicio = 0
    fim   = len(uma_lista)-1
    encontrou = False

    while inicio<=fim and not encontrou:
        meio   = (inicio + fim)//2
        if uma_lista[meio] == item_pesquisado:
            encontrou = True
        else:
            if item_pesquisado < uma_lista[meio]:
                fim = meio-1
            else:
                inicio = meio+1
    return encontrou

lista_teste = [0, 1, 2, 8, 13, 17, 19, 32, 42,]
print('Procurando 3 na lista',lista_teste)
print('resultado',busca_binaria(lista_teste, 3))
print('Procurando 13 na lista',lista_teste)
print('resultado',busca_binaria(lista_teste, 13))
```

- Exemplo da estratégia **dividir e conquistar**
- Dividir o problema em partes menores
- Resolvendo as partes uma dia chegará na solução do problema todo

Busca Binária - Código Recursivo em Python

```
def busca_binaria(uma_lista, item_procurado):
    if len(uma_lista) ==  0:
        return False
    else:
        meio = len(uma_lista)//2
        if uma_lista[meio]==item_procurado:
            return True
        else:
            if item_procurado<uma_lista[meio]:
                return busca_binaria(uma_lista[:meio],item_procurado)
            else:
                return busca_binaria(uma_lista[meio+1:],item_procurado)

lista_teste = [0, 1, 2, 8, 13, 17, 19, 32, 42,]
print('Procurando 3 na lista',lista_teste)
print('resultado',busca_binaria(lista_teste, 3))
print('Procurando 13 na lista',lista_teste)
print('resultado',busca_binaria(lista_teste, 13))
```

Análise da Busca Binária

- A cada iteração (comparação), metade dos elementos é eliminada (n/2)

Comparações	Números aproximado de itens deixados na lista
1	n/2
2	n/4
3	n/8
…	…
i	n/2^i

- Quando são necessárias i comparações para se chegar a 1: n/(2^i) == 1
- 2^i = n
- logo, i = log(n)

- A busca binária é portanto O (log n)

Análise da Busca Binária (2)

- A análise anterior supõe que o operador slice `return busca_binaria(uma_lista[:meio],item_procurado)` é realizado em tempo constante O(1)
- Mas na verdade ele é realizado em O(k)
- Isso precisa ser melhorado!!!

Questões

Q-3. Suponha que você tem a seguinte lista ordenada [3, 5, 6, 8, 11, 12, 14, 15, 17, 18] e está usando o algoritmo de busca binária recursivo. Qual o grupo de números corretamente mostra a sequência de comparações usadas para encontrar 8?

1. 11, 5, 6, 8
2. 12, 6, 11, 8,
3. 3, 5, 6, 8
4. 18, 12, 6, 8

Q-4. Suponha que você tem a seguinte lista ordenada [3, 5, 6, 8, 11, 12, 14, 15, 17, 18] e está usando o algoritmo de busca binária recursivo. Qual o grupo de números corretamente mostra a sequência de comparações usadas para encontrar 16?

1. 11, 14, 17
2. 18, 17, 15
3. 14, 17, 15
4. 12, 17, 15

Próximo assunto

- Busca sequencial: O(n)

- Busca binária: O(log n)

- É possível busca em O(1)

◘ - Como?

> Hashing

Para estudar

- Pesquisa
- Seções 5.1 a 5.4 do livro [5] https://interactivepython.org/runestone/static/pythonds/SortSearch/toctree.html
- Capítulo sobre Pesquisa em qualquer livro sobre estruturas de dados

Referências

1. Tradução do livro *How to Think Like a Computer Scientist: Interactive Version*, de Brad Miller e David Ranum. link: https://panda.ime.usp.br/pensepy/static/pensepy/index.html
2. Allen Downey, Jeff Elkner and Chris Meyers. *Aprenda Computação com Python 3.0*. link: https://chevitarese.files.wordpress.com/2009/09/aprendacomputaocompython3k.pdf
3. SANTOS, A. C. *Algoritmo e Estrutura de Dados I*. 2014. Disponível em http://educapes.capes.gov.br/handle/capes/176522
4. SANTOS, A. C. *Algoritmo e Estrutura de Dados II*. 2014. Disponível em 2014. Disponível em https://educapes.capes.gov.br/handle/capes/176557
5. Tradução do livro [5] *Problem Solving with Algorithms and Data Structures using Python* de Brad Miller and David Ranum. link: https://panda.ime.usp.br/pythonds/static/pythonds_pt/index.html
6. Brad Miller and David Ranum. *Problem Solving with Algorithms and Data Structures using Python* link: https://panda.ime.usp.br/pythonds/static/pythonds_pt/index.html
7. Caelum. Algoritmos e Estruturas Dados em Java. Disponível em https://www.caelum.com.br/download/caelum-algoritmos-estruturas-dados-java-cs14.pdf

That’s all Folks