SSC0503 - Introdução à Ciência de Computação II
Respostas da 3ª Lista
Professor: Claudio Fabiano Motta Toledo (claudio@icmc.usp.br)
Estagiário PAE: Jesimar da Silva Arantes (jesimar.arantes@usp.br)

Resposta pergunta 1:

```c
#include <stdio.h>
#include <string.h>

int decToBin(int n);
int mult(int a, int b);
int mdc(int x, int y);
int palindromoRec(char str[], int i);

void testMult();
void testBin();
void testMDC();
void testPalindromo();

int main(){
 testBin();
 testMult();
 testMDC();
 testPalindromo();
 return 0;
}

int decToBin(int n){
 if (n == 1){
  return 1;
 }
 return 10*decToBin(n/2) + (n%2);
}

void testBin(){
 int dec = 20;
 int bin = decToBin(dec);
 printf("Dec[%d] -> Bin[%d]\n", dec, bin);
}

int mult(int a, int b){
 if (a == 0){
  return 0;
 }
 return b + mult(a - 1, b);
}

void testMult(){
 int a = 6;
 int b = 4;
 int m = mult(a, b);
 printf("mult(%d, %d) = %d\n", a, b, m);
```

```c
}

int mdc(int x, int y){
  if (x > y && x % y == 0){
    return y;
  }else if (x < y){
    return mdc(y, x);
  }else{
    return mdc(y, x%y);
  }
}

void testMDC(){
  int x = 28;
  int y = 14;
  int v = mdc(x, y);
  printf("MDC(%d, %d) = %d\n", x, y, v);
}

int palindromoRec(char str[], int i){
  if (i >= strlen(str)){
    return 1;
  }
  if (str[i] != str[strlen(str)-i-1]){
    return -1;
  }
  return palindromoRec(str, i+1);
}


void testPalindromo(){
  char str[] = "amoraroma";
  int p = palindromoRec(str, 0);
  printf("Palindromo: %d\n", p);
}
```

Listing 1: Resposta do exercício 1 codificado na linguagem C

Resposta pergunta 2:

- $T(n) = T(n-1) + c$
  Expandindo a expressão temos:
  $T(n) = T(n-1) + c$
  $T(n) = T(n-2) + c + c$
  $T(n) = T(n-3) + c + c + c$
  ...
  $T(n) = T(n-k) + kc$
  Fazendo $n - k = 1$, então $k = n - 1$ e substituindo temos:
  $T(n) = T(n-(n-1)) + (n-1)c$
  $T(n) = T(1) + (n-1)c$
  $T(n) = 0 + nc - c$
  $T(n) = nc - c$

Logo, $T(n)$ é linear, ou ainda, $T(n) = O(n)$
Aplicando o método da substituição para comprovar temos:
Hipótese Indutiva (Valido para $T(n)$)
$T(n) = O(n)$
$T(n) \leq cn$
Passo de Indução (Valido para $T(n-1)$)
$T(n) \leq c(n-1) + c$
$T(n) \leq cn - c + c$
$T(n) \leq cn$
Dessa forma, qualquer valor de $c > 0$ resolvem o problema.

- $T(n) = cT(n-1)$
  Expandindo a expressão temos:
  $T(n) = cT(n-1)$
  $T(n) = c \cdot cT(n-2)$
  $T(n) = c \cdot c \cdot cT(n-3)$
  $\cdots$
  $T(n) = c^k T(n-k)$
  Fazendo $n - k = 0$, então $k = n$ e substituindo temos:
  $T(n) = c^n T(n-n)$
  $T(n) = c^n T(0)$
  $T(n) = c^n \cdot K$
  $T(n) = Kc^n$
  Logo, $T(n)$ é exponencial, ou ainda, $T(n) = O(c^n)$
  Aplicando o método da substituição para comprovar temos:
  Hipótese Indutiva (Valido para $T(k)$)
  $T(k) = O(c^k)$
  $T(k) \leq ac^k \qquad \forall k < n$
  Passo de Indução (Valido para $T(n-1)$)
  $T(n) = cT(n-1) \leq c \cdot a \cdot c^{n-1} = a \cdot c^n$
  $T(n) \leq ac^n$
  Dessa forma, qualquer valor de $a > 0$ e $c > 1$ resolve o problema.

