Ano Lectivo de 2007/2008

Frequência

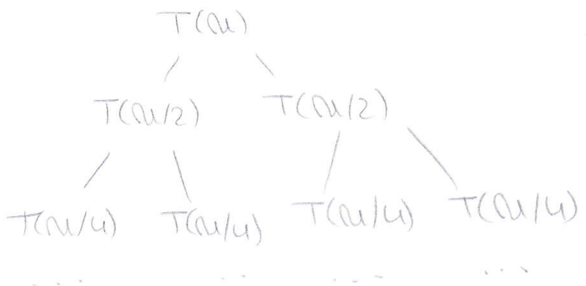
3. Considere a seguinte função recursiva:

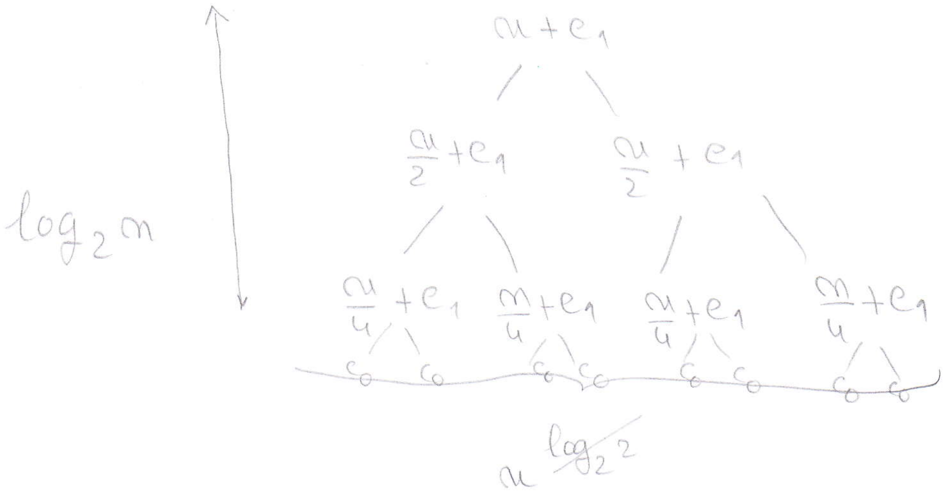
```
void exemplo(int a[], int n) {
   int x = n/2;
   if (n>0) {
      exemplo(a,x);
      processa(a,n);
      exemplo(a+x,n-x);
   }
}
```

Sabendo que $T_{processa}(n) = \Theta(n)$, escreva uma recorrência que descreva o comportamento temporal da função **exemplo** e indique, justificando, a solução dessa recorrência.

$$T(n) = \begin{cases} c_0 & \text{se } n=0 \\ c_1 + T(n/2) + T(n-n/2) + T_{processa}(n) & \text{se } n>0 \end{cases}$$

$$T(n) = \begin{cases} c_0 & \text{se } n=0 \\ c_1 + 2T(n/2) + \Theta(n) & \text{se } n>0 \end{cases}$$





$n + c_1$

$2\frac{n}{2} + c_1$

$4\frac{n}{4} + c_1$

$2^i \frac{n}{2^i} + c_1$

$n^{\log_2 2}$

$$T(n) = \sum_{i=0}^{\log_2 n - 1} (n + c_1) + cn$$

$$T(n) = \sum_{i=0}^{\log_2 n - 1} n + \sum_{i=0}^{\log_2 n - 1} c_1 + cn$$

$$T(n) = n \sum_{i=0}^{\log_2 n - 1} 1 + c_1 \sum_{i=0}^{\log_2 n - 1} 1 + c_0 n$$

