Algoritmos e Programação de Computadores

Busca Sequencial e Binária

Profa. Sandra Avila
Instituto de Computação (IC/Unicamp)

MC102, 23 Maio, 2018

Agenda

- O Problema da Busca
- Busca Sequencial
- Busca Binária
- Exercícios

Busca


Google
Pesquisa Google
Estou com sorte
Disponibilizado pelo Google em: English

Busca

Temos uma coleção de elementos, onde cada elemento possui um identificador/chave único, e recebemos uma chave para busca. Devemos ***encontrar o elemento da coleção que possui a mesma chave*** *ou identificar que não existe nenhum elemento com a chave dada.*

Busca

- O problema da busca é um dos mais básicos em Computação e também possui diversas aplicações.
  - Suponha que temos um cadastro com registros de motoristas.
  - Uma lista de registros é usado para armazenar as informações dos motoristas. Podemos usar como chave o número da carteira de motorista, ou o RG, ou o CPF.
- Veremos algoritmos simples para realizar a busca assumindo que dados estão em uma lista.

Busca

- Nos nossos exemplos vamos criar a função:
  - **`busca(lista,chave)`**, que recebe uma lista e uma chave para busca.
  - A função deve retornar o índice da lista que contém a chave ou -1 caso a chave não esteja na lista.

Busca

- Python já contém um método em listas que faz a busca `index()`:

```
lista = [20, 5, 15, 24, 67, 45, 1, 76]
lista.index(24)
```

```
3
```

mas esse método não funciona da forma que queremos para chaves que **não** estão na lista.

```
lista.index(100)
```

```
Traceback (most recent call last):
    File "<std in>", line 1, in <module>
ValueError : 100 is not in list
```

Busca Sequencial

Busca Sequencial

- A busca sequencial é o algoritmo mais simples de busca:
  - Percorra toda a lista comparando a chave com o valor de cada posição.
  - Se for igual para alguma posição, então devolva esta posição.
  - Se a lista toda foi percorrida então devolva -1.

Busca Sequencial

```
def buscaSequencial(lista, chave):
    for i in range(len(lista)):
        if lista[i] == chave:
            return i
    return -1
```

```
lista = [20, 5, 15, 24, 67, 45, 1, 76]
buscaSequencial(lista, 24)
buscaSequencial(lista, 100)
```

```
3
-1
```

Busca Binária

Busca Binária

- A busca binária é um algoritmo um pouco mais sosticado.
- É mais eficiente, mas requer que a **lista esteja ordenada** pelos valores da chave de busca.

Busca Binária

- A ideia do algoritmo é a seguinte (assuma que a lista está ordenada):
  - Verifique se a chave de busca é igual ao valor da posição do meio da lista.
  - Caso seja igual, devolva esta posição.
  - Caso o valor desta posição seja maior, então repita o processo mas considere que a lista tem metade do tamanho, indo até posição anterior a do meio.
  - Caso o valor desta posição seja menor, então repita o processo mas considere que a lista tem metade do tamanho e inicia na posição seguinte a do meio.

Busca Binária

Pseudo código

```
#vetor começa em inicio e termina em fim
inicio = 0
fim = tam-1
repita enquanto tamanho da lista considerado for >= 1
    meio = (inicio + fim)/2
    se lista[meio] == chave então
        devolva meio
    se lista[meio] > chave então
        fim = meio - 1
    se lista[meio] < chave então
        inicio = meio + 1
```

Busca Binária

chave = 15

lista

1	5	15	20	24	45	67	76	78	100
0	1	2	3	4	5	6	7	8	9

inicio =

fim =

meio =

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

inicio = 0

fim = 9

meio = 4

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

```
se lista[meio] > chave então
    fim = meio - 1
```

inicio = 0

fim = 9

meio = 4

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

```
se lista[meio] > chave então
    fim = meio - 1
```

inicio = 0

fim = 3

meio = 4

Como o valor da posição do meio é maior que a chave, atualizamos o **fim** da lista considerada.

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

inicio = 0

fim = 3

meio = 1

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

```
se lista[meio] < chave então
    inicio = meio + 1
```

inicio = 0

fim = 3

meio = 1

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

```
se lista[meio] < chave então
    inicio = meio + 1
```

inicio = 0

fim = 3

meio = 1

Como o valor da posição do meio é menor que a chave, atualizamos **inicio** da lista considerada.