- $T(n) = T(n-1) + n$ é $O(n^2)$
  Assumindo que a complexidade é quadrática temos:
  Hipótese Indutiva (Valido para $T(n)$)
  $T(n) = O(n^2)$
  $T(n) \leq cn^2$
  Passo de Indução (Valido para $T(n-1)$)
  $T(n) \leq c(n-1)^2 + n$
  $T(n) \leq c(n^2 - 2n + 1) + n$
  $T(n) \leq cn^2 - 2cn + c + n$
  $T(n) \leq cn^2 - (2cn - c - n)$

Devemos ter $(2cn - c - n) \geq 0$ para que seja valida a essa complexidade.
Logo, $c(2n-1) - n \geq 0$ então $c \geq \frac{n}{2n-1}$
Dessa forma, quando o valor de n tende ao infinito então $c \geq \frac{1}{2}$.

- $T(n) = T(\lceil n/2 \rceil) + 1$ é $O(lg\ n)$
  Assumindo que a complexidade é logarítmica temos:
  Hipótese Indutiva (Valido para $T(n)$)
  $T(n) = O(lg\ n)$
  $T(n) \leq c \cdot lg\ n$
  Passo de Indução (Valido para $T(n/2)$)
  $T(n) \leq c \cdot lg\ (n/2) + 1$
  $T(n) \leq c \cdot (lg\ n - lg 2) + 1$
  $T(n) \leq c \cdot (lg\ n - 1) + 1$
  $T(n) \leq c \cdot lg\ n - c + 1$
  $T(n) \leq c \cdot lg\ n - (c - 1)$
  $(c-1)$ é o resíduo
  Dessa forma, $c - 1 \geq 0$ então $c \geq 1$.

- $T(n) = 2T(\lfloor n/2 \rfloor) + n$ é $\Theta(n\ lg\ n)$
  Assumindo que a complexidade é $\Theta(n\ lg\ n)$ temos:
  Hipótese Indutiva (Valido para $T(n)$)
  $T(n) = \Theta(n\ lg\ n)$
  Dessa forma temos duas condições a serem obedecidas:
  $T(n) \geq c_1 n\ lg\ n$
  $T(n) \leq c_2 n\ lg\ n$
  Resolvendo a primeira condição
  $T(n) \geq c_1 n\ lg\ n$
  Passo de Indução (Valido para $T(n/2)$)
  $T(n) \geq c_1 \cdot 2 \cdot \frac{n}{2} \cdot lg \frac{n}{2} + n$
  $T(n) \geq c_1 \cdot n(lg\ n - lg\ 2) + n$
  $T(n) \geq c_1 \cdot n(lg\ n - 1) + n$
  $T(n) \geq c_1 \cdot n \cdot lg\ n - c_1 n + n$
  $T(n) \geq c_1 \cdot n \cdot lg\ n + (-c_1 n + n)$
  $T(n) \geq c_1 \cdot n \cdot lg\ n$
  Devemos ter que $(-c_1 n + n) \geq 0$ então $c_1 \leq 1$
  Dessa forma para qualquer valor de $c_1 \leq 1$ garantimos essa primeira condição.
  Resolvendo a segunda condição
  $T(n) \leq c_2 n\ lg\ n$
  Passo de Indução (Valido para $T(n/2)$)
  $T(n) \leq c_2 \cdot 2 \cdot \frac{n}{2} \cdot lg \frac{n}{2} + n$
  $T(n) \leq c_2 \cdot n(lg\ n - lg\ 2) + n$
  $T(n) \leq c_2 \cdot n(lg\ n - 1) + n$
  $T(n) \leq c_2 \cdot n \cdot lg\ n - c_2 n + n$
  $T(n) \leq c_2 \cdot n \cdot lg\ n - (c_2 n - n)$

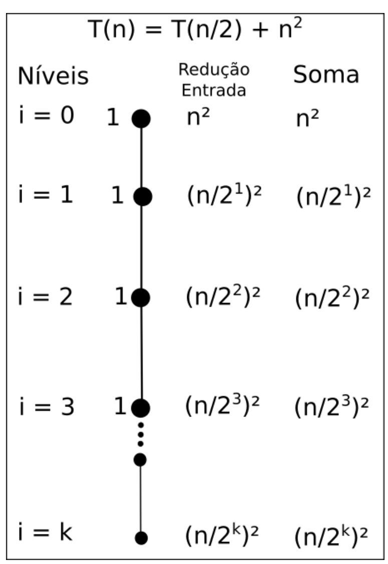
$T(n) \leq c_2 \cdot n \cdot lg\ n$
Devemos ter que $c_2 n - n \geq 0$ então $c_2 \geq 1$
Dessa forma para qualquer valor de $c_2 \geq 1$ garantimos a segunda condição.
Logo, $T(n)$ possui complexidade $\Theta(n\ lg\ n)$.

