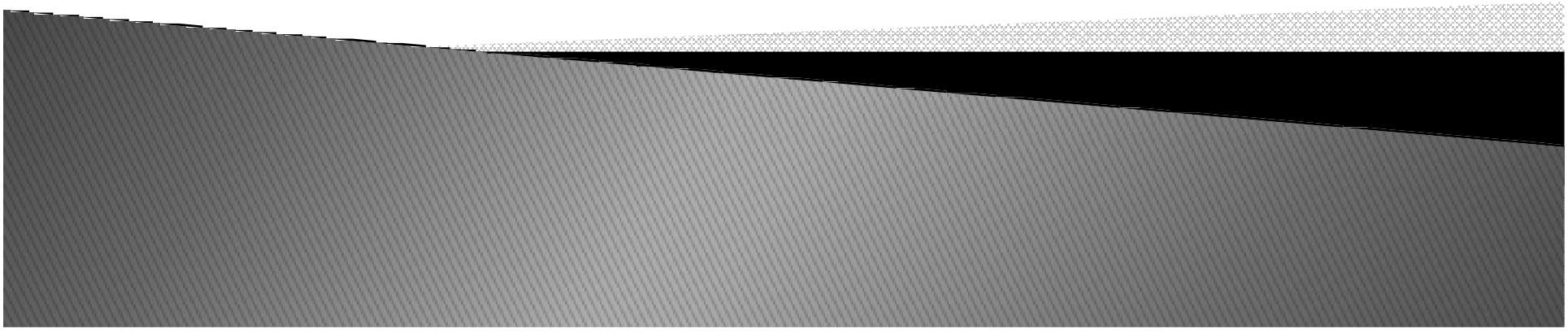
Pesquisa Sequencial e Binária

Gustavo Callou
gcallou@gmail.com

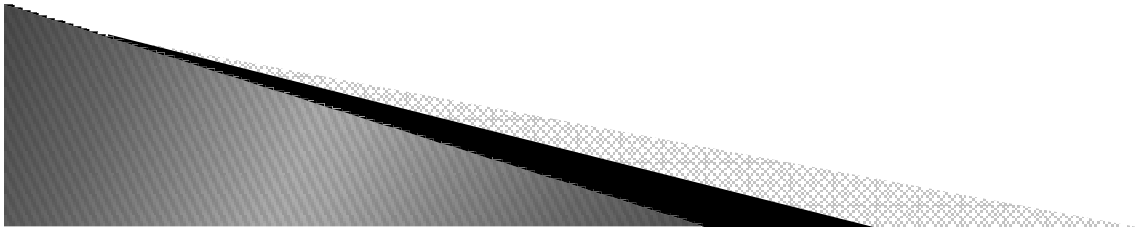


Contextualização

- Apresentaremos e discutiremos diferentes estratégias para efetuarmos a pesquisa (busca) de um elemento específico em um conjunto de dados.

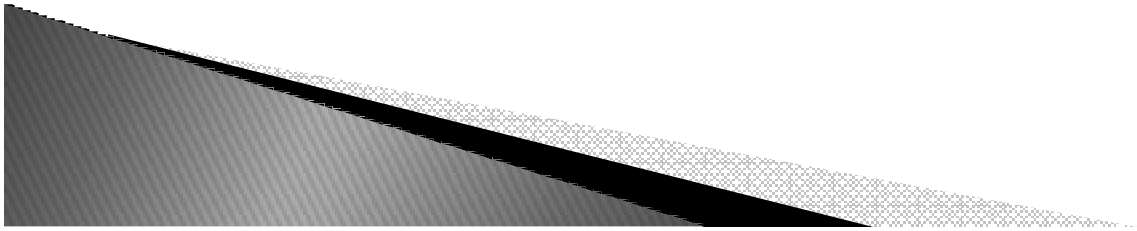
- Esta operação é muito importante, pois é encontrada com muita freqüência em diversas aplicações.

- Apresentaremos dois métodos:
  - Pesquisa Seqüencial (linear search ou sequencial search)
  - Busca Binária (binary search)



Pesquisa Sequencial

- Métodologia Básica:
  - É efetuada a verificação de cada elemento do conjunto, sequencialmente, até que o elemento desejado seja encontrado (pesquisa bem sucedida) ou que todos os elementos do conjunto tenham sido verificados sem que o elemento procurado tenha sido encontrado (pesquisa mal sucedida).



Pesquisa Sequencial

- Pesquisa Sequencial (PS)
  - Forma **mais simples** de realizar pesquisas.
  - **Metodologia**: Percorre o vetor, **elemento por elemento**, verificando se o elemento desejado está presente no vetor.

