Time complexity represents the number of times a statement is executed. The time complexity of an algorithm is NOT the actual time required to execute a particular code, since that depends on other factors like programming language, operating software, processing power, etc. The idea behind time complexity is that it can measure only the execution time of the algorithm in a way that depends only on the algorithm itself and its input. To express the time complexity of an algorithm, we use something called the *"Big O notation"*. The Big O notation is a language we use to describe the time complexity of an algorithm. (**Performance of an algorithm described using Big O Notation)**

1. **Constant Time Algorithms -O(1):**
   ```
   Int result=n1+n2; //10+20
   ```
   This piece of code takes a constant amount of time to run. It's not dependent on the size of n.

2. **Linear Time Algorithms – O(n)**
   we mean that it grows directly proportional to the size of its inputs.
   ```
   for ( i = 0; i < N; i++ )
       statement; // this for loop run n times.
   ```

3. **Quadratic (N*N) Time algorithms**
   ```
   for ( i = 0; i < N; i++ ) {
               for ( j = 0; j < N; j++ )
                 statement;
   }
   ```
   The running time of the two loops is proportional to the square of N. When N doubles, the running time increases by N * N.

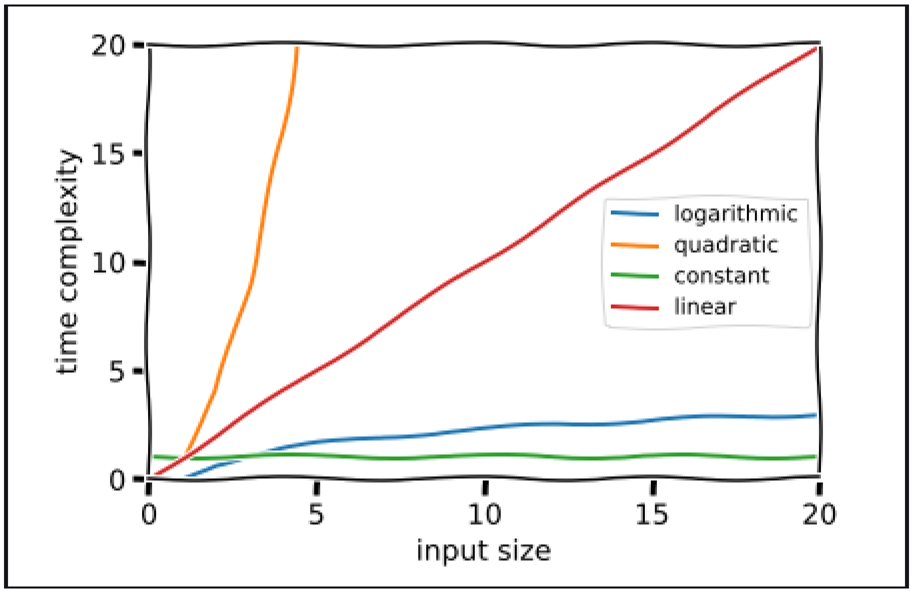
4. **Logarithmic Time Algorithms – O(log n)**
   ```
   while ( low <= high )
    {
     mid = ( low + high ) / 2;
     if ( target < list[mid] )
       high = mid - 1;
     else if ( target > list[mid] )
       low = mid + 1;
     else break;
   }
   ```
   The running time of the algorithm is proportional to the number of times N can be divided by 2. This is because the algorithm divides the working area in half with each iteration.

5. **N * log (N) Time Algorithms**

```
void quicksort ( int list[], int left, int right ){
  int pivot = partition ( list, left, right );

  quicksort ( list, left, pivot - 1 );

  quicksort ( list, pivot + 1, right );

}
```

The running time consists of N loops (iterative or recursive) that are logarithmic, thus the algorithm is a combination of linear and logarithmic.

In general, doing something with every item in one dimension is linear, doing something with every item in two dimensions is quadratic, and dividing the working area in half is logarithmic. There are other Big O measures such as cubic, exponential, and square root, but they're not nearly as common. Big O notation is described as O ( ) where is the measure. The quicksort algorithm would be described as O (N * log (N)).


time complexity
input size
logarithmic
quadratic
constant
linear