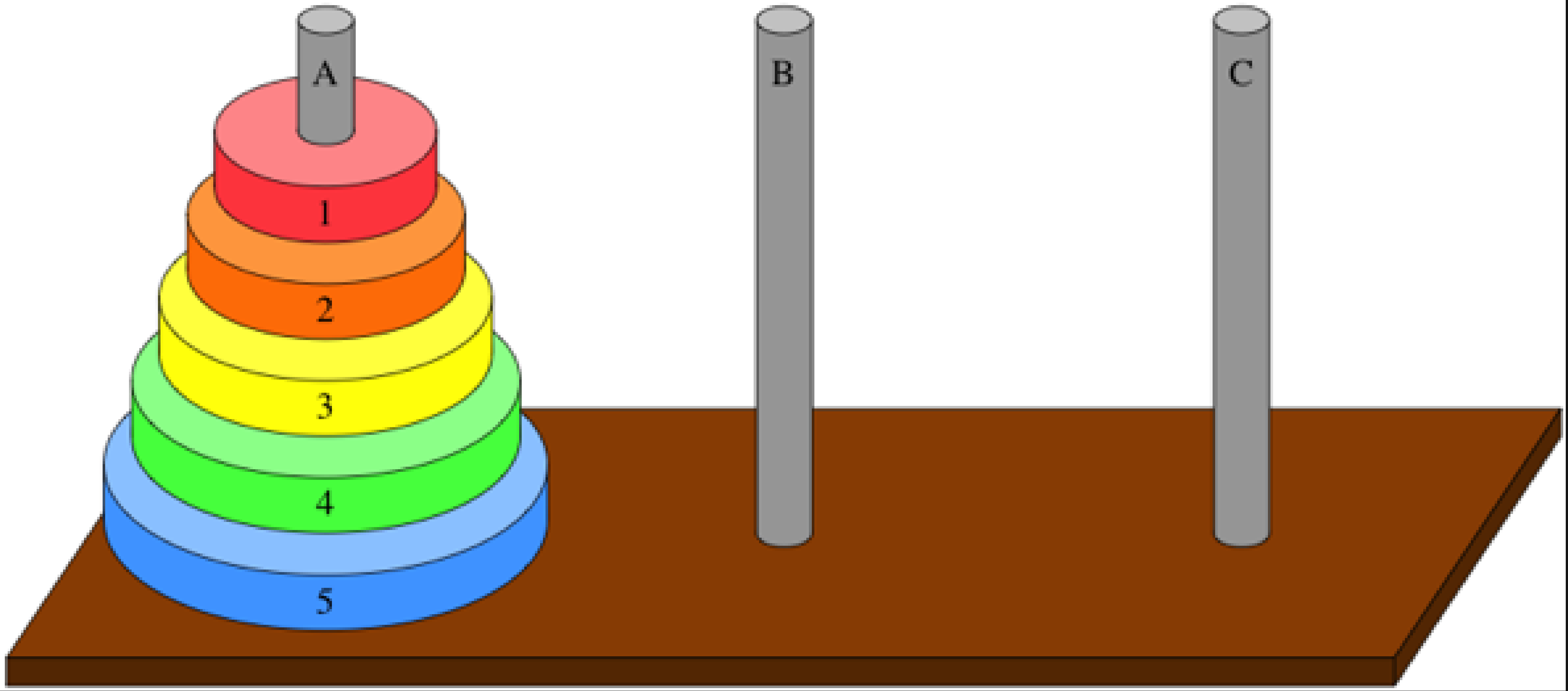
Algoritimo de Hanoi

KAIO GUILHERME
ANGELO FERRO

Problema de hanoi



Hanoi na Pratica

Codigo de Hanoi.C

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void hanoi(int n, char origem, char destino, char auxiliar) {
    if (n > 0) {
        hanoi(n - 1, origem, auxiliar, destino);
        //printf("Move o disco %d de %c para %c\n", n, origem, destino);
        hanoi(n - 1, auxiliar, destino, origem);
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Uso: %s <numero_de_discos>\n", argv[0]);
        return 1;
    }

    int n = atoi(argv[1]);
    if (n <= 0) {
        printf("Numero de discos invalido.\n");
        return 1;
    }

    clock_t inicio = clock();
    hanoi(n, 'A', 'C', 'B');
    clock_t fim = clock();
    double tempo = (double)(fim - inicio) / CLOCKS_PER_SEC;
    printf("Tempo de execucao: %.6f segundos\n", tempo);