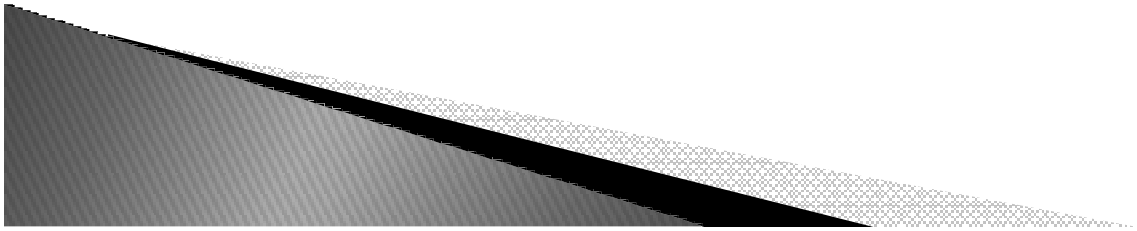
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
14	21	5	45	12	3	86	98	46	53	24	2	1	15	90	47

Pergunta: Como verificar se o **elemento 90** está presente no vetor acima?

Pergunta: Quantas comparações são necessárias para achar o **elemento 90**?

Características

- Algoritmo extremamente simples;
- Pode ser muito ineficiente quando o conjunto de dados se torna muito grande.



Desempenho Computacional

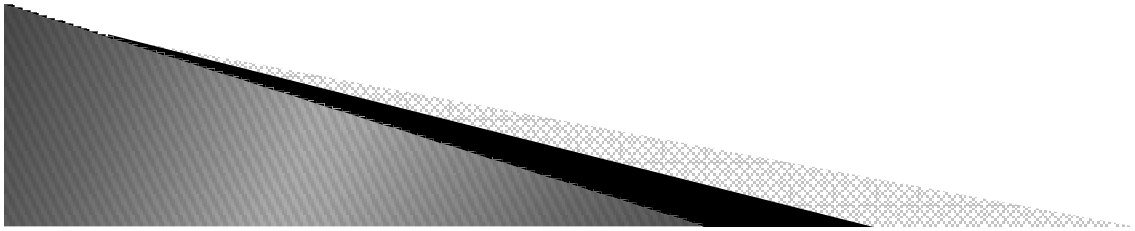
- **Pior Caso**: é quando é necessário realizar n comparações (onde n é o número de elementos);

  Qual o cenário de pior caso possível ?

- **Melhor Caso**: é quando é necessário realizar somente uma comparação;

  Qual o cenário de melhor caso possível ?

- **Caso Médio**: (Pior Caso + Melhor Caso)/2.



Desempenho Computacional

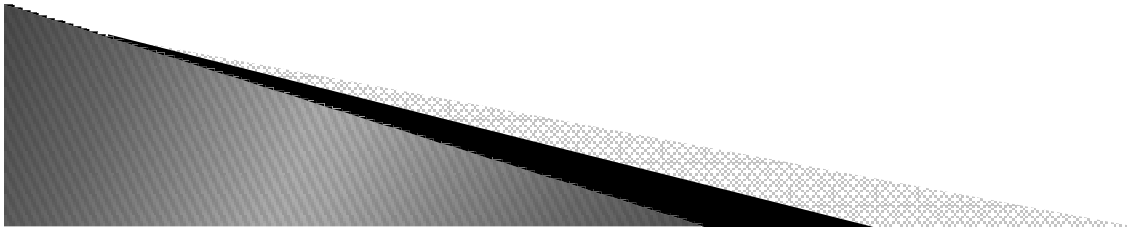
- **Pior Caso**: é quando é necessário realizar n comparações (onde n é o número de elementos);

  Qual o cenário de pior caso possível ?
  Elemento procurado é o último elemento do vetor.

- **Melhor Caso**: é quando é necessário realizar somente uma comparação;

  Qual o cenário de melhor caso possível ?
  Elemento procurado corresponde ao primeiro elemento do vetor.

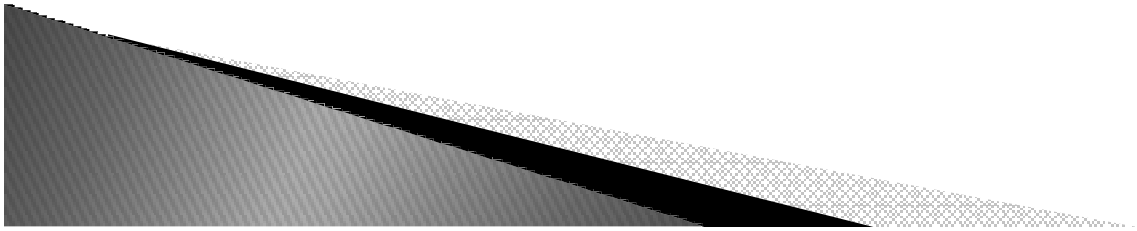
- **Caso Médio**: (Pior Caso + Melhor Caso)/2.



