```
1. EXERCÍCIO (Lista 1 Rudini, q3): limite t(n) = 2*t(n-2) + 1  assintoticamente.
================================================================================
RESPOSTA:

Pelo método da árvore de recursão, vemos que t(n) equivale aproximadamente à
soma dos termos

  1 + 2 + 4 + ... + 2^(k-1),

onde n - 2k = 0 -- ou seja, k = n/2.

Como a soma dos termos de uma P.G. é a*(1 - r^n)/(1 - r), temos:

  1 + 2 + 4 + ... + 2^(k-1) = 1*(1 - 2^(n/2))/(1 - 2)
                            = sqrt(2^n) - 1
                            = θ(sqrt(2^n)).

O nosso palpite é, portanto, que t(n) = θ(sqrt(2^n)).

Nós podemos provar o palpite acima como segue. Seja

  t(n) = 2*t(n-2) + 1, se n > k,
  t(n) = a,            se 0 <= n <= k,

onde "a" e "k" são naturais tais que k >= 1.

Nós mostraremos primeiramente que t(n) = O(sqrt(2^n)).

1.1. EXERCÍCIO: ∀n >= 0, t(n) >= 0.

1.2. LEMA: t(n) = O(sqrt(2^n)).
     ----
======================================================================
PROVA:

Temos que mostrar que existem um real "c" e um natural "m" positivos
tais que, ∀n >= m, 0 <= t(n) <= c*sqrt(2^n).

Sejam m = 1, m' = máx{2,k} e c = 1 + máx{ t(i) : ∀ 1 <= i <= m' }.

Nós mostraremos então que, ∀n >= m, 0 <= t(n) <= c*sqrt(2^n). O
exercício 1.1 implica que, ∀n >= m, 0 <= t(n). Resta, então, mostrar
que, ∀n >= m, t(n) <= c*n, e nós o faremos por indução forte:

  * TESE: ∀n >= m, t(n) <= c*sqrt(2^n) - 1.

  * BASE (1 <= n <= m'): Temos:

      t(n) <= c - 1   ---> pois c >= 1 + máx{ t(i) : ∀ 1 <= i <= m' }.
           <= c*sqrt(2) - 1   ---> pois sqrt(2) >= sqrt(1) = 1.
            = c*sqrt(2^n) - 1, CQD. ---> pois 2^n >= 2, já que n >= 1.

  * H.I.: ∀ 1 <= l < n, t(l) <= c*sqrt(2^l) - 1.

  * PASSO (n > m'): Temos:

      t(n)  = 2*t(n-2) + 1                   ---> pois n > k.
           <= 2*(c*sqrt(2^(n-2)) - 1) + 1    ---> H.I. (1 <= n-2 < n).
            = c*sqrt(2^2)*sqrt(2^(n-2)) - 2 + 1
            = c*sqrt(2^2 * 2^(n-2)) - 1
            = c*sqrt(2^n) - 1, CQD.

Logo, ∀n >= m, t(n) <= c*sqrt(2^n) - 1 <= c*sqrt(2^n), CQD.
======================================================================
```

```
1.3. LEMA: t(n) = Ω(sqrt(2^n)).
     ----
======================================================================
PROVA:

Temos que mostrar que existem um real "c" e um naturam "m" positivos
tais que, ∀n >= m, 0 <= c*sqrt(2^n) <= t(n).

Sejam m = 1, m' = máx{2,k} e c = t(n) / sqrt(2^m').

Temos que mostrar então que, ∀n >= m, 0 <= c*sqrt(2^n) <= t(n).
Observe que a inequação 0 <= c*sqrt(2^n) é trivialmente verdadeira,
pois c,sqrt(2^n) >= 0. Logo, resta mostrar que, ∀n >= m,
c*sqrt(2^n) <= t(n), e nós o faremos por indução forte:

  * TESE: ∀n >= m, c*sqrt(2^n) <= t(n).

  * BASE (1 <= n <= m'): Temos:

      t(n) >= c*sqrt(2^n), CQD.  ---> c <= t(n)/sqrt(2^m')
                                        <= t(n)/sqrt(n).

  * H.I.: ∀ 1 <= l < n, c*sqrt(2^l) <= t(l).

  * PASSO (n > m'): Temos:

      t(n)  = 2*t(n-2) + 1
           >= 2*c*sqrt(2^(n-2)) + 1    ---> H.I. (1 <= n-2 < n).
           >= 2*c*sqrt(2^(n-2))
            = c*sqrt(2^2)*sqrt(2^(n-2))
            = c*sqrt(2^2 * 2^(n-2))
            = c*sqrt(2^n), CQD.

======================================================================

Logo, t(n) = θ(sqrt(2^n)), CQD.

================================================================================
```

2. EXERCÍCIO (aula_correcao_de_alg_3.pdf): dado o algoritmo

```
=======================================================
Algoritmo: máx
Entrada: um vetor A[1..n].  // n >= 1
Saída: um elemento do vetor.
-------------------------------------------------------
1. m := A[1] ; i := 2
   // VARIANTE: n -i +1
   // INVARIANTES: * 2 <= i <= n+1.
   //             * ∀j ∈ {1..i-1}, m >= A[j].
   //             * ∃j ∈ {1..i-1} tal que A[j] = m.
2. ENQUANTO i <= n
3. | SE A[i] > m
4. | | m := A[i]
5. | ++i
6. RETORNE m.
=======================================================
```

e supondo que os invariantes e variante acima estão corretos, prove a correção total do algoritmo com relação à seguinte especificação:

"Toda chamada máx(A) termina e retorna um valor "k" tal que

a) $\forall j \in \{1..n\}$, $A[j] \le k$.
b) $\exists j \in \{1..n\}$ tal que $A[j] = k$."

==

RESPOSTA:

Temos que argumentar primeiramente que toda chamada máx(A) chega ao fim. Dado o código do algoritmo, é necessário mostrar apenas que, em toda execução do algoritmo, o laço efetivamente chega ao fim. Como o laço possui um variante, então os casos possíveis são:

* CASO 1: o laço termina.

* CASO 2: o laço não termina. Nesse caso, necessariamente alguma das iterações do laço não termina, mas isso é impossível, dado que o corpo do laço é composto apenas por instruções sequenciais.

Nós concluímos, então, que o laço do algoritmo sempre chega ao fim, e portanto que toda execução do algoritmo termina.

Seja agora uma execução do algoritmo que retorna um valor "k". Pelo código do algoritmo, "k" é o valor da variável "m" logo antes de a linha 6 ser executada, e portanto logo após a execução do laço. Nesse estado do algoritmo, observe:

1) Pelo primeiro invariante do laço, $i \le n+1$, e, como a condição do laço é falsa (pois o laço terminou), $i > n$. Logo, $i = n+1$.

2) Pelo segundo invariante do laço, temos

   $\forall j \in \{1..(n+1)-1\}$, $k \ge A[j]$,

   o que prova o item "a" do enunciado.

3) Pelo terceiro invariante, temos

   $\exists j \in \{1..(n+1)-1\}$ tal que $A[j] = k$,

   o que prova o item "b" do enunciado, CQD.

==