Da Contagem de Operações à Complexidade

Torres de Hanói

Mover n discos utilizando 3 postes, sendo um poste de origem e um de destino.

Menor instância do problema

ou *caso trivial* ou *caso base* ?

Apenas **1** disco

1 --- x --- x
A --- B --- C

trivial pois basta um movimento

x --- x --- 1
A --- B --- C

Instância "Clássica" - 3 discos

origem = A, destino = C

Parte 1 - mover n-1 discos de A para B (intermediario)

mover 2 discos de A para B

1
2
3 --- X --- X
A --- B --- C

origem = A, destino = B

2
3 --- X --- 1
A --- B --- C

origem = A, destino = B

---

3 --- 2 --- 1
A --- B --- C

origem = A, destino = B

---

x --- 1
3 --- 2 ---
A --- B --- C

origem = A, destino = B

Parte 1 (mover 2 discos) está resolvida

Parte 2 (mover 1 disco) de origem para destino

x --- 1
x --- 2 --- 3
A --- B --- C

origem = A, destino = C

Parte 3 (mover n-1 discos) de intermediário para destino

Agora o problema passa a ser: mover 2 discos

x --- 1
x --- 2 --- 3
A --- B --- C

origem = B, destino = C

---

1 --- 2 --- 3
A --- B --- C

origem = B, destino = C

---

x --- x --- 2
1 --- x --- 3
A --- B --- C

origem = B, destino = C

x --- x --- 1
x --- x --- 2
x --- x --- 3
A --- B --- C

origem = B, destino = C

Contagem de operações

Operação relevante - movimentacao!

```
hanoi(N, ori, int, dest) {
    if (N == 1)
        move(ori, dest)                // movimenta!
    else
        hanoi(N-1, ori, dest, int)
        hanoi(1,   ori, int , dest)
        hanoi(N-1, int, orig, dest)
}
```

Equação de recorrência:

Caso base:

$$T(1) = 1$$

Caso recursivo:

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

Resolvendo a equação para forma fechada

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

$$T(n) = 1 + 2 \cdot 1 + 2 \cdot 2[1 + 2T(n-3)]$$
$$T(n) = 1 + 2 \cdot 1 + 2 \cdot 2 \cdot 1 + 2 \cdot 2 \cdot 2T(n-3)$$
$$T(n) = 2^0 + 2^1 + 2^2 + 2^3T(n-3)$$

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

acaba quando

$$n - k = 1$$

$$k = n - 1$$

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

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

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

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

Notação assintótica!

$T(n)$ é $O(2^n)$?

$c, n_0 \geq 1$ de forma que

$2^n - 1 \leq c2^n$

$n_0 = 1, c = 1$

$2^n - 1 \leq 1 \cdot 2^n$ para $n \geq 1$

O algoritmo é $O(2^n)$

$T(n)$ é $\Theta(2^n)$?

$c_1, c_2, n_0 \geq 1$ de forma que

$c_1 2^n \leq 2^n - 1 \leq c_2 2^n$

Já encontramos $c_2 = c = 1$ e $n_0 = 1$

Agora para o outro lado

O menor valor que temos para a função seria com:

$n = 1$

$2^n - 1 = 2 - 1 = 1$

Então tomando $c_1 = 1/2$

$\frac{1}{2} 2 \leq 2 - 1$

Assim:

$\frac{1}{2} 2^n \leq 2^n - 1$

$\frac{1}{2} \cdot 2^n \leq 2^n - 1 \leq 1 \cdot 2^n$ para $n \geq 1$

Então: $2^n - 1$ é $\Theta(2^n)$, com: $c_1 = 1/2, c_2 = 1, n_0 = 1$

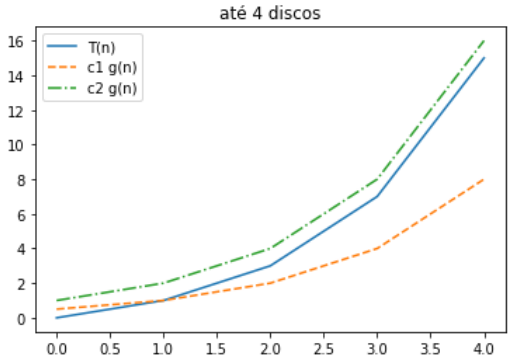
Usando Python para mostrar o crescimento das funções

In [25]:
```
import matplotlib.pyplot as plt
import numpy as np

n = np.arange(5)
T = np.power(2,n) - 1
g = np.power(2,n)
```

In [27]:
```
plt.plot(T, '-')
plt.plot((0.5)*g, '--')
plt.plot((1)*g, '-.')
plt.legend(['T(n)', 'c1 g(n)', 'c2 g(n)'])
plt.title("até 4 discos")
```

Out[27]: Text(0.5, 1.0, 'até 4 discos')



In [28]:
```
n = np.arange(11)
T = np.power(2,n) - 1
g = np.power(2,n)
plt.plot(T, '-')
plt.plot((0.5)*g, '--')
plt.plot((1)*g, '-.')
plt.legend(['T(n)', 'c1 g(n)', 'c2 g(n)'])
plt.title("até 10 discos")
```

Out[28]: Text(0.5, 1.0, 'até 10 discos')