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

```
se lista[meio] < chave então
    inicio = meio + 1
```

inicio = 2

fim = 3

meio = 1

Como o valor da posição do meio é menor que a chave, atualizamos **inicio** da lista considerada.

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

inicio = 2

fim = 3

meio = 2

Busca Binária

chave = 15

lista	1	5	15	20	24	45	67	76	78	100
	0	1	2	3	4	5	6	7	8	9

```
se lista[meio] == chave então
    devolva meio
```

inicio = 2

fim = 3

meio = 2

Finalmente encontramos a chave e podemos devolver sua posição 2.

Busca Binária

```
def buscaBinaria(lista, chave):
    inicio = 0
    fim = len(lista)-1
    while inicio <= fim:
        meio = (inicio + fim)//2
        if lista[meio] == chave:
            return meio
        elif lista[meio] > chave:
            fim = meio - 1
        else:
            inicio = meio + 1
    return -1
```

Busca Binária

```
lista = [20, 5, 15, 24, 67, 45, 1, 76]
insertionSort(lista)
lista
```

```
[1, 5, 15, 20, 24, 45, 67, 76]
```

```
buscaBinaria(lista, 24)
buscaBinaria(lista, 25)
```

```
4
-1
```

Exercício

Refaça as funções de busca sequencial e busca binária assumindo que a lista possui chaves que podem aparecer repetidas.

Neste caso, você deve retornar uma lista com todas as posições onde a chave foi encontrada.

Se a chave só aparece uma vez, a lista contera apenas um índice. E se a chave não aparece, as funções devem retornar a lista vazia.

Referências

- Os slides dessa aula foram baseados no material de MC102 do Prof. Eduardo Xavier (IC/Unicamp).

Exercícios:
Matrizes

Exercícios

- Escreva um programa que leia todas as posições de uma matriz 10x10. O programa deve então exibir o número de posições não nulas na matriz.

```
# Adiciona cada número na matriz 10x10
matriz = []
for i in range(10):
     lista = []
     for j in range(10):
          numero = int(input())
          lista.append(numero)
     matriz.append(lista)

# Conta quantos número são não nulos
nao_nulo = 0
for i in matriz:
     for j in i:
          if j != 0:
               nao_nulo += 1

print("O número de posições não nulas é", nao_nulo)
```

```
# Adiciona cada número na matriz 10x10
matriz = []
nao_nulo = 0
for i in range(10):
     lista = []
     for j in range(10):
          numero = int(input())
          lista.append(numero)
          if numero != 0: # Conta quantos número são não nulos
               nao_nulo += 1
     matriz.append(lista)

print("O número de posições não nulas é", nao_nulo)
```

```
# Lê uma matriz 10x10 e conta quantos número são não nulos
nao_nulo = 0
for i in range(10):
     for j in range(10):
          if matriz[i][j] != 0:
               nao_nulo += 1

print("O número de posições não nulas é", nao_nulo)
```

```
# Lê uma matriz e conta quantos número são não nulos
nao_nulo = 0
for linha in matriz:
     for numero in linha:
          if numero != 0:
               nao_nulo += 1

print("O número de posições não nulas é", nao_nulo)
```

Lab07:
Critérios para Aprovação em MC102

Critérios para Aprovação em MC102

O critério de aprovação em MC102 é composto por vários itens com pesos diferentes. Nesta tarefa, vamos fazer um programa em Python para podermos calcular a média e situação final de um(a) aluno(a) com facilidade.

Elementos componentes da avaliação

Atividades conceituais: são os questionários que podem ser respondidos via Moodle. Neste semestre não teremos atividades presenciais e, portanto, a média das atividades conceituais M_{AC} será dada pela média aritmética simples das n notas das atividades conceituais.

Tarefas de laboratório: são os programas desenvolvidos e entregues para correção automática no SuSy. Para compor a média dos laboratórios M_L, cada tarefa tem um peso indicado no enunciado do laboratório.

Provas teóricas: são as avaliações aplicadas em sala de aula, sem consulta. Para compor a média de provas M_P a primeira prova tem peso 2 e a segunda, peso 3.

