# Busca em lista ordenada


Fonte: http://www.php5dp.com/

PF 7.1 a 7.8
http://www.ime.usp.br/~pf/algoritmos/aulas/bub

# Busca em lista ordenada

Um lista v[0 : n] é **crescente** se

$$v[0] \leq v[1] \leq v[2] \leq \cdots \leq v[n-1].$$

**Problema**: Dado um número x e um lista crescente v[0 : n] encontrar um índice m tal que v[m]==x.

Entra: x == 50

| | 0 | | | | | | 7 | | | | n |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

Sai: m == 7

# Busca em lista ordenada

Entra: x == 57

| | 0 | | | | | | | | | | n |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

Sai: m == None (x não está em v)

# Busca sequencial

```
def busca_sequencial(x, v):
    '''(numero,lista) -> int ou None
    Recebe x e um lista cresc.  v e ret.
    um índice m tal que v[m] == x.
    Se não existe m retorna None.
    '''
0   n = len(v)
1   m = 0
2   while m < n and v[m] < x:  # /*1*/
3       m += 1
4   if m < n and v[m] == x
5       return m
6   return None
```

# Exemplo

x == 55

| | 0 | | | | | | | | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

| | m | | | | | | | | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

| | 0 | m | | | | | | | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

| | 0 | | m | | | | | | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

| | 0 | | | m | | | | | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

# Exemplo

x == 55

| | 0 | | | | m | | | | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

| | 0 | | | | | m | | | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

| | 0 | | | | | | m | | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

| | 0 | | | | | | | m | | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

| | 0 | | | | | | | | m | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

## Correção

Relação **invariante** chave:

(i0) em /*1*/ vale que: $v[m-1] < x$. ♡

x == 55

| | 0 | | | | | | | m | | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

A relação (i0) vale no começo da primeira iteração se supusermos que $v[-1] = -\infty$.

No início da última iteração $m \geq n$ ou $v[m] \geq x$.

Portanto, se a função retorna None, então x não está em $v[0 : n]$

## Consumo de tempo busca_sequencial

Se a execução de cada linha de código consome 1 **unidade** de tempo o consumo total é:

| linha | todas as execuções da linha | |
|---|---|---|
| 0-1 | $= 1$ | $= 1$ |
| 2 | $\leq n + 1$ | $\approx n$ |
| 3 | $\leq n$ | $= n$ |
| 4 | $= 1$ | $= 1$ |
| 5 | $\leq 1$ | $\leq 1$ |
| 6 | $\leq 1$ | $\leq 1$ |
| total | $\leq 2\,n + 5$ | $= O(n)$ |

## Conclusão

O consumo de tempo do algoritmo busca_sequencial no pior caso é proporcional a n.

O consumo de tempo do algoritmo busca_sequencial é $O(n)$.

## Busca binária

```
 def busca_binaria(x, v):
0   n = len(v)
1   e = 0
2   d = n
3   while e < d:  # /*1*/
4      m = (e + d) // 2
5      if v[m] == x:  return m
6      if v[m] <  x:  e = m + 1
7      else:  d = m
8   return None
```

## Exemplo

x == 48

| | 0 | | | | | 5 | | | | | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
| | e | | | | | 5 | | | | | d |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
| | e | | | | | m | | | | | d |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
| | 0 | | | | | 5 | e | | | | d |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
| | 0 | | | | | 5 | e | m | | | d |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

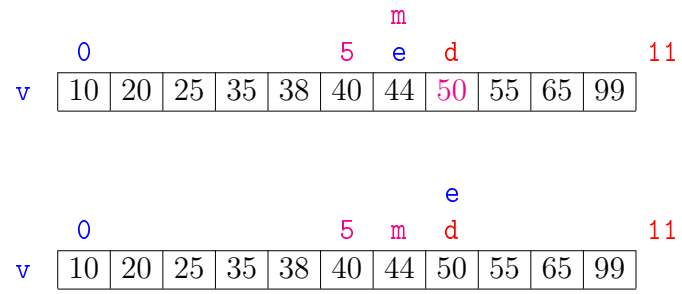
## Exemplo

x == 48

| | 0 | | | | | 5 | e | | d | | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
| | 0 | | | | | 5 | e | m | d | | 11 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |
| | | | | | | | | d | | | |
| | 0 | | | | | 5 | e | m | | | 11 |
| v | 10 | 20 | 25 | 35 | 38 | 40 | 44 | 50 | 55 | 65 | 99 |

## Exemplo

x == 48

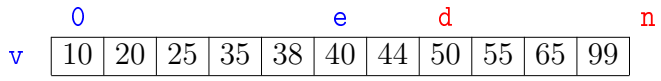



## Correção

Relação **invariante** chave:

(i0) em /*1*/ vale que: v[e] $\leq$ x $<$ v[d]. ♡

x == 48



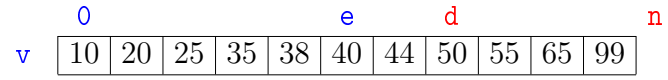

A relação (i0) vale no começo da primeira iteração se supusermos que v[0] $\leq$ x e v[n] $= +\infty$.

