

Torre de Hanoi

(Análise do algoritmo)



Nome: Lucas Bessa Façanha Pereira

O que é a torre de Hanói

- A Torre de Hanói é um “quebra-cabeça” que consiste em uma base contendo três pinos, em um dos quais são dispostos alguns discos uns sobre os outros, em ordem crescente de diâmetro, de cima para baixo.
- O problema consiste em passar todos os discos de um pino para outro qualquer, usando um dos pinos como auxiliar, de maneira que um disco maior nunca fique em cima de outro menor em nenhuma situação e os discos sejam movidos individualmente.





Código em C

```
void TorreHanoi(int origem, int destino, int auxiliar, int quantidade){  
    if( quantidade == 1 ){  
        count+=1;  
    } else{  
        TorreHanoi(origem, auxiliar, destino, quantidade-1);  
        TorreHanoi(origem, destino, auxiliar, 1);  
        TorreHanoi(auxiliar, destino, origem, quantidade-1);  
    }  
}
```

função de custo e complexidade

- Analisando a função recursiva :

$$T(n) = 1, n = 1$$

$$T(n) = 2T(n-1) + 1, n > 1$$

- Utilizando o método iterativo :

$$T(n) = 2T(n - 1) + 1$$

$$T(n-1) = 2(2T(n-2) + 1) + 1$$

$$T(n-2) = 2(2(2T(n-3) + 1) + 1) + 1$$

função de custo e complexidade

- Calculando e padronizando em função de k :

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

- Analisando o somatório acima, podemos identificar uma P.G de razão 2, portanto utilizando a fórmula da soma de uma P.G :

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

e portanto,

$$T(k) = 2^k T(n - k) + 2^{k-1} - 1$$

função de custo e complexidade

- Para voltarmos ao caso base, consideremos $k = n - 1$:

$$T(n) = 2^{n-1}T(n - (n - 1)) + 2^{n-2} - 1$$

- Função de custo :

$$T(n) = 2^{n-1} + 2^{n-2} - 1$$

$$T(n) = \frac{2^n}{2} + \frac{2^n}{4} - 1 = \frac{6*2^n - 8}{8}$$

- Complexidade :