Média ponderada dos elementos: será dada pela fórmula:

$$M_{Elem} = 0.6 * M_P + 0.3 * M_L + 0.1 * M_{AC}$$

Resultado final

Seja `freq` a porcentagem de frequências às aulas e M_{Pre} definida como:

$$M_{Pre} = \min(M_{Elem}, M_P, M_L)$$

O resultado final será dado pelas seguintes regras:

- Caso `freq` ≥ 75%:
  - Caso $M_{Pre} \geq 5$: o(a) aluno(a) estará **aprovado(a) por nota e frequência** com $M_{Final} = M_{Elem}$.
  - Caso $2.5 \leq M_{Pre} < 5$: o(a) aluno(a) terá direito a fazer o exame. Sua média final será $M_{Final} = (M_{Pre}+Exame)/2$.

```python
def tupla_float_int(x) :
    x = x[1:-1]          # remove parênteses
    x = x.split(",")     # separa em duas strings
    f = float(x[0])      # converte primeiro elemento para float
    i = int(x[1])        # converte segundo elemento para int
    return (f,i)         # retorna tupla

# Leitura de dados
ac = [float(x) for x in input().split()]
labs = [tupla_float_int(x)  for x in input().split()]
provas = [float(x) for x in input().split()]
freq = int(input())
```

```
# Media atividades conceituais
# Media laboratorios
# Media das provas
# Media dos elementos
# Media preliminar
```

```
# Media atividades conceituais
media_ac = 0
for i in range(len(ac)):
    media_ac += ac[i]
media_ac = media_ac/len(ac)
print("Média das atividades conceituais: %.1f"  % media_ac)

# Media laboratorios
media_labs = 0
peso_labs = 0
for i in range(len(labs)):
    media_labs += labs[i][0]*labs[i][1]
    peso_labs += labs[i][1]
media_labs = media_labs/peso_labs
print("Média das tarefas de laboratório: %.1f"  % media_labs)
```

```
# Media das provas
media_provas = (2*provas[0] + 3*provas[1]) / 5
print("Média das provas: %.1f" % media_provas)

# Media dos elementos
media_elementos = 0.6*media_provas + 0.3*media_labs + 0.1*media_ac
print("Média ponderada dos elementos: %.1f" % media_elementos)

# Media preliminar
media_pre = min(media_elementos, media_provas, media_labs)
print("Média preliminar: %.1f" % media_pre)
```

```
# Resultado final
if freq < 75:
     print("Reprovado(a) por frequência.")
     print("Média final: %.1f" % media_pre)
elif media_pre < 2.5:
     print("Reprovado(a) por nota.")
     print("Média final: %.1f" % media_pre)
elif media_pre >= 5:
     print("Aprovado(a) por nota e frequência.")
     print("Média final: %.1f" % media_elementos)
else:
     exame = float(input())
     print("Nota no exame: %.1f" % exame)
     media_final = (media_pre + exame)/2
     if media_final >= 5:
          print("Aprovado(a) por nota e frequência.")
          print("Média final: %.1f" % media_final)
     else:
          print("Reprovado(a) por nota.")
          print("Média final: %.1f" % media_final)
```

```python
def tupla_float_int(x) :
    x = x[1:-1]         # remove parênteses
    x = x.split(",")    # separa em duas strings
    f = float(x[0])     # converte primeiro elemento para float
    i = int(x[1])       # converte segundo elemento para int
    return (f,i)        # retorna tupla

# Leitura de dados
ac = [float(x) for x in input().split()]
labs = [tupla_float_int(x)  for x in input().split()]
provas = [float(x) for x in input().split()]
freq = int(input())

# Media atividades conceituais
media_ac = 0
for i in range(len(ac)):
        media_ac += ac[i]
media_ac = media_ac/len(ac)
print("Média das atividades conceituais: %.1f"  % media_ac)

# Media laboratorios
media_labs = 0
peso_labs = 0
for i in range(len(labs)):
        media_labs += labs[i][0]*labs[i][1]
        peso_labs += labs[i][1]
media_labs = media_labs/peso_labs
print("Média das tarefas de laboratório: %.1f"  % media_labs)

# Media das provas
media_provas = (2*provas[0] + 3*provas[1]) / 5
print("Média das provas: %.1f"  % media_provas)
```

```python
# Media dos elementos
media_elementos = 0.6*media_provas + 0.3*media_labs +
0.1*media_ac
print("Média ponderada dos elementos: %.1f"  %
media_elementos)

# Media preliminar
media_pre = min(media_elementos, media_provas, media_labs)
print("Média preliminar: %.1f"  % media_pre)

# Resultado final
if freq < 75:
        print("Reprovado(a) por frequência." )
        print("Média final: %.1f"  % media_pre)
elif media_pre < 2.5:
        print("Reprovado(a) por nota." )
        print("Média final: %.1f"  % media_pre)
elif media_pre >= 5:
        print("Aprovado(a) por nota e frequência." )
        print("Média final: %.1f"  % media_elementos)
else:
        exame = float(input())
        print("Nota no exame: %.1f"  % exame)
        media_final = (media_pre + exame)/2
        if media_final >= 5:
                print("Aprovado(a) por nota e frequência." )
                print("Média final: %.1f"  % media_final)
        else:
                print("Reprovado(a) por nota." )
                print("Média final: %.1f"  % media_final)
```