Análise de Complexidade da PS

- Pode-se **desconsiderar** os casos extremos (**melhor** e **pior** caso).

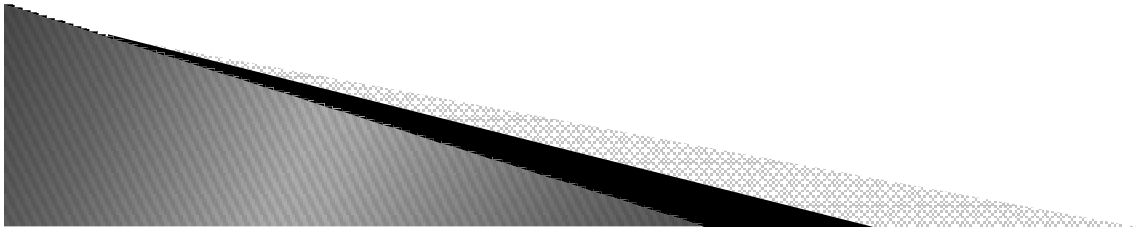
- Portanto, qual a complexidade do algoritmo de busca seqüencial sobre vetores ?



Análise de Complexidade da PS

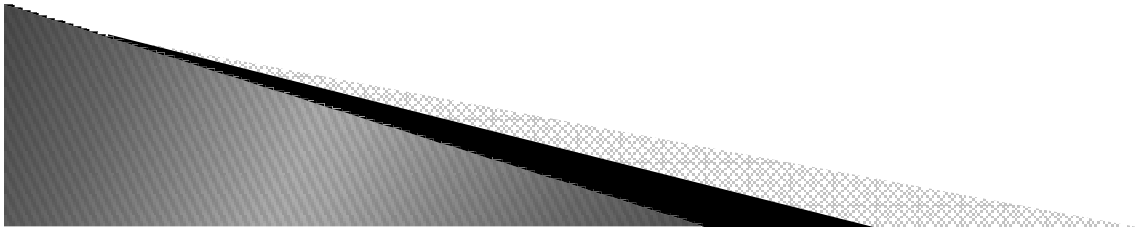
- Pode-se **desconsiderar** os casos extremos (**melhor** e **pior** caso).

- Portanto, qual a complexidade do algoritmo de busca seqüencial sobre vetores ? **O(n)**



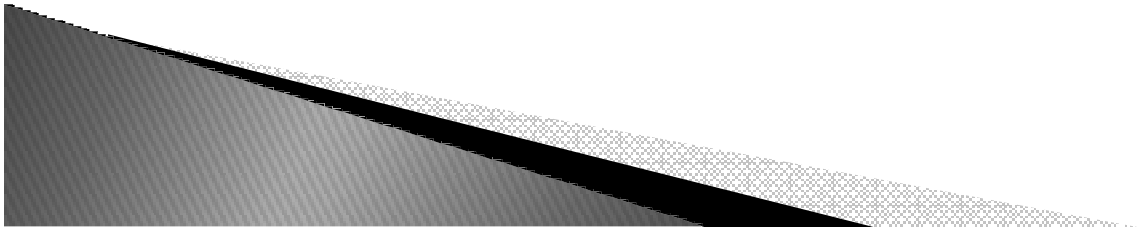
Implementação

Faça uma função em python que realize a pesquisa sequencial em uma lista.



Busca Binária

- Algoritmo de busca em vetores que requer acesso aleatório aos elementos do mesmo.
- Parte do pressuposto de que o vetor está ordenado.
- Realiza sucessivas divisões do espaço de busca e compara o elemento buscado (chave) com o elemento no meio do vetor.
- 3 opções:
  - Se igual, a busca termina com sucesso.
  - Se o elemento do meio for menor que o elemento buscado, então a busca continua na metade posterior do vetor.
  - Se o elemento do meio for menor que a chave, a busca continua na metade anterior do vetor.