- $T(n) = T(\lfloor n/2 \rfloor + 17) + n$ é $O(n\ lg\ n)$. Assumindo que a complexidade é $O(n\ lg\ n)$ temos:
  $T(n) = O(n\ lg\ n)$
  Hipótese Indutiva (Valido para $T(k)$)
  $T(k) \leq c \cdot (k-a)\ lg\ (k-a) \qquad \forall k < n$
  Passo de Indução
  $T(n) \geq c \cdot (\frac{n}{2} + 17 - a) \cdot lg(\frac{n}{2} + 17 - a) + n$
  $T(n) \geq c \cdot \frac{(n+34-2a)}{2} \cdot lg(\frac{(n+34-2a)}{2}) + n$
  $T(n) \geq c \cdot (n + 34 - 2a) \cdot (lg(n + 34 - 2a) - lg2) + n$
  $T(n) \geq c \cdot (n + 34 - 2a) \cdot (lg(n + 34 - 2a) - 1) + n$
  $T(n) \geq c \cdot (n + 34 - 2a) \cdot lg(n + 34 - 2a) - c \cdot (n + 34 - 2a) + n$
  $T(n) \geq c \cdot (n + 34 - 2a) \cdot lg(n + 34 - 2a) - [c \cdot (n + 34 - 2a) - n]$
  $T(n) \geq c \cdot (n - a) \cdot lg(n - a)$

  Devemos ter que $c \cdot (n + 34 - 2a) - n \geq 0$ então $c \geq 1$ e $2a - 34 \geq 0 (a \geq 17)$

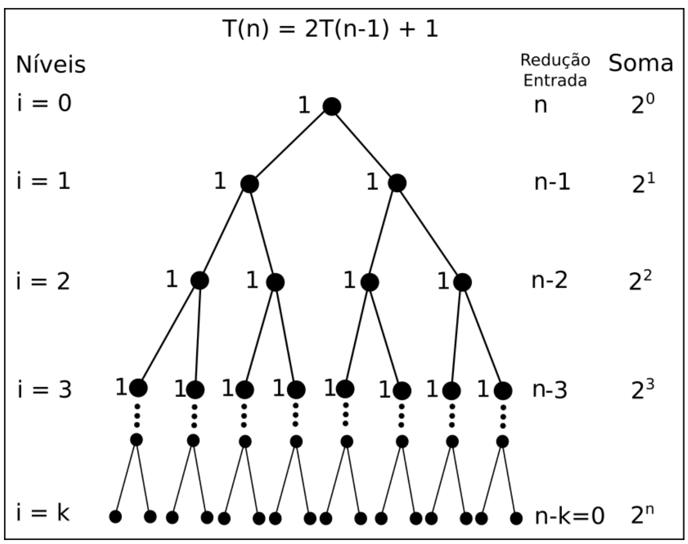
Resposta pergunta 3:

- $T(n) = T(n/2) + n^2$
  A figura abaixo representa a árvore de recursão para esse problema. Repare que devido ao fator 1 que multiplica a chamada recursiva $1 \cdot T(n/2)$ então essa árvore é na verdade um grafo linear, ou seja, não tem ramificações.



Resolvendo pelo método da árvore de recursão temos:
Somando todos os custos para a árvore inteira temos:
$T(n) = n^2 + (n/2)^2 + (n/2^2)^2 + \cdots + (n/2^k)^2$
Colocando o n em evidência temos:
$T(n) = n^2(1 + 1/2 + 1/2^2 + \cdots + 1/2^k)$
$T(n) = n^2(\sum_{i=0}^{lg\ n} (\frac{1}{2^i})^2) = n^2(\sum_{i=0}^{lg\ n} \frac{1}{2^{2i}})$
Aplicando a expressão para a soma dos termos de uma PG temos:
$T(n) = n^2(\frac{\frac{1}{4}^{lg\ n+1} - 1}{\frac{1}{4} - 1})$
$T(n) = n^2(\frac{\frac{1}{4} \cdot \frac{1}{4}^{lg\ n} - 1}{\frac{-3}{4}})$
$T(n) = n^2(\frac{-4}{3} \cdot (\frac{1}{4} \cdot n^{lg\frac{1}{4}} - 1))$
$T(n) = n^2(\frac{-4}{3} \cdot (\frac{1}{4} \cdot n^{-2} - 1))$
$T(n) = \frac{-4}{3}n^2 \cdot (\frac{1}{4} \cdot n^{-2} - 1)$
$T(n) = \frac{-4}{3}n^2 \cdot (\frac{1-4n^2}{4n^2})$
$T(n) = \frac{4}{3}n^2 - \frac{1}{3}$
Logo, a complexidade dessa expressão é quadrática.