$$T(n) = n \times \log_2 n + c_1 \log_2 n + c_0 n$$

$$T(n) = \Theta(n \log_2 n)$$

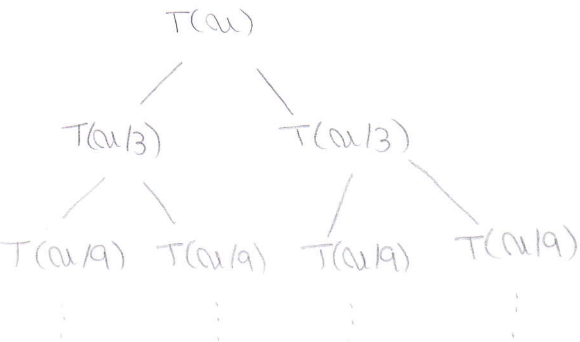
Ano Lectivo de 2006/2007

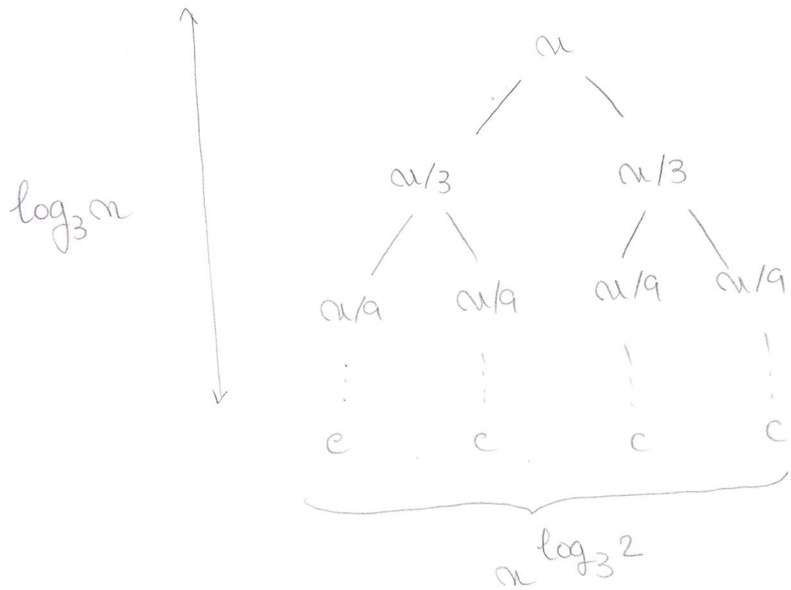
Exame da Época Especial

3. Considere a seguinte relação de recorrência que traduz a complexidade de um determinado algoritmo, em função do tamanho N do *input*.

$$T(N) = \begin{cases} c & \text{se} \quad N < 2 \\ N + 2.T(N/3) & \text{se} \quad N > 1 \end{cases}$$

Apresente, justificando, uma expressão que descreva o comportamento assimptótico desse algoritmo.





n

$2(n/3)$

$4(n/9)$

$2^i(n/3^i) = \left(\frac{2}{3}\right)^i n$

$$T(n) = \sum_{i=0}^{\log_3 n} \left(\frac{2}{3}\right)^i n + c n^{\log_3 2}$$

$$T(n) = n \sum_{i=0}^{\log_3 n} \left(\frac{2}{3}\right)^i + c n^{\log_3 2}$$

$$T(n) = n \left(\frac{2/3^{\log_3 n} - 1}{2/3 - 1} \right) + c n^{\log_3 2}$$

$$T(n) = n \left(\frac{n^{\log_3 2/3} - 1}{-1/3} \right) + c n^{\log_3 2}$$

$$T(n) = -3n\left(n^{\log_3 2/3} - 1\right) + c\,n^{\log_3 2}$$

$$T(n) = -3\left(n^{\log_3(2/3)+1} - n\right) + c\,n^{\log_3 2}$$

$$T(n) = -3n^{\log_3(2/3)+1} + 3n + c\,n^{\log_3 2}$$

$$T(n) = \Theta(n)$$

Ano Lectivo de 2006/2007

Exame da Época de Recurso

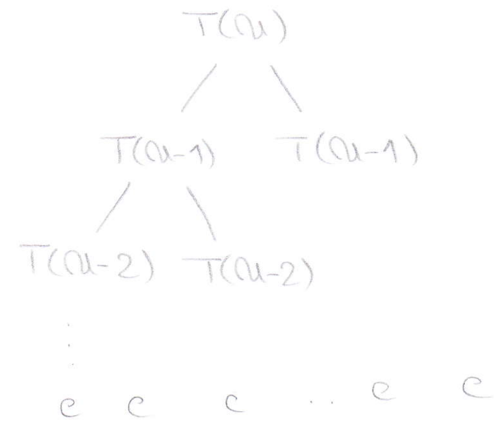
5. Considere o seguinte função para o problema das torres de Hanoi:

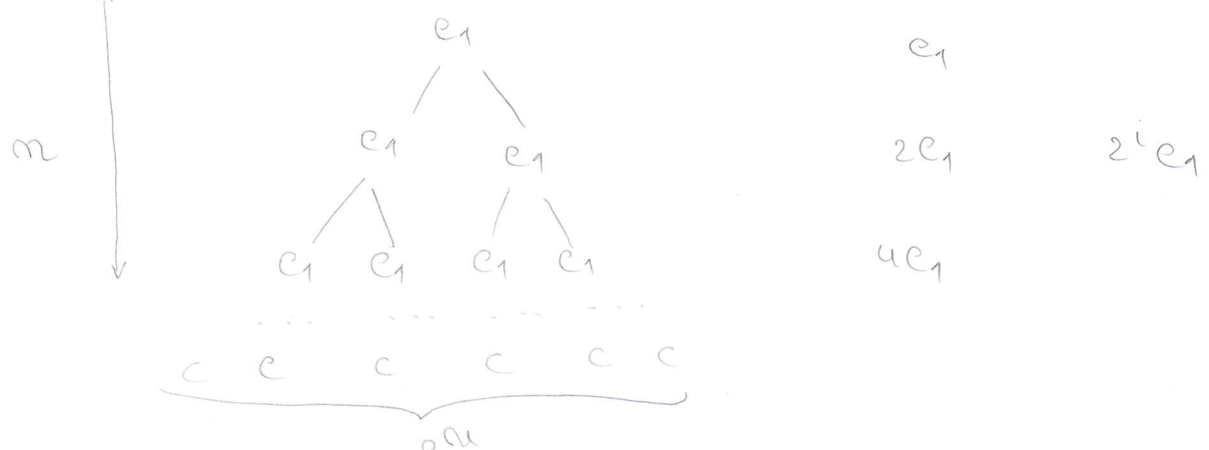
```
int Hanoi(int N, int esq, int dir, int meio) {
  if (N > 0) {
     Hanoi(N-1, esq, meio, dir);
     printf("Mover disco de %d para %d\n", esq, dir);
     Hanoi(N-1, meio, dir, esq);
  }
}
```

Escreva uma recorrência que descreva o comportamento temporal da função `Hanoi` e indique, justificando, a solução dessa recorrência.

$$T(n) = \begin{cases} c \\ c_1 + T(n-1) + T(n-1) \end{cases}$$

$$T(n) = \begin{cases} c \\ c_1 + 2T(n-1) \end{cases}$$





$$T(n) = \sum_{i=0}^{n} 2^i c_1 + c \times 2^n = c_1 \sum_{i=0}^{n} 2^i + c\, 2^n =$$

$$T(n) = c_1 \times \frac{2^n - 1}{2 - 1} + c_2 n = c_1 2^n - c_1 + c_2 n = k_0 2^n + k_1$$