## Correção

Relação **invariante** chave:

(i0) em /*1*/ vale que: v[e] $\leq$ x $<$ v[d]. ♡

x == 48



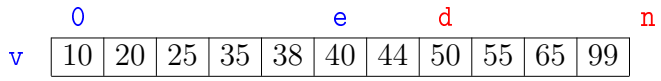

No início da última iteração quando e == d nenhum elemento é "$\geq$ v[e]" e "$<$ v[d]", pois o lista é crescente (!). Logo, x não está em v[0 : n] e a função retorna None

## Correção

Relação **invariante** chave:

(i0) em /*1*/ vale que: v[e] $\leq$ x $<$ v[d]. ♡

x == 48




O valor de d $-$ e diminui a cada iteração. Portanto, se a função não encontra m tal que v[m]==x, então a função para quando d $-$ e $\leq 0$.

## Consumo de tempo busca_binaria

O consumo de tempo da função busca_binaria é proporcional ao número k de iterações do while.

No início da 1a. iteração tem-se que d $-$ e $=$ n.

Sejam

$$(e_0, d_0), (e_1, d_1), \ldots, (e_k, d_k),$$

os valores das variáveis e e d no início de cada uma das iterações. No pior caso x não está em v. Assim, $d_{k-1} - e_{k-1} > 0$ **e** $d_k - e_k \leq 0$

## Número iterações

Estimaremos o valor de k em função de d $-$ e.

Note que $d_{i+1} - e_{i+1} \leq (d_i - e_i)/2$ para i=1,2,...,k-1.

Desta forma tem-se que

$$\begin{array}{lcccl}
d_0 - e_0 & = & n & \leq & n \\
d_1 - e_1 & \leq & (d_0 - e_0)/2 & \leq & n/2 \\
d_2 - e_2 & \leq & (d_1 - e_1)/2 & \leq & (n/2)/2 = n/2^2 \\
d_3 - e_3 & \leq & (d_2 - e_2)/2 & \leq & (n/2^2)/2 = n/2^3 \\
d_4 - e_4 & \leq & (d_3 - e_3)/2 & \leq & (n/2^3)/2 = n/2^4 \\
\vdots & & \vdots & & \vdots \quad \vdots
\end{array}$$

## Número iterações

Percebe-se que depois de cada iteração o valor de $d - e$ é reduzido pela metade.
Seja $t$ o número inteiro tal que

$$2^t \leq n < 2^{t+1}$$

Da primeira desigualdade temos que

$$t \leq \lg n,$$

onde $\lg n$ denota o logaritmo de $n$ na base 2.

## Número iterações

Da desigualde estrita, concluímos que

$$0 < d_{k-1} - e_{k-1} \leq n/2^{k-1} < 2^{t+1}/2^{k-1}.$$

Assim, em particular temos que

$$1 \leq 2^{t+1}/2^{k-1}$$

ou, em outras palavras

$$k \leq t + 2.$$

Portanto , o número $k$ de iterações é não superior a

$$t + 2 \leq \lg n + 2.$$

## Conclusão

O consumo de tempo do algoritmo `busca_binaria` no pior caso é proporcional a $\lg n$.

O consumo de tempo do algoritmo `busca_binaria` é $O(\lg n)$.

## Número de iterações

| busca_sequencial | busca_binaria |
|---|---|
| $n$ | $\lg n$ |
| 256 | 8 |
| 512 | 9 |
| 1024 | 10 |
| 2048 | 11 |
| 4096 | 12 |
| 8192 | 13 |
| 16384 | 14 |
| 32768 | 15 |
| 65536 | 16 |
| 262144 | 18 |
| 1048576 | 20 |
| ⋮ | ⋮ |
| 4294967296 | 32 |

## Versão recursiva da busca binária

Para formular uma versão recursiva vamos generalizar um pouco o problema trocando `v[0 : n]` por `v[e : d]`.

```
 def busca_binaria(x, v):
0    n = len(v)
1    return busca_binariaR(x,0,n,v)
```

## Versão recursiva da busca binária

Recebe um lista crescente `v[e : d]` e retorna um índice `m`, $e \leq m < d$ tal que `v[m] == x`. Se tal `m` não existe, retorna `None`.

```
 def busca_binariaR(x, e, d, v):
1    if d <= e:  return None
2    m = (e + d) // 2
3    if v[m] == x:  return m
4    if v[m] <  x:
5       return busca_binariaR(x,m+1,d,v)
6    return busca_binariaR(x,e,m,v)
```

## Outra versão recursiva

**A função abaixo não resolve o problema...**
Por quê? Como consertar?

```
 def busca_binariaR(x, v):
0    n = len(v)
1    if n == 0:  return None
2    m = n // 2
3    if v[m] == x:  return m
4    if v[m] <  x:
5       return busca_binariaR(x,v[m+1:])
6    return busca_binariaR(x,m,v)
```