Busca Binária

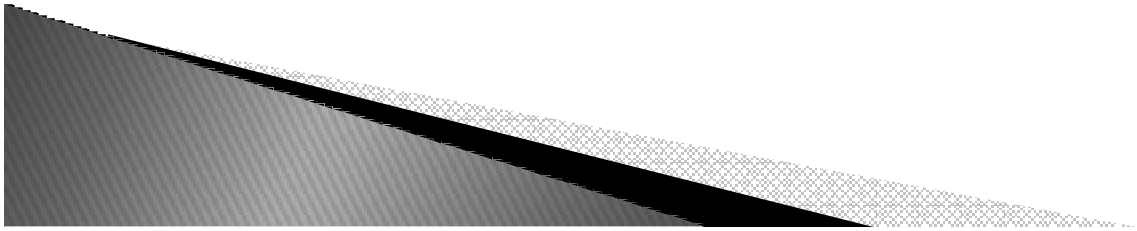
- Metodologia

  1) Checar onde está o **ponto médio** do vetor.

  2) Comparar o **elemento do ponto médio** (EPM) com elemento **chave.**

  3) Caso não encontre o dado no passo 2, continuar a pesquisa da seguinte forma:

  - Caso chave=EPM, então a pesquisa para com sucesso, pois achou o dado desejado!
  - Caso chave<EPM realizar a pesquisa no sub-vetor a esquerda do EPM, partindo do **passo 1**.
  - Caso chave>EPM realizar a pesquisa no sub-vetor a direita do EPM, partindo do **passo 1**.



Exemplo de Busca Binária

Exemplo Inicial:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
14	21	5	45	12	3	86	98	46	53	24	2	1	15	90	47

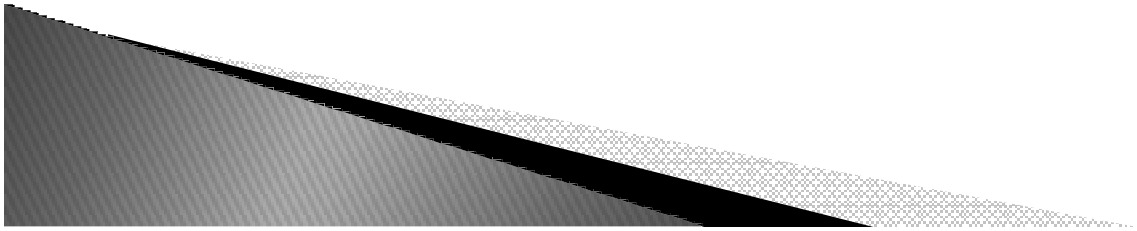
Após ordenação:

!!!???

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	5	12	14	15	21	24	45	46	47	53	86	90	98

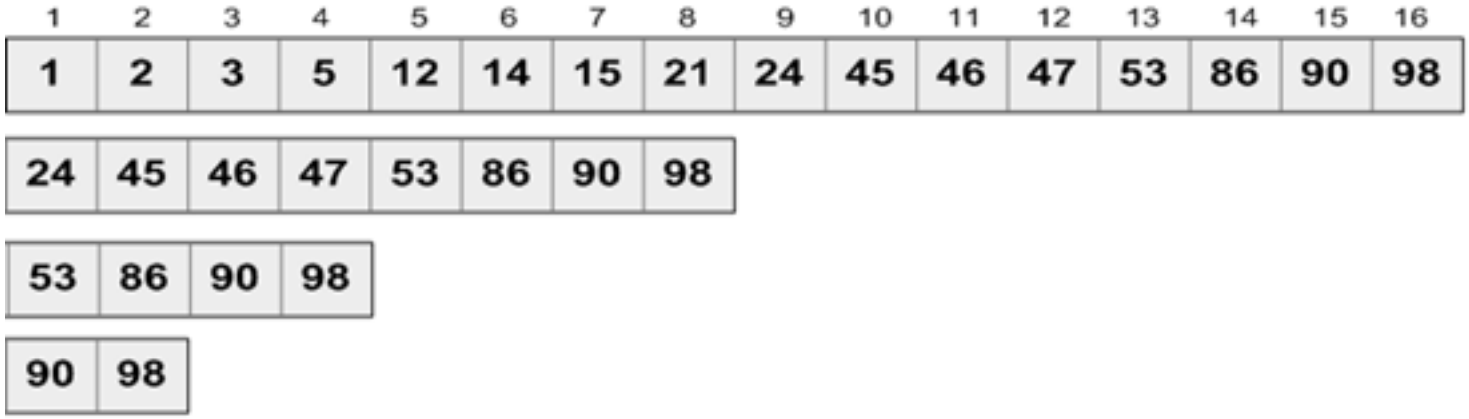
Pergunta: Como verificar se o **elemento 90** está presente no vetor acima?

Pergunta: Quantas comparações são necessárias para achar o **elemento 90**?



