**PRÁTICA 3 – Trabalho 2 Exercício 3: Torres de Hanói**

**Nome:** Davi Gabriel Domingues

**Número USP:** 15447497

O trabalho consiste na implementação em C++ de uma função que implemente o famigerado algoritmo de resolução das Torres de Hanói, a partir do princípio de códigos computacionalmente recursivos. Tem – se como procedimento/função padrão para esta atividade a função recHanoi().

O objetivo é ilustrar a aplicação do algoritmo recursivo em si, além de se verificar as implicações nas análises da complexidade de tempo e de uso de memória provocados pelo programa durante a sua execução.

Sendo assim, foram elaboradas as seguintes respostas apropriadas para cada um dos itens:  
  
a) O código, implementado em C++, do programa para resolução do problema das Torres de Hanói baseado no procedimento *recHanoi*( ) se encontra a seguir:

Texto

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente com confiança baixa

b) Tem – se, a seguir, uma série de valores de n, que representam o número de discos dispostos inicialmente nas Torres de Hanói, pequenos, a fim de se checar o quão correta está a solução disposta em C++:

**Para n = 2:**

**Texto

Descrição gerada automaticamente**

**Para n = 3:**  
  
Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Para n = 4:**

Texto

Descrição gerada automaticamente

**Para n = 5:**

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

**Teclado de computador

Descrição gerada automaticamente**

c) A análise experimental de tempo do programa se embasa no seguinte código:

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Com isso, gera – se o gráfico de n x tempo de execução a seguir:

Gráfico

Descrição gerada automaticamente

d) A análise experimental sobre o número de variáveis criadas pelo programa se dá pelo programa a seguir:

Texto

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

Dessa forma, é visível que o número de variáveis criadas fornece um bom indicativo para o uso da memória. Observa – se que, o padrão da resolução das Torres de Hanói é definido previamente pela equação 2^(n) – 1, sem a análise computacional.

Como cada função é chamada recursivamente, então cada uma é referenciada, segundo o algoritmo construído, 2\*2(n) vezes, ou seja, 2^(n + 1) vezes. Entretanto, na última vez em que a segunda referência à recHanoi() é feita, a recursão é interrompida e executada uma vez apenas e não duas, dado que o total de discos a serem reposicionados se tornou nulo, assim, o total de vezes, na verdade, é igual a 2^(n + 1) – 1.  
  
Considerando que a cada chamada foram criadas quatro variáveis, além de se considerar as quatro variáveis iniciais existentes na main(), tem – se que o total verdadeiro de variáveis criadas é:

4\*(2^(n + 1) – 1) + 4 🡪 2^(n + 3) – 4 + 4 🡪 2^(n + 3) variáveis

Logo, a função que retrata o total de variáveis criadas é f(n) = 2^(n + 3).

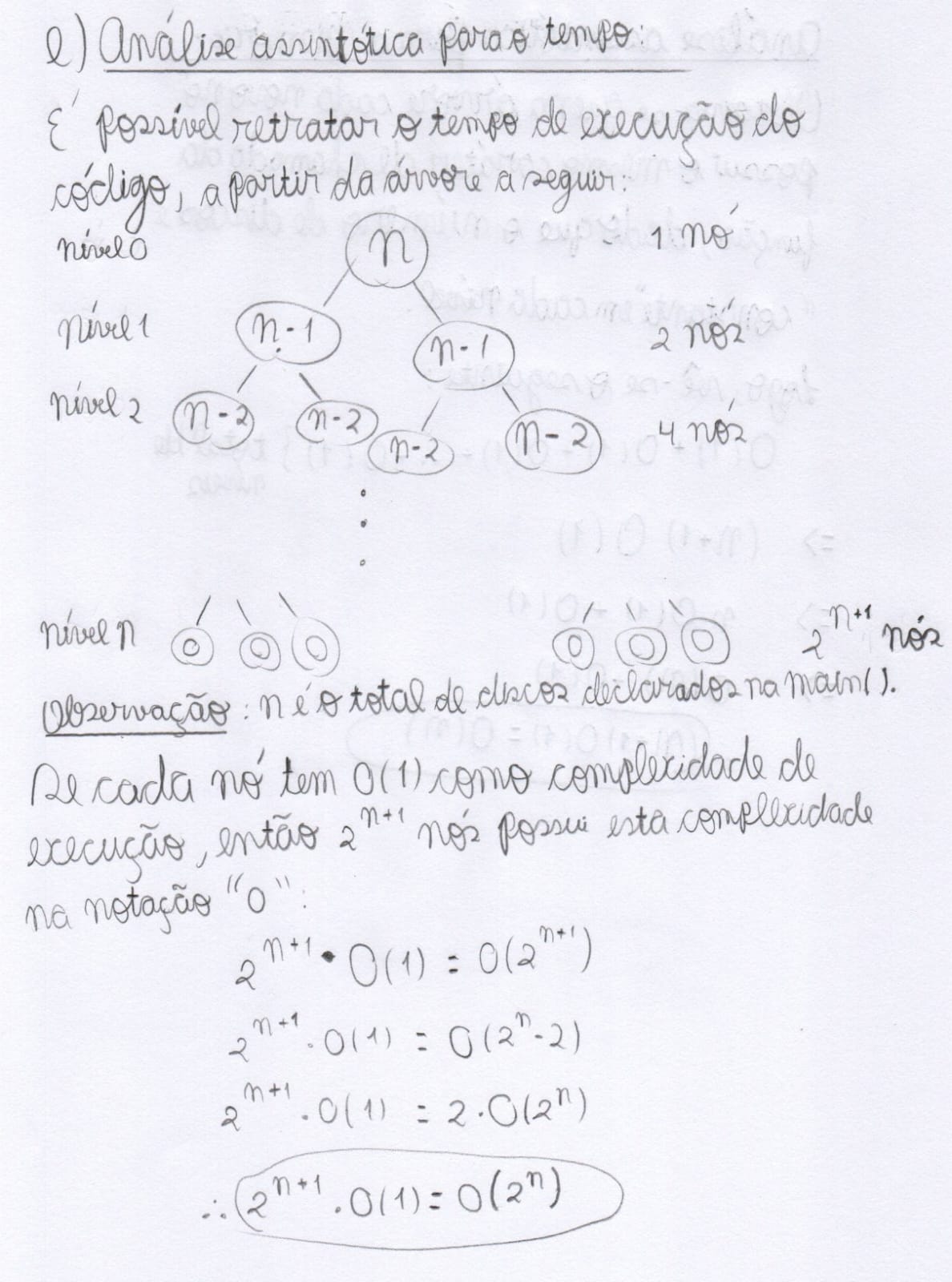
Nesse panorama, tem – se este gráfico de n x número de variáveis criadas associado:

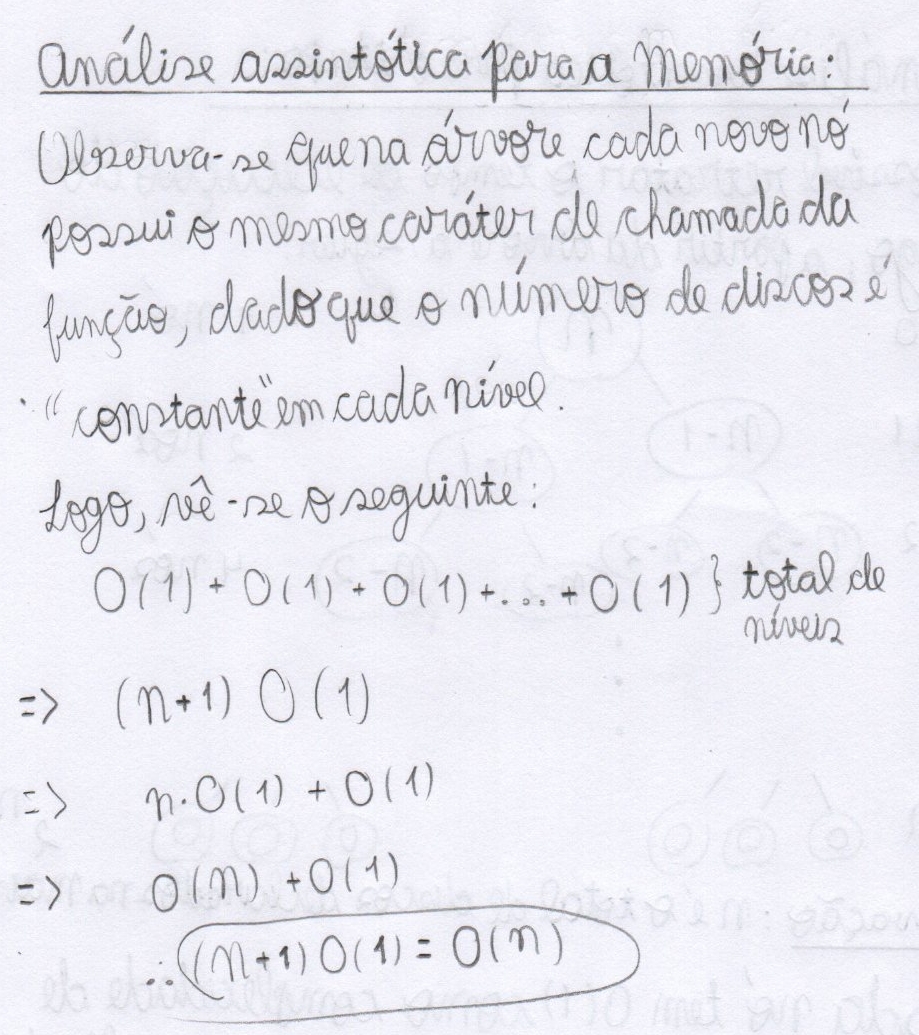
Gráfico, Gráfico de linhas

Descrição gerada automaticamente

Obs: O gráfico gerado em Excel está na escala logarítmica de base 2, por isso a aparência de uma reta. Se não utilizarmos essa escala, tal função seria, na escala usual, uma curva ascendente, semelhante ao que se vê no item “c)”.

e) A análise dos resultados experimentais obtidos nos itens “c)” e “d)” permite uma comparação com os resultados teóricos da análise assintótica para tempo e memória, sendo tal relação a seguinte:





Dessa forma, percebe – se que a complexidade de tempo bate com a descrição fornecida em “c)”, respaldada pelos dados obtidos e analisados nos devidos valores de n. Entretanto, vê – se que no caso da memória, há uma disparidade evidente: a quantidade de variáveis criadas segue uma complexidade O(2^n), diferentemente do gasto de memória durante a execução da função recHanoi(), o qual é de complexidade O(n), como demonstrado acima.  
  
Ou seja, há um gasto de memória linear, mas com um total de variáveis envolvidas que cresce exponencialmente, assim como o tempo de execução do algoritmo recursivo em questão.