$$\Theta(2^n)$$



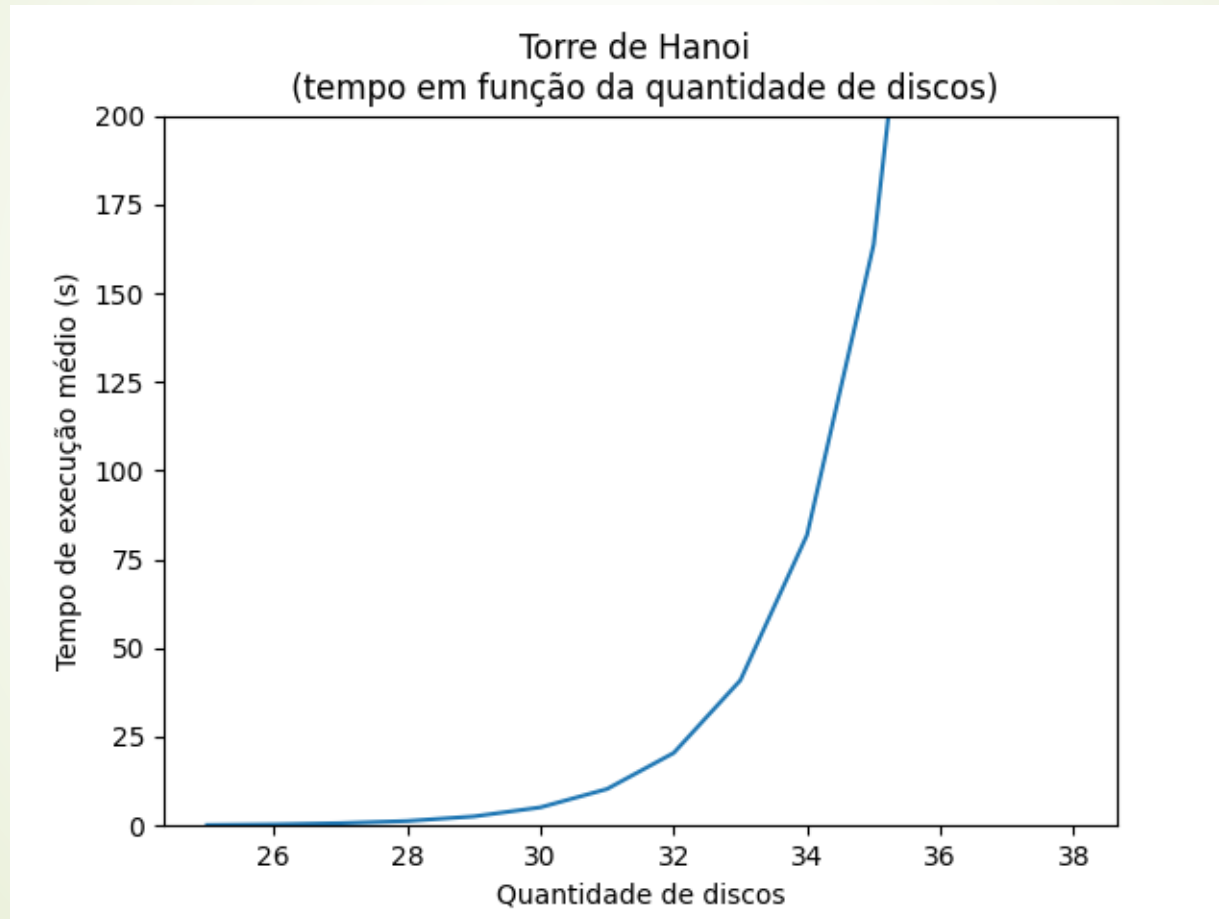
Experimentação

- O programa utilizado após finalizar, exibe na tela o seu tempo de execução.
- Para cada entrada do algoritmo o programa foi executado 13 vezes e os tempos de execução foram salvos em arquivos de log.
- As entradas foram de no intervalo 25 a 38.
- O gráfico do comportamento do algoritmo foi criado com base nas entradas e os seus respectivos tempos de execução médios.