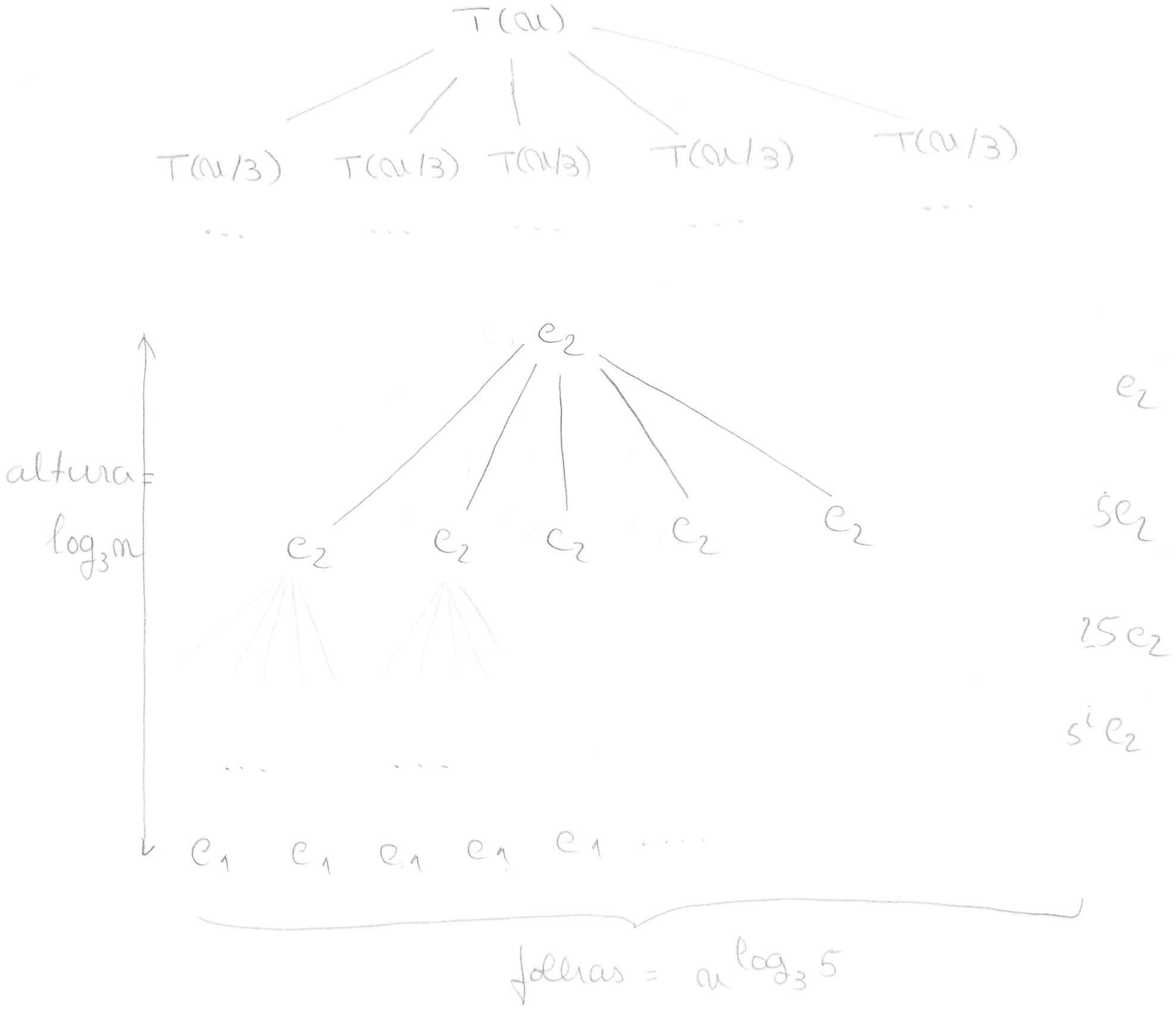
$$T(n) = \Theta(2^n)$$

Ano Lectivo de 2006/2007

Exame da 2ª Chamada

5. Indique, justificando, a solução da seguinte recorrência:

$$T(n) = \begin{cases} c_1 & \text{se } n \leq 1 \\ c_2 + 5\,T(n/3) & \text{se } n > 1 \end{cases}$$



$$T(n) = \sum_{i=0}^{\log_3 n - 1} 5^i c_2 + c_1 n^{\log_3 5}$$

$$T(n) = c_2 \sum_{i=0}^{\log_3 n - 1} 5^i + c_1 n^{\log_3 5} \iff T(n) = c_2 \times \frac{5^{\log_3 n} - 1}{5 - 1} + c_1 n^{\log_3 5}$$

$$T(n) = \frac{c_2}{4} 5^{\log_3 n} - \frac{c_2}{4} + c_1 n^{\log_3 5} \iff T(n) = \frac{c_2}{4} n^{\log_3 5} - \frac{c_2}{4} + c_1 n^{\log_3 5}$$