Lab06:
Detetives e Assassinos

Detetives e Assassinos

Nesta tarefa, vamos ler uma sequência de crimes no seguinte formato:

`<assassino> <vítima> <detetive>`

Estas informações não estão ordenadas nem por tempo nem por ordem alfabética. Sua tarefa será ao final da leitura gerar um relatório ordenado informando para cada pessoa:

- Se ela ainda está viva ou se foi assassinada. Caso tenha sido assassinada o comentário ***(in memoriam)*** deve ser colocado após o nome.
- Se a pessoa não atuou como detetive e não matou ninguém uma indicação de que foi uma **vítima inocente.**
- Se a pessoa matou uma ou mais pessoas e não atuou como detetive, uma indicação de **assassino(a)**. Logo após uma lista de quantos detetives, assassinos(as) ou vítimas inocentes ele matou.
- Se a pessoa atuou como detetive, uma indicação de **detetive** e o número de casos que ela resolveu. Um detetive também pode ter matado pessoas. Os crimes podem ter ocorrido em serviço ou o detetive pode ter matado de maneira premeditada. Não é necessário entrar neste mérito, mas apenas apresentar uma lista quantos detetives, assassinos ou vítimas inocentes foram mortas por este detetive.

Entrada

A entrada consiste de um inteiro N (1 ≤ N ≤ 100) seguido de N linhas no formato `<assassino> <vítima> <detetive>` .

Caso seja fornecido um número menor do que 1 ou maior do que 100, a mensagem de erro `"Valor inválido na entrada."` deve ser emitida.

Saída

A saída é um relatório por pessoa, em ordem alfabética, separado por linhas contendo 60 hífens.

- Caso a pessoa seja uma vítima, reportar:

```
------------------------------------------------------------
nome_vitima (in memoriam): vítima inocente.
------------------------------------------------------------
```