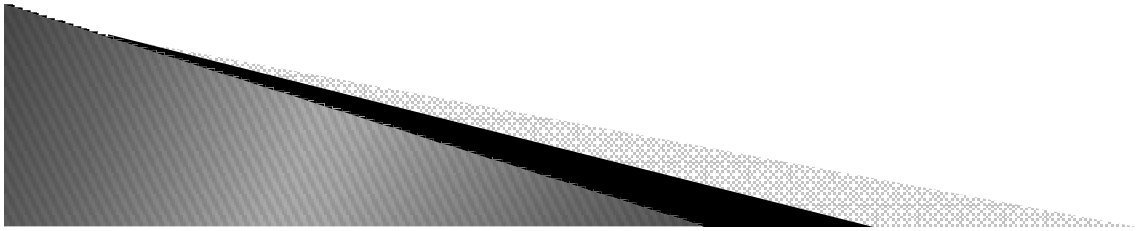
Exemplo de Busca Binária

- Procurando pelo elemento 90



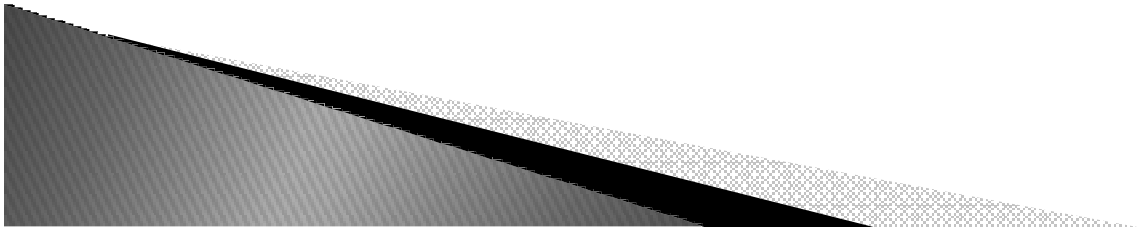
Complexidade da Busca Binária

- **Pior Caso**: quando o dado desejado encontra-se na folha da árvore ou não existe. Portanto: $O(\log_2 n)$
- **Melhor Caso**: quando o elemento procurado corresponde a exatamente o elemento do meio do vetor (raiz da árvore). $O(1)$
- **Caso Médio**: quando o dado desejado encontra-se próximo do "meio" da árvore. Portanto: $O(\log_2 n)$



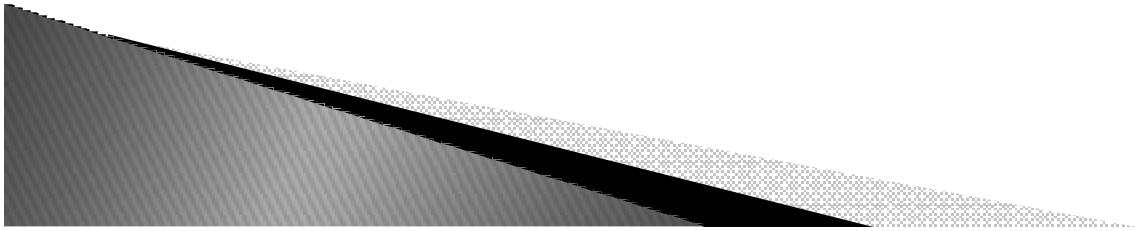
Pesquisa Sequencial versus Busca Binária

qtd de dados	O(n)	$O(\log_2 n)$
10	1,00E+01	3,32E+00
50	5,00E+01	5,64E+00
100	1,00E+02	6,64E+00
500	5,00E+02	8,97E+00
→ 1.000	1,00E+03	9,97E+00
5.000	5,00E+03	1,23E+01
10.000	1,00E+04	1,33E+01
50.000	5,00E+04	1,56E+01
100.000	1,00E+05	1,66E+01
500.000	5,00E+05	1,89E+01
→ 1.000.000	1,00E+06	1,99E+01
5.000.000	5,00E+06	2,23E+01
10.000.000	1,00E+07	2,33E+01



Implementação Busca Binária

```
def bs(key, lista):
    left = 0
    right = len(lista) -1
    while(left <= right):
        middle = int((left+right)/2)
        print middle
        if(key == lista[middle]):
            return middle
        elif(key > lista[middle]):
            left = middle + 1
        else:
            right = middle -1
    else:
        return -1
```



Exercício

Considere o vetor com 11 elementos abaixo e diga quantas **comparações de igualdade** realizam os algoritmos de **Busca Linear** e **Busca Binária**, na tentativa de se encontrar no vetor os valores:

a) 3

b) 25

c) 70

1	2	7	15	23	56	57	58	70	72	78