$$T(n) = \Theta(n^{\log_3 5})$$

Nota: $\log_3 5 = \omega$

$3^{\omega} = 5$

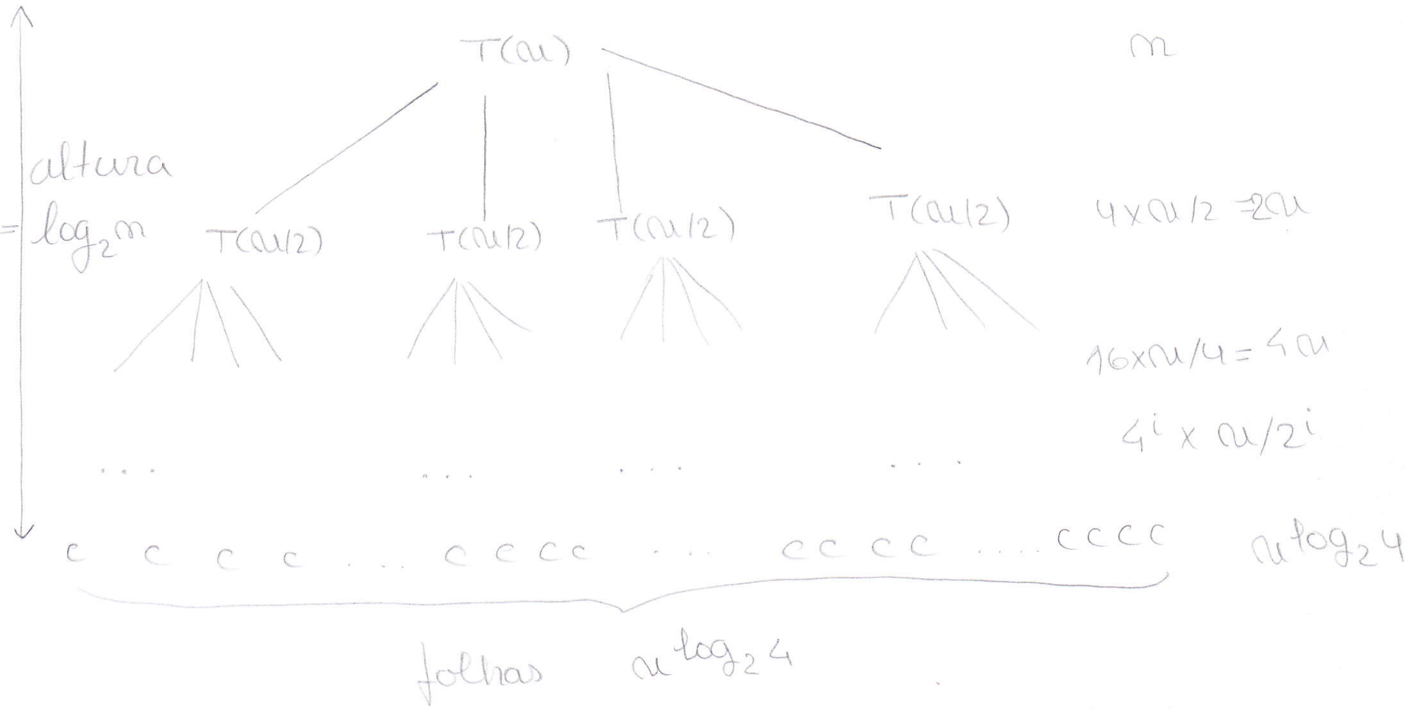
$1 < \omega < 2$

Ano Lectivo de 2006/2007

Exame da 1ª Chamada

5. Indique, justificando, a solução da seguinte recorrência:

$$T(n) = \begin{cases} c & \text{se } n \leq 1 \\ 4\,T(n/2) & \text{se } n > 1 \end{cases}$$