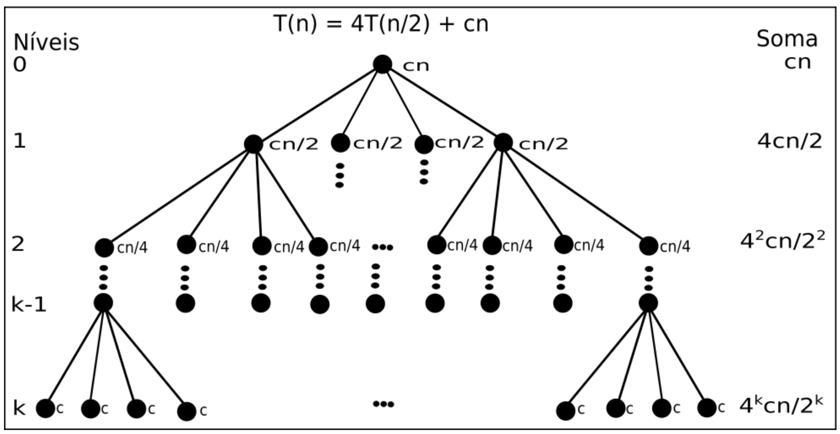
- $T(n) = 2T(n-1) + 1$
  A figura abaixo mostra a árvore de recursão para esse exercício.



Resolvendo pelo método da árvore de recursão temos:
Somando todos os custos para a árvore inteira temos:
$T(n) = 2^0 + 2^1 + 2^2 + \cdots + 2^n$
Aplicando expressão para somas dos termos de uma PG temos:
$T(n) = 2^{n+1} - 1$
Dessa forma, essa expressão é uma função exponencial.
Provando por substituição temos:
$T(n) = O(2^n)$
Hipótese de indução:

$T(k) \leq c \cdot 2^k - d \leq c2^k \qquad \forall k < n$
$T(n) = 2T(n-1) + 1$
$T(n) \leq 2(c \cdot 2^{n-1} - d) + 1$
$T(n) \leq c \cdot 2^n - 2d + 1$
$T(n) \leq c \cdot 2^n - [2d - 1]$
Dessa forma, $2d - 1 \geq 0$, logo $d \geq 1/2$
$T(n) \leq c \cdot 2^n$
Logo, a complexidade da árvore dada é exponencial.

- $T(n) = 4T(n/2) + cn$
  A figura abaixo mostra a árvore de recursão para esse exercício.



Calculando o tamanho do problema:
$n/2^k = 1$
$k = ln\ n$
Calculando o custo dos nós folhas e fazendo $T(1) = 1$:
$4^k \cdot T(1) = 4^{ln\ n} \cdot 1 = n^{ln\ 4} = n^2$
Calculando o custo total temos:
$T(n) = \sum_{i=0}^{k-1} 2^i cn + cn^2$
$T(n) = cn \cdot \sum_{i=0}^{lg\ n-1} 2^i + n^2$
Calculando a soma da PG temos:
$T(n) = cn \cdot (2^{lg\ n} - 1) + n^2$
$T(n) = cn \cdot (n-1) + n^2$
$T(n) = cn^2 - cn + n^2$
Logo, $T(n)$ é quadrática.
Provando por substituição temos:
$T(n) = O(n^2)$
$T(k) \leq ck^2 - d \cdot k \leq ck^2$ com $d \geq 0$
Para $k = n$ temos
$T(n) = 4T(n/2) + \bar{c}n$
$T(n) \leq 4(cn^2/4 - dn/2) + \bar{c}n$
$T(n) \leq cn^2 - 2dn + \bar{c}n$

$T(n) \leq cn^2 - [2d - \overline{c}]n$
Dessa forma, $d \geq \overline{c}/2$ então:
$T(n) \leq cn^2$
Logo, $T(n)$ possui complexidade quadrática.