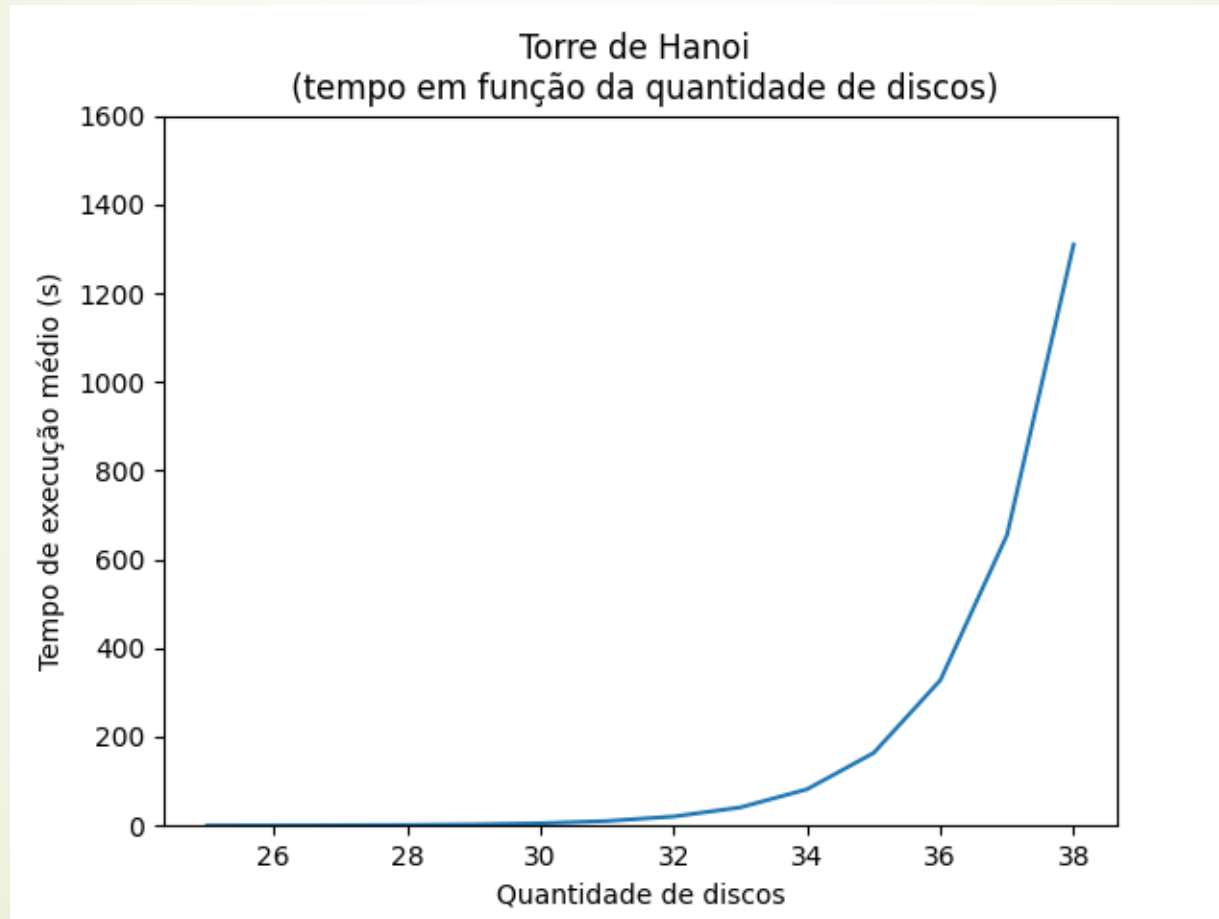
Gráficos



Gráficos



Gráficos





Análise do comportamento assintótico

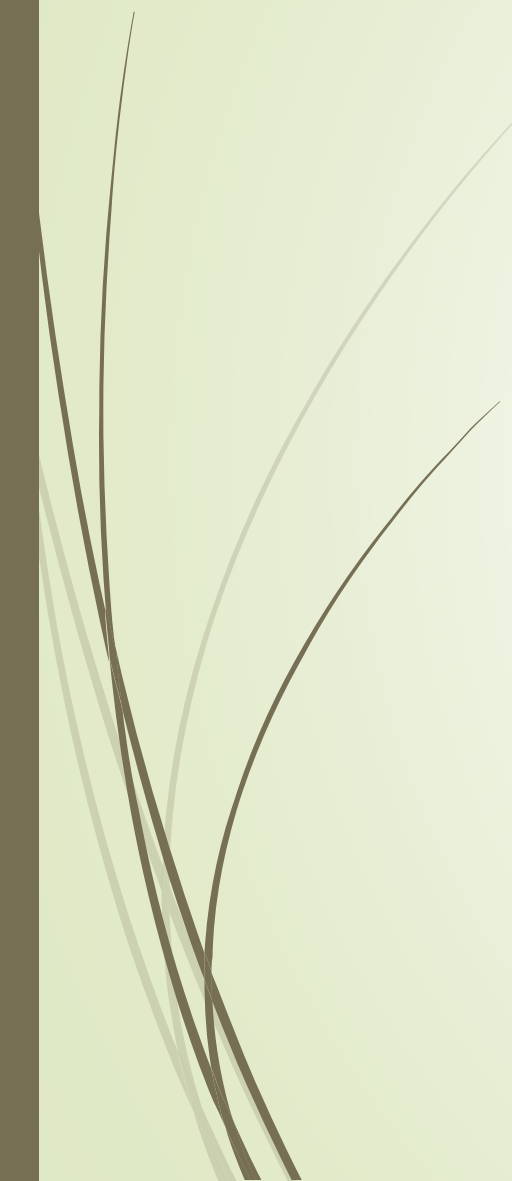
- ▶ Como foi analisado no gráfico, para entradas maiores o tempo de execução do algoritmo tende a ser exponencial
- ▶ Além disso, calculada a complexidade deste algoritmo, podemos afirmar que a função de crescimento da torre de Hanoi é limitada assintoticamente pela função $f(x) = 2^n$

Solução alternativa (modo iterativo)

```
for(k=1; k < (1 << n); k++) {  
    /* obtem haste origem */  
    from = (k&(k-1))%3 ;  
  
    /* obtem haste destino */  
    to  = ((k|(k-1))+1)%3 ;  
  
    /* descreve o movimento */  
    printf("Mova disco de %d para %d\n", from, to) ;  
}
```



Solução alternativa (modo iterativo)

- Função de custo : $f(n) = 2^n - 1$
 - Complexidade : $O(2^n)$
 - A solução é menos custosa mas em termos de complexidade os dois algoritmos são iguais.
- 



Referências

- <https://sites.google.com/a/liesenberg.biz/cjogos/home/materiais-de-apoio/topicos-relativos-a-c/recursao/torre-de-hanoi>
- <https://updatedcode.wordpress.com/2015/03/19/torre-de-hanoi-em-c/>
- <https://www.geeksforgeeks.org/iterative-tower-of-hanoi/>
- <https://daemoniolabs.wordpress.com/2012/05/16/solucao-iterativa-para-o-problema-da-torre-de-hanoi>