$$T(n) = \sum_{i=0}^{\log_2 n - 1} 2^i \times 0 + n^{\log_2 4} = n^2$$

$$T(n) = \Theta(n^2)$$

Ano Lectivo de 2007/2008

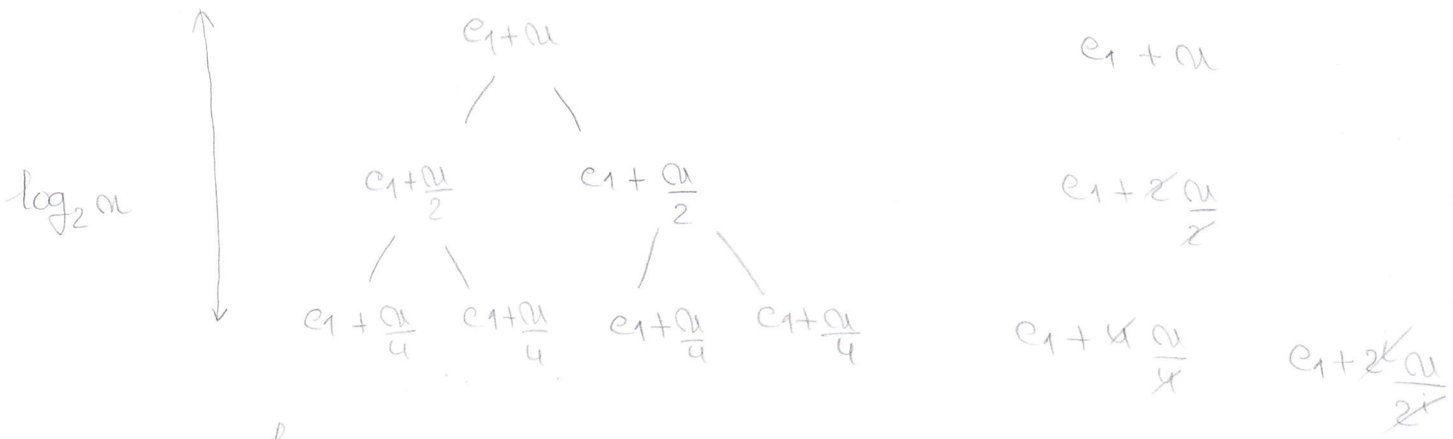
Exame da Época de Recurso

6. Analise a complexidade da seguinte função que calcula os factores de balanço de uma árvore. Assuma que árvore está balanceada e que a função **altura** executa em tempo linear no tamanho da árvore de entrada.

```
void bals(Arvore a) {
   if (!a) return;
   a->bal = altura(a->dir) - altura(a->esq);
   bals(a->esq);
   bals(a->dir);
}
```

$$T(n) = \begin{cases} c_0 & \text{se } n = 1 \\ c_1 + T_{altura}(n/2) + T_{altura}(n/2) + T(n/2) + T(n/2) \end{cases}$$

$$T(n) = \begin{cases} c_0 \\ c_1 + T_{altura}(n) + 2T(n/2) \end{cases}$$



$$T(n) = \sum_{i=0}^{\log_2 n - 1} (c_1 + n) + cn$$

$$T(n) = (c_1 + n) \sum_{i=0}^{\log_2 n - 1} 1 + cn$$

$$T(n) = (c_1 + n) \log_2 n + cn$$

$$T(n) = c_1 \log_2 n + n \log_2 n + cn$$

$$T(n) = \Theta(n \log_2 n)$$

Ano Lectivo de 2005/2006

Exame da Época de Recurso

1. Implemente uma versão do algoritmo Merge Sort que, em vez de utilizar sub-listas de tamanho aproximadamente $N/2$, utiliza sub-listas de tamanho aproximadamente $N/3$. **Sugestão:** comece por implementar a função de combinação de sub-listas ordenadas

   ```
   void merge(int A[],int esq,int div1,int div2,int dir);
   ```

   em que `esq` e `dir` representam os limites da zona válida do *array* A e os sub-*arrays* a serem combinados estão nas regiões $[esq \ldots div_1]$, $[div_1 + 1 \ldots div_2]$ e $[div_2 + 1 \ldots dir]$.