Resposta pergunta 4:
Seja T(n) a função que calcula a complexidade de Pesquisa(n).
Os custos das linhas 3 e 4 são $O(1)$. Dessa forma, $T(1) = 2$
A linha 7 é a que possui a chamada recursiva e é a linha mais importante de ser analisada.
A expressão dessa função recursiva é:
$T(n) = n + T(\frac{3}{5}n)$
$T(n) = T(\frac{3}{5}n) + n$
Expandindo essa recursão temos:
$T(n) = T(\frac{3^2}{5^2}n) + \frac{3}{5}n + n$
$T(n) = T(\frac{3^3}{5^3}n) + \frac{3^2}{5^2}n + \frac{3}{5}n + n$
$\ldots$
$T(n) = T(\frac{3^k}{5^k}n) + \frac{3^{k-1}}{5^{k-1}}n + \cdots + \frac{3^2}{5^2}n + \frac{3}{5}n + n$
Assumindo que n é da forma $n = \frac{5^k}{3^k}$, logo, $k = \log_{\frac{5}{3}} n$
$T(n) = T(1) + \frac{3^{k-1}}{5^{k-1}}n + \cdots + \frac{3^2}{5^2}n + \frac{3}{5}n + n$
$T(n) = 2 + \frac{3^{k-1}}{5^{k-1}}n + \cdots + \frac{3^2}{5^2}n + \frac{3}{5}n + n$
$T(n) = 2 + n \cdot \sum_{i=0}^{k-1} (\frac{3}{5})^i$
Calculando o valor de $\sum_{i=0}^{k-1} (\frac{3}{5})^i$ temos:
$\sum_{i=0}^{k-1} (\frac{3}{5})^i = \frac{(\frac{3}{5})^k - 1}{\frac{3}{5} - 1} = -\frac{5}{2} \cdot (\frac{3}{5}^{\log_{\frac{5}{3}} n} - 1)$
Simplificando temos:
$\frac{3}{5}^{\log_{\frac{5}{3}} n} = n^{\log_{\frac{5}{3}} 3/5} = n^{-1}$
Então:
$-\frac{5}{2} \cdot (n^{-1} - 1) = \frac{-5}{2} \cdot (\frac{1-n}{n}) = \frac{5n-5}{2n} = d$
Quando n tende a infinito a expressão acima tende a $d = \frac{5}{2}$
Sendo assim:
$T(n) = 2 + n \cdot \frac{5}{2}$
Logo, a complexidade da função Pesquisa(n) é linear $O(n)$.
Aplicando o método da substituição para comprovar temos:
Hipótese Indutiva (Valido para $T(n)$)
$T(n) = O(n)$
$T(n) \leq cn$
Passo de Indução (Valido para $T(\frac{3}{5}n)$)
$T(n) \leq c \cdot \frac{3}{5} \cdot n + n$
$T(n) \leq cn(\frac{5}{5} - \frac{2}{5}) + n$
$T(n) \leq cn\frac{5}{5} - cn\frac{2}{5} + n$
$T(n) \leq cn - cn\frac{2}{5} + n$
Devemos ter que $-cn\frac{2}{5} + n \leq 0$, então temos $c \leq \frac{1}{2}$
Dessa forma, para qualquer $c \leq \frac{1}{2}$ resolve o problema.

Resposta pergunta 5:

```
1 void Cubo(int n)
2 {
3     if (n = 1){
4         return 1;
5     }else{
6         return Cubo(n-1) + n*n*n;
7     }
8 }
```

