01. LEMA: Seja t(n) uma função de naturais em reais que respeite as seguintes equações de recorrência
(como, por exemplo, a função do tempo de execução da busca binária recursiva):

```
    t(1) = c_1.
    t(m) = c_2 + t(y), para algum y <= piso(m/2).
```

Nós mostraremos que t(m) <= c_1 + c_2*piso(lg m),
para todo natural m >= 1.

```
=====================================
--------------- PROVA ---------------
```

PROVA: por indução em m:

* Base (m = 1):
  Temos que mostrar que t(m) <= c_1 + c_2*piso(lg m).
  Como m = 1, temos:
  t(m) = c_1 = c_1 + c_2*0 = c_1 + c_2*piso(0) = c_1 + c_2*piso(lg 1), como desejado.

* Hipótese de indução: para todo m' < m, t(m') <= c_1 + c_2*piso(lg m').

* Passo:
  Temos que mostrar que t(m) <= c_1 + c_2*piso(lg m).
  Temos:

```
  t(m) = c_2 + t(y), para algum y <= piso(m/2) < m.
      <= c_2 + c_1 + c_2*piso(lg y)  --> pela H.I., pois y < m
      <= c_2 + c_1 + c_2*piso(lg piso(m/2))
      <= c_2 + c_1 + c_2*piso(lg m/2) --> pois piso(m/2) <= m/2
       = c_2 + c_1 + c_2*piso(lg m - lg 2)
       = c_2 + c_1 + c_2*piso(lg m - 1)
       = c_2 + c_1 + c_2*piso(lg m) - c_2
       = c_1 + c_2*piso(lg m), CQD.
```

```
--------------- PROVA ---------------
=====================================
```

02. Algoritmo recursivo para cálculo do máximo de um vetor:

```
==================================================
Algoritmo: máx_rec
Entrada: um vetor A[1..n] de números reais
Saída: um número real
--------------------------------------------------
SE n=1
| RETORNE A[1]
SENÃO
| RETORNE máx( A[1] , máx_rec(A[2..n]) )
==================================================
```

03. O tempo de execução do algoritmo acima é dado por:

```
  t(1) = c_1
  t(n) = c_2 + t(n-1) , para n > 1.
```

04. LEMA: t(n) = c_1 + c_2*(n-1), para todo n >= 1.

```
=====================================
--------------- PROVA ---------------
```

Por indução em n:

* BASE (n=1): t(1) = c_1 = c_1 + c_2(n-1).

```
* HIPÓTESE DE INDUÇÃO: t(n') = c_1 + c_2*(n'-1), para todo n' < n.

* PASSO (n > 1): temos:
  t(n) = c_2 + t(n-1)
       = c_2 + c_1 + c_2*(n-2)
       = c_1 + c_2*(n-1), CQD.
--------------- PROVA ---------------
=====================================
```

5. EXEMPLO: considere a função definida pelas equações abaixo (e que é semelhante àquela do tempo de execução do "mergesort"):

```
   t(1) = 1
   t(n) = 2*t(piso(n/2)) + n
```

6. LEMA: t(n) = O(n*lg(n)), para todo n >= 1.

```
=====================================
--------------- PROVA ---------------
Nós mostraremos o resultado mostrando a seguinte inequação:
  t(n) <= n + n*teto(lg n).

Por indução em n:

* BASE (n=1): t(1) = 1 = 1 + 0 = 1 + 1*teto(lg 1).

* HIPÓTESE DE INDUÇÃO: t(n') <= n' + n'*teto(lg n'), para todo n' < n.

* PASSO (n>1): temos:
  t(n) = 2*t(piso(n/2)) + n
      <= 2 * ( piso(n/2) + piso(n/2)*teto(lg piso(n/2)) ) + n --> pois
"piso(n/2) < n"
      <= 2 * ( n/2 + (n/2)*teto(lg n/2) ) + n --> pois piso(n/2) <= n/2
       = n + n*teto(lg n/2) + n
       = n + n*teto(lg n - lg 2) + n
       = n + n*teto(lg n - 1) + n
       = n*teto(lg n) + n, CQD.
--------------- PROVA ---------------
=====================================
```

7. EXEMPLO:

```
==================================================
Algoritmo: máx_rec_2
Entrada: um vetor A[1..n] de números reais
Saída: um número real
--------------------------------------------------
SE n=1
| RETORNE A[1]
SENÃO
| m := piso((1+n)/2)
| RETORNE máx( máx_rec_2(A[1..m]) , máx_rec_2(A[m+1..n]) )
==================================================
```

O custo desse algoritmo é:

```
  t(1) = a
  t(n) = b + t(piso(n/2)) + t(teto(n/2))
```

8. EXERCÍCIO: mostre que t(n) = O(a*n), ou, se preferir, que t(n) = O(n) (naturalmente, os resultados são equivalentes, pois "a" é constante).

9. EXERCÍCIO: mostre que t(n) = Ômega(n).