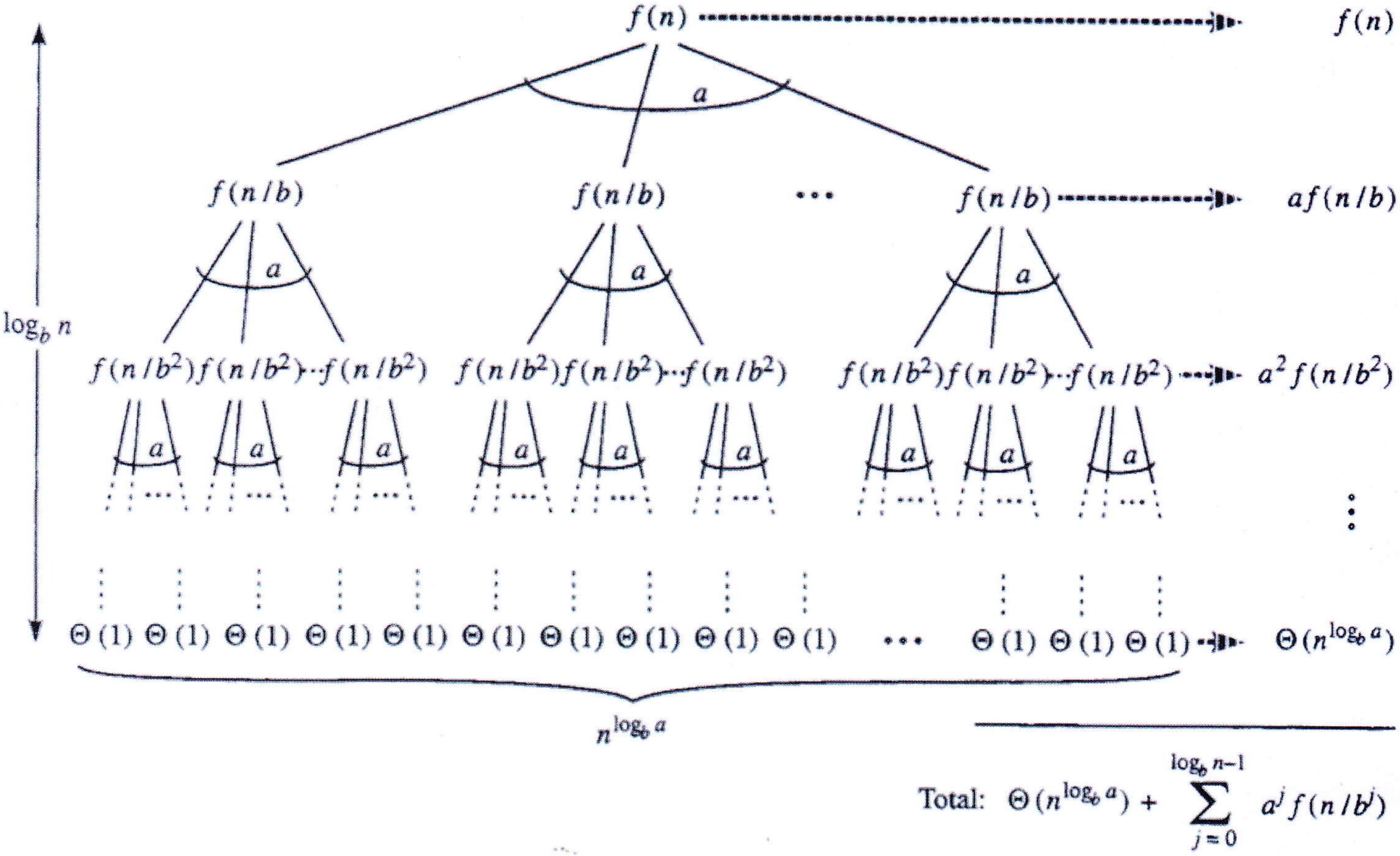
2. Escreva uma equação de recorrência que descreva o tempo de execução $T(N)$ do algoritmo anterior e desenhe a árvore correspondente para um caso em que a

