Brief Introduction to Algorithm Complexity—CS2163 Java

**1, the Big-O notation:** it is used to describe the algorithm complexity, and it can be understood as “in the order of”, e,g,

**O(n)** : in the order of n

**O(log(n))** : in the order of log(n), where log(n) is an abbreviation of log2n, with base 2 logarithm.

**O(nlog(n))** : in the order of n times log(n)

**O(n2) :** in the order of n square, n2

**O(2n) :** in the order of 2 to the power of n, 2n

**2, algorithm complexity:** the following table summarizes the algorithm complexity for the search and sorting algorithms mentioned in Lesson 7.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| algorithm name | linear search | binary search | insertion sort | selection sort | bubble sort | quick sort |
| time complexity | O(n) | O(log(n)) | O(n2) | O(n2) | O(n2) | O(nlog(n)) |

Student are required to remember the above table, but not required to know how it is derived.

**3, time complexity curves** : the following figure draws different time complexity curves in a graph, and the x-axis is the size of the problem, denoted by N (the same as the n we used above), and the Y-axis is the time complexity of algorithms in Big-O notation. From this figure, we can tell which big-O notation has higher complexity than others, such as:

O(2n) > O(n3) > O(n2) > O(nlog(n)) > O(n) > O(log(n))

Students need to understand the above relationship with the help of the figure below.

