Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: Estrutura de Dados I – BCC 202
Aluno: Felipe Fontenele de Ávila Magalhães
Matrícula: 15.1.4331

5 – A equação de recorrência da função recursiva da potência é:

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

Expandindo a equação:

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

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

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

Fórmula Padrão:

T(n) = k + T(n-k)

Igualar **n-k = 1** para chegar ao caso base:

n – k = 1

k = n – 1

Substituindo k na fórmula padrão:

T(n) = k + T(n-k)

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

= n - 1 + T(1)

= n - 1 +1

= n

Provar no caso base:

T(1) = 1

T(n) = n

Provar por indução na equação geral:

Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: Estrutura de Dados I – BCC 202
Aluno: Felipe Fontenele de Ávila Magalhães
Matrícula: 15.1.4331

T(n+1) = **n +1**

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

T(n+1) = 1 + T(n) **c.q.d**

T(n+1) = **1 + n**

Chegamos à conclusão que a complexidade do algoritmo da potência recursiva é **O(n).** As duas funções, tanto recursiva quanto iterativa, são equivalentes apresentando a mesma complexidade **O(n)**, contudo a função **iterativa** é mais **eficiente** devido ao baixo custo de memória em comparação com a função recursiva.

6b - A equação de recorrência da função da busca binária é:

$$\begin{cases} T(1) = 3 \\ T(n) = T(n/2) + 3 \end{cases}$$

Expandindo a equação:

T(n) = T(n/2) + 3

= T(n/2/2) + 3 + 3 = T(n/4) + 6

= T(n/2/2/2) + 3 + 3 + 3 = T(n/8) + 9

= T(n/2/2/2/2) + 3 + 3 + 3 + 3 = T(n/16) + 12

Fórmula Padrão:

T(n) = T(n / 2^k) + 3K

Se colocarmos n como 2^k temos:

n = 2 ^ k

lg n = lg (2 ^ k)

lg n = k

Assim, substituindo o n, temos que:

T(2 ^ k) = T (2 ^ k / 2 ^ k) + 3k

= T (1) + 3K

= 3 + 3K

Como k = lg n, então, por fim temos:

T(n) = 3 + 3 * lg n

E que:

T(1) = 3 + 3 * lg 1

Sabemos que lg 1 = 0, pois qualquer numero elevado a 0 dá 1

T(1) = 3 + 3 * 0 = 3, provando assim o caso base.

Por fim, a complexidade é **O (lg n).**