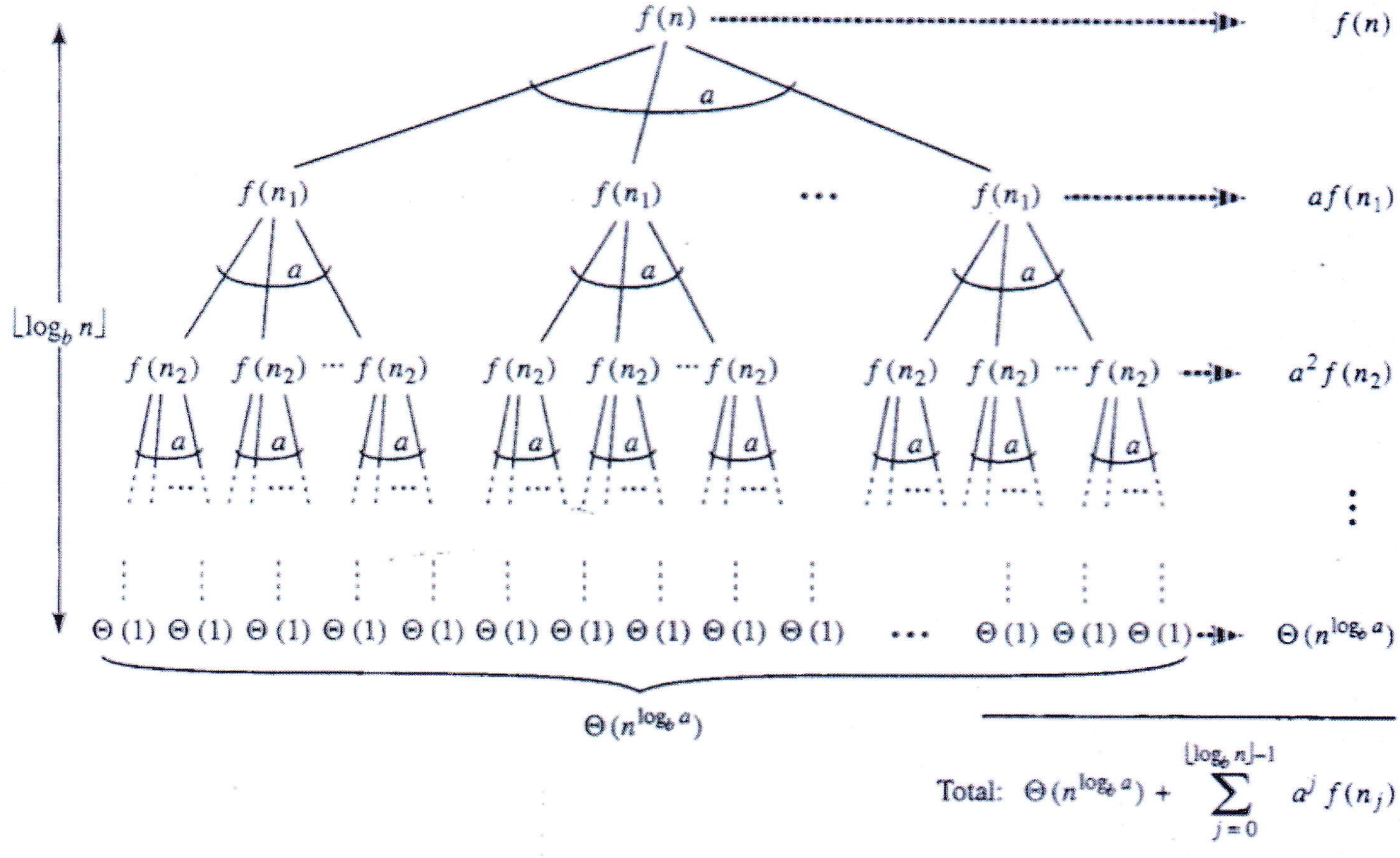
3. É possível exprimir o comportamento assimptótico de $T(N)$ na notação Θ? Justifique a sua resposta e, em caso afirmativo, apresente esse resultado.

Ano Lectivo de 2005/2006

Exame da 2ª Chamada

2. Assumindo que as operações de adição, multiplicação por 2 e divisão por 2 executam em tempo constante `c`, calcule o tempo de execução deste algoritmo como uma função do valor de `y`. Sugestão: Exprima o número de iterações do ciclo como uma relação de recorrência sobre o valor de `y`.


f(n)
a
f(n/b)
af(n/b)
log_b n
f(n/b^2)
a^2 f(n/b^2)
Θ(1)
Θ(n^{log_b a})
n^{log_b a}
Total: Θ(n^{log_b a}) + Σ_{j=0}^{log_b n−1} a^j f(n/b^j)


f(n)
a
f(n_1)
af(n_1)
⌊log_b n⌋
f(n_2)
a^2 f(n_2)
Θ(1)
Θ(n^{log_b a})
Total: Θ(n^{log_b a}) + Σ_{j=0}^{⌊log_b n⌋−1} a^j f(n_j)