    return 0;
}
```

Custo e Complexidade

Recorrência:

$$T(N) = \begin{cases} 0, & \text{se } N = 0 \\ 2T(N-1) + 1, & \text{se } N > 0 \end{cases}$$

Tentativa de aplicar o Teorema Mestre:

$$T(N) = a \cdot T\left(\frac{N}{b}\right) + f(N) \quad \text{(não se aplica, pois } b > 1\text{)}$$

Desenvolvendo a recorrência:

$$T(N-1) = 2\,[2T(N-1-1) + 1] + 1 = 2^2T(N-2) + 2 + 1$$

$$T(N-2) = 2^3\,[2T(N-2-1) + 1] + 2 + 1$$

Custo e Complexidade

Generalizando:

$$T(N) = 2^k T(N-k) + \sum_{i=0}^{k-1} 2^i$$

A fórmula recorrente generalizada era:

$$T(N) = 2^k T(N-k) + \sum_{i=0}^{k-1} 2^i$$

Assumindo $N = k$, temos:

$$T(k) = 2^k T(N-k) + \sum_{i=0}^{k-1} 2^i \Rightarrow T(k) = 2^k T(0) + \sum_{i=0}^{k-1} 2^i$$

Como $T(0) = 0$, então:

$$T(k) = \sum_{i=0}^{k-1} 2^i$$

Agora, substituímos $k = N$:

$$T(N) = \sum_{i=0}^{N-1} 2^i$$

Custo e Complexidade

Solução fechada:

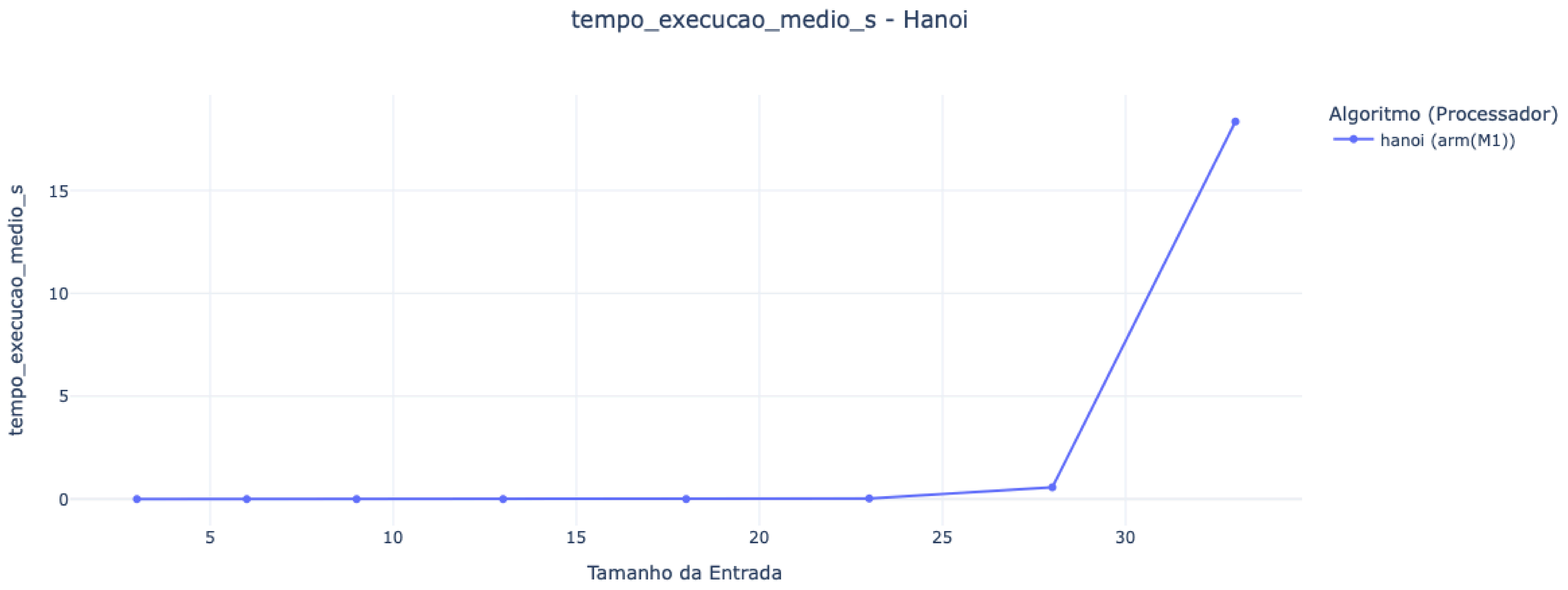
Sabemos que a soma geométrica:

$$\sum_{i=0}^{N-1} 2^i = 2^N - 1$$

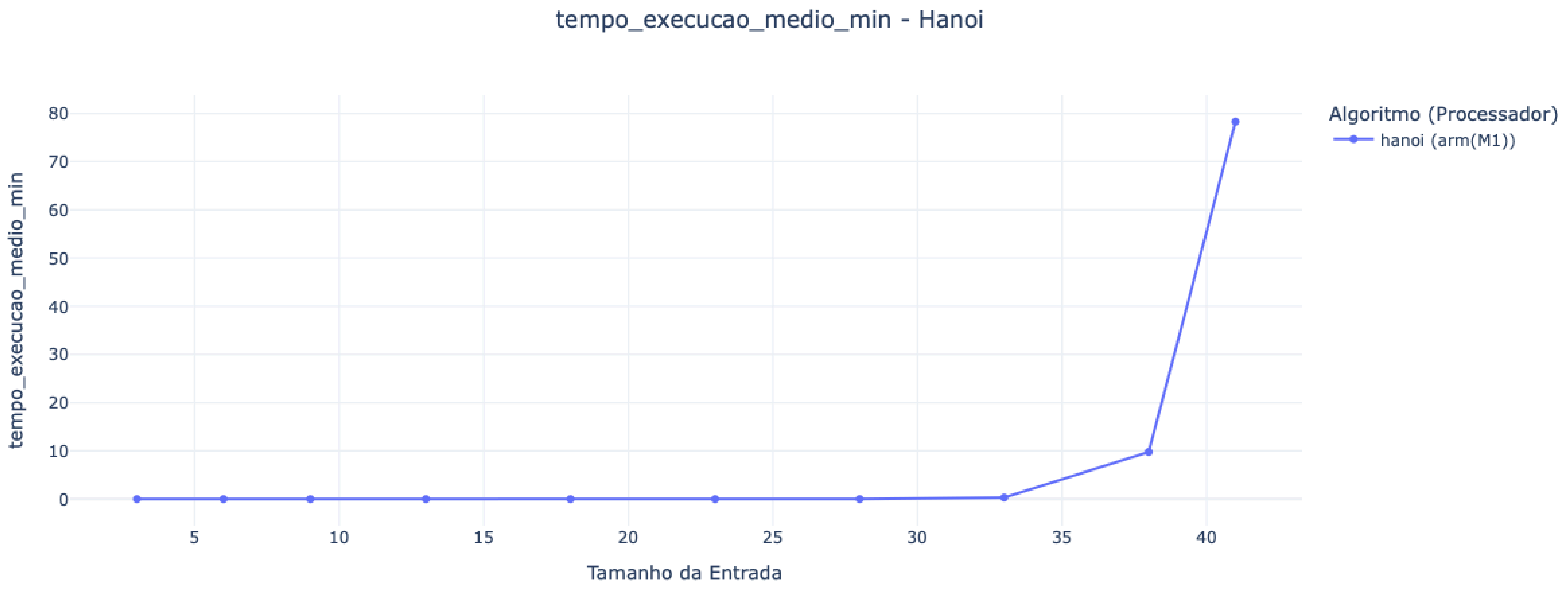
Portanto,

$$T(N) = 2^N - 1 \quad \Rightarrow \quad T(N) \in \mathcal{O}(2^N)$$

Bentchmark Tempo 3-33



Bentchmark Tempo 3-44



Curiosidade

- Teste Simples
- Ambiente Antigo
- Complexidade: $O(2^n)$
- n = 1 → $6{,}5 \times 10^{-5}$ s