Seja T(n) a função que calcula a complexidade de Cubo(n).
Os custos das linhas 3 e 4 são O(1)
A linha 6 é a que possui a chamada recursiva e é a linha mais importante de ser analisada.
Devemos responder a pergunta: quantas vezes a linha 6 é executada?
Para isso nesse problema temos: 1 comparação, 1 subtração, 1 soma, 2 multiplicações e 1 retorno. Totalizando 6 operações.
Veja que a cada chamada da função cubo pelo menos 6 operações são realizadas. Dessa forma temos:
$T(n) = T(n-1) + 6$
Desenvolvendo a recursão temos:
$T(n) = T(n-1) + 6$
$T(n) = (T(n-2) + 6) + 6$
$T(n) = ((T(n-3) + 6) + 6) + 6$
$\ldots$
$T(n) = T(n-k) + 6k$
Quando a recursão termina? Quando $n - k = 1$. Dessa forma, consideramos $k = n - 1$.
$T(n) = T(n-(n-1)) + 6(n-1)$
$T(n) = T(1) + 6(n-1)$
temos que $T(1) = 2$, pela contagem das linhas 3 e 4.
$T(n) = 2 + 6n - 6$
$T(n) = 6n - 4$
Dessa forma, podemos dizer que Cubo(n) é uma função linear, ou ainda, $T(n) = O(n)$.
Aplicando o método da substituição para comprovar temos:
Hipótese Indutiva (Valido para $T(n)$)
$T(n) = O(n)$
$T(n) \geq cn$
Passo de Indução (Valido para $T(n-1)$)
$T(n) \geq c \cdot n - 1 + 6$
$T(n) \geq cn - c + 6$
Fazendo $-c + 6 \leq 0$ então $c \geq 6$ temos:
Dessa forma, para qualquer $c \geq 6$ resolve o problema.

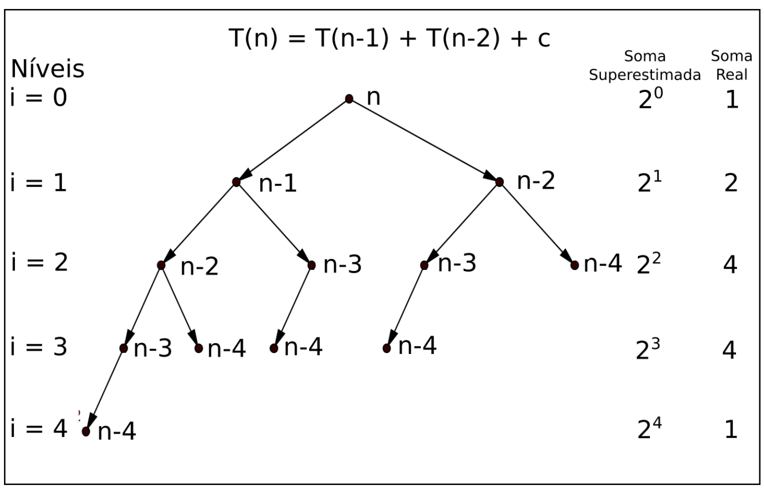
Resposta pergunta 7:

Código para o calculo do número de Fibonacci iterativo e recursivo.

```
int fibIterativo(int n){
    int fib1 = 1;
    int fib2 = 1;
    for (int i = 0; i < n-1; i++){
        int fibn = fib1 + fib2;
        fib1 = fib2;
        fib2 = fibn;
    }
    return fib1;
}

int fibRecursivo(int n){
    if (n == 1){
        return 1;
    }else if (n == 2){
        return 1;
    }else{
        return fibRecursivo(n-1)+fibRecursivo(n-2);
    }
}
```

A figura abaixo descreve um pouco da árvore de recursão para essa chamada recursiva. Repare que dois custos foram feitos um superestimando a quantidade de nós por nível e outro com o custo real para esse exemplo.



- Análise de complexidade da versão recursiva.
  A expressão que descreve as chamadas recursivas de fibonacci é:
  $T(n) = T(n-1) + T(n-2) + k$
  De acordo com a super-estimativa da árvore de recursão podemos pensar que a complexidade é $O(2^n)$
  Dessa forma, temos:
  $T(n) = T(n-1) + T(n-2) + k$
  $T(n) = O(2^n)$
  $T(n) \leq c2^n$
  $T(n) \leq c2^{n-1} + c2^{n-2} + k$

$T(n) \leq \frac{c}{2}2^n + \frac{c}{4}c2^n + k$
$T(n) \leq \frac{2c2^n+1c2^n+4k}{4}$
$T(n) \leq \frac{3c2^n+4k}{4}$
$T(n) \leq \frac{4c2^n-1c2^n+4k}{4}$
$T(n) \leq c2^n + \frac{4k-1c2^n}{4}$
Dessa forma, devemos ter: $\frac{4k-1c2^n}{4} \leq 0$ então $c \geq 4k/2^n$.
Fazendo n tender ao infinito e k sendo uma constante temos que $c \geq 0$.
Nesse sentido, $T(n)$ é $O(2^n)$.
Observação: Nesse exercício fizemos uma super-estimativa da complexidade. A complexidade real de fibonacci recursivo é $O(\phi^n)$.