Data Structure & Algorithms using Python

Quiz Answers

Time Complexity Analysis

Name : **Harsh Siddhapura**

Degree : **Bachelor of Technology**

Dept. : **Information & Communication Technology**

Two main measures for the efficiency of an algorithm are -

Processor and memory
Complexity and capacity
Time and space
Data and space

In theoretical analysis the time factor when determining the efficiency of algorithm is measured by -

Counting microseconds
Counting the number of statements in code
Counting the number of unit operations
Counting the kilobytes of algorithm

If the number of primary operations of an algorithm that takes an array of size n as input are 3n^2 + 5n. The worst case time complexity of the algorithm will be ?

O(n^3)
O((n^2)*logn)
O(n^2)
O(n)

The worst case time complexity of Linear search is :

O(n)
O(n^2)
O(nlogn)
O(logn)

Worst case time complexity of insertion sort is ?

O(N)
O(N^2)
O(NLogN)
O(LogN)