- n = 100
- Calculo Simples
- Tempo excede a vida da Terra

Portanto, esse é a fórmula fechada para o número de movimentos necessários para resolver a torre de Hanói com n discos. Trata-se de um algoritmo exponencial. Por exemplo, se n = 100, o número de movimentos a ser executados é:

1267650600228229401496703205376

Fazendo um rápido teste, a função a seguir mede o tempo gasto pelo Python para executar uma única instrução:

```
def tempo():
    inicio = time.time()
    x = 1 + 2 + 3
    print(x)
    fim = time.time()
    return(fim-inicio)
```

A execução da função em um processador Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz demora cerca de 6.556×10^{-5} segundos. Assim, o tempo estimado para resolver esse problema com um programa em Python seria de aproximadamente:

$1267650600228229401496703205376 \times 6.556 \times 10^{-5} = 8.310 \times 10^{25}$ segundos

o que é igual a

2.308×10^{22} horas $= 9.618 \times 10^{20}$ dias $= 2.63 \times 10^{18}$ anos

Sabendo que a idade do planeta Terra é estimada em 4.543×10^{9} anos, se esse programa tivesse sua execução iniciada no momento da criação do planeta, estaria executando até hoje. Estima-se que o

Obrigado!

GITHUB

Angelo_Ferro_Kaio_Guilherme_ws_AA_RR_2025