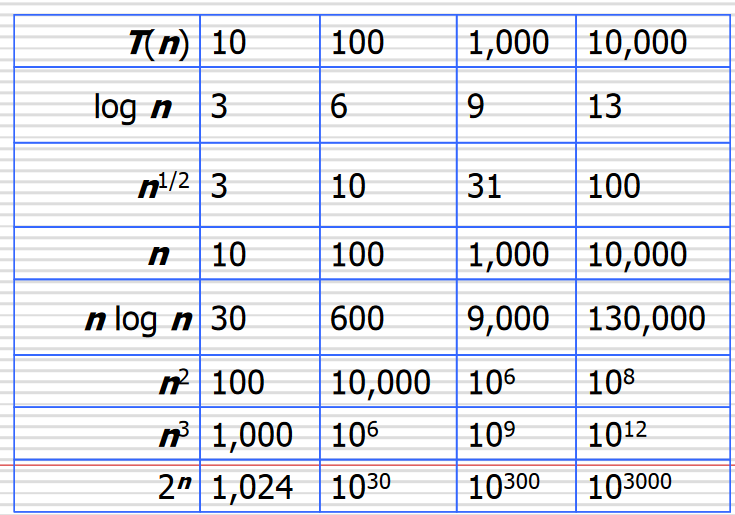
Dalam menganalisis suatu algoritma, meskipun dapat dicari tahu secara akurat berapa nilai kompleksitas waktu () dan kompleksitas ruang () dari algoritma tersebut, namun seringkali kita hanya tertarik pada nilai kompleksitasnya secara asimtotik atau biasa disebut *order of growth*. Dengan hanya memfokuskan penganalisisan algoritma pada *order of growth*, kita dapat mengklasifikasikan fungsi-fungsi yang merepresentasikan kompleksitas algoritma menjadi lebih sederhana. Dalam hal ini, kita dapat menggunakan notasi Big-O, Big-Theta, dan Big-Omega.

 Gambar 1. Pengklasifikasian kompleksitas algoritma berdasarkan order of growth memudahkan perhitungan komputasi yang dilakukan.

Secara definisi matematis, notasi Big-O, Big-Theta, dan Big-Omega dapat didefinisikan sebagai berikut:

Atau secara definisi limitnya dapat didefinisikan sebagai berikut:

$$

$$

Secara sederhana dapat diartikan bahwa: 1. Big-O: menunjukkan kelas *order-of-growth* yang menjadi upper bound dari kompleksitas algoritma yang dianalisis. 2. Big-Omega: menunjukkan kelas *order-of-growth* yang menjadi lower bound dari kompleksitas algoritma yang dianalisis. 3. Big-Theta: menunjukkan kelas *order-of-growth* yang menjadi upper bound dan lower bound dari kompleksitas algoritma yang dianalisis.

Dalam mengklasifikan kompleksitas algoritma, pengunaan Big-Theta jauh lebih sering digunakan dibandingkan Big-O dan Big-Omega.

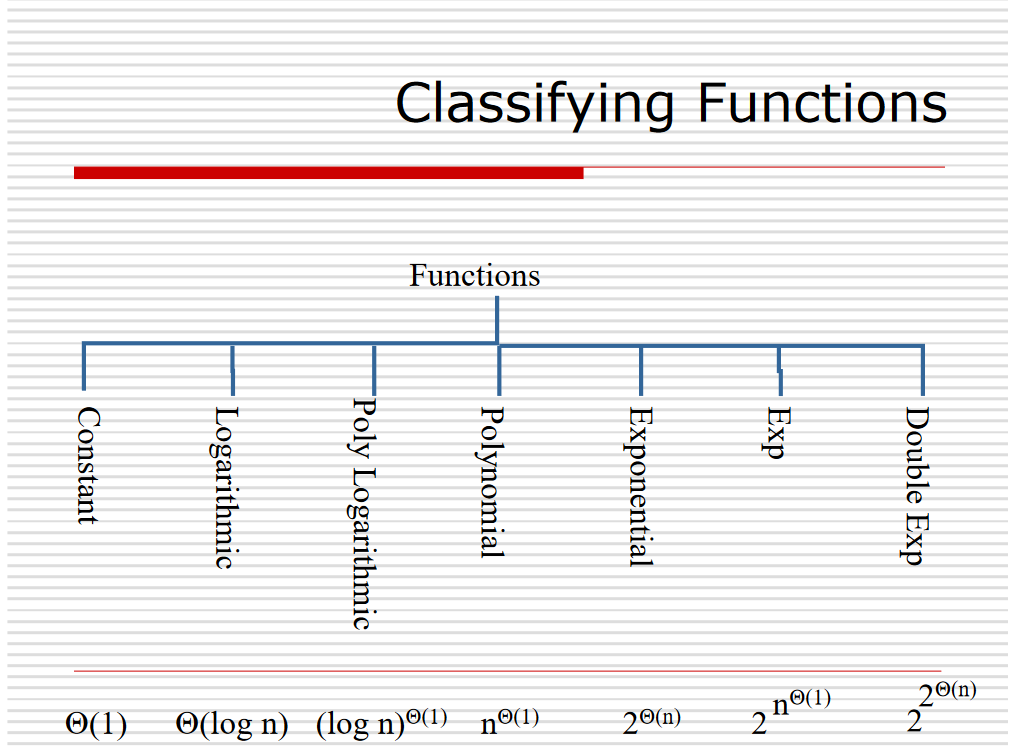
Meskipun kita dapat menggunakan definisi matematis untuk mengklasifikasikan kompleksitas algoritma, namun dalam prakteknya, kita dapat menggunakan aturan-aturan berikut untuk mengklasifikasikan kompleksitas algoritma:

1. Abaikan *low-order* terms berarti bahwa kita hanya tertarik pada fungsi yang memiliki “nilai yang mendominasi” ketika n menuju takhingga. Sebagai contoh, jika kita memiliki fungsi , maka kita hanya tertarik pada fungsi saja. Hal ini dikarenakan fungsi .
2. Abaikan konstanta-konstanata pengali pada fungsi.

Dengan mengikuti r*ule of thumb* tersebut,

merupakan fungsi yang memiliki kompleksitas yang sama, yaitu . Dan fungsi-fungsi berikut:

merupakan fungsi yang memiliki kompleksitas yang sama, yaitu atau .

 Gambar 2. Pengklasifikasian fungsi-fungsikompleksitas algoritma berdasarkan order of growth (Big-Tehta).