- Caso a pessoa seja um assassino, reportar:

```
linhas = int(input())
if linhas < 1 or linhas > 100:
     print("Valor inválido na entrada.")
     exit()
pessoas = {} # Dicionário para pessoas
detetives = {} # Dicionário para detetives
assassinos = {} # Dicionário para assassinos
```

```
# Entrada dos dados
for i in range(linhas):
    assassino, vitima, detetive = input().split()

    pessoas[vitima] = False
    # Adiciona os nomes nos respectivos dicionários,
    # adequando os valores nas chaves se elas já existissem.
    if not detetive in pessoas:
        pessoas[detetive] = True
    if not assassino in pessoas:
        pessoas[assassino] = True

    if (detetive in detetives):
        detetives[detetive] += 1
    else:
        detetives[detetive] = 1

    if (assassino in assassinos):
        assassinos[assassino].append(vitima)
    else:
        assassinos[assassino] =[vitima]
```

```
# Monta a saída para todas as pessoas.
for person in sorted(pessoas):
    # Primeiro, a verificação para detetive, já que uma mesma pessoa
    # pode estar nos dicionários detetives e assassinos.
    if person in detetives:
        # ...
        if person in assassinos:
        # ...

    # Depois, se essa pessoa não estiver em detetives, mas estiver em
    assassinos:
    elif person in assassinos:
        # ...

    # Por último, vítima inocente
    else:
        # ...
```

```
# Monta a saída para todas as pessoas.
for person in sorted(pessoas):
    # Primeiro, a verificação para detetive, já que uma mesma pessoa
    # pode estar nos dicionários detetives e assassinos.
    if person in detetives:
        print("-"*60)
        if pessoas[person]==False:
            isdead = ' (in memoriam)'
        else:
            isdead = ''
        print(person+isdead+": detetive.")
        print("  Resolveu %d caso(s)." % detetives[person])

        # ...
```

```
# ...
if person in assassinos:
     kill_vitima = 0
     kill_assassinos = 0
     kill_detetives = 0
     for vitima in assassinos[person]:
          if vitima in detetives:
               kill_detetives += 1
          elif vitima in assassinos:
               kill_assassinos += 1
          else:
               kill_vitima += 1
     if kill_detetives > 0:
          print("  Matou %d detetive(s)." % kill_detetives)
     if kill_assassinos > 0:
          print("  Matou %d assassinos(s)." % kill_assassinos)
     if kill_vitima > 0:
          print("  Matou %d inocente(s)." % kill_vitima)
```

```
    # Depois, se essa pessoa não estiver em detetives, mas estiver em assassinos:
    elif person in assassinos:
        print("-"*60)
        if pessoas[person]==False:
            isdead = ' (in memoriam)'
        else:
            isdead = ''
        print(person+isdead+": assassino(a).")
        kill_vitima = 0
        kill_assassinos = 0
        kill_detetives = 0
        for vitima in assassinos[person]:
            if vitima in detetives:
                kill_detetives += 1
            elif vitima in assassinos:
                kill_assassinos += 1
            else:
                kill_vitima += 1
        if kill_detetives > 0:
            print("  Matou %d detetive(s)." % kill_detetives)
        if kill_assassinos > 0:
            print("  Matou %d assassinos(s)." % kill_assassinos)
        if kill_vitima > 0:
            print("  Matou %d inocente(s)." % kill_vitima)
```

```
    # Por último, vítima inocente
    else:
        print("-"*60)
        print(person+" (in memoriam): vítima inocente.")
print("-"*60)
```

```
linhas = int(input())
if linhas < 1 or linhas > 100:
          print("Valor inválido na entrada.")
          exit()
pessoas = {} # Dicionário para pessoas
detetives = {} # Dicionário para detetives
assassinos = {} # Dicionário para assassinos
# Entrada dos dados
for i in range(linhas):
          assassino, vitima, detetive = input().split()
          pessoas[vitima] = False
          # Adiciona os nomes nos respectivos dicionários,
          # adequando os valores nas chaves se elas já existissem.
          if not detetive in pessoas:
                    pessoas[detetive] = True
          if not assassino in pessoas:
                    pessoas[assassino] = True
          if (detetive in detetives):
                    detetives[detetive] += 1
          else:
                    detetives[detetive] = 1
          if (assassino in assassinos):
                    assassinos[assassino].append(vitima)
          else:
                    assassinos[assassino] =[vitima]
# Monta a saída para todas as pessoas.
for person in sorted(pessoas):
          # Primeiro, a verificação para detetive, já que uma mesma pessoa
          # pode estar nos dicionários detetives e assassinos.
          if person in detetives:
                    print("-"*60)
                    if pessoas[person]==False:
                              isdead = ' (in memoriam)'
                    else:
                              isdead = ''
                    print(person+isdead+": detetive.")
                    print("  Resolveu %d caso(s)." % detetives[person])
                    if person in assassinos:
                              kill_vitima = 0
                              kill_assassinos = 0
                              kill_detetives = 0
                              for vitima in assassinos[person]:
                                        if vitima in detetives:
                                                  kill_detetives += 1
                                        elif vitima in assassinos:
                                                  kill_assassinos += 1
                                        else:
                                                  kill_vitima += 1
                              if kill_detetives > 0:
                                        print("  Matou %d detetive(s)." % kill_detetives)
                              if kill_assassinos > 0:
                                        print("  Matou %d assassinos(s)." %
                    kill_assassinos)
                              if kill_vitima > 0:
                                        print("  Matou %d inocente(s)." % kill_vitima)
          # Depois, se essa pessoa não estiver em detetives, mas estiver em assassinos:
          elif person in assassinos:
                    print("-"*60)
                    if pessoas[person]==False:
                              isdead = ' (in memoriam)'
                    else:
                              isdead = ''
                    print(person+isdead+": assassino(a).")
                    kill_vitima = 0
                    kill_assassinos = 0
                    kill_detetives = 0
                    for vitima in assassinos[person]:
                              if vitima in detetives:
                                        kill_detetives += 1
                              elif vitima in assassinos:
                                        kill_assassinos += 1
                              else:
                                        kill_vitima += 1
                    if kill_detetives > 0:
                              print("  Matou %d detetive(s)." % kill_detetives)
                    if kill_assassinos > 0:
                              print("  Matou %d assassinos(s)." % kill_assassinos)
                    if kill_vitima > 0:
                              print("  Matou %d inocente(s)." % kill_vitima)
          # Por último, vítima inocente
          else:
                    print("-"*60)
                    print(person+" (in memoriam): vítima inocente.")
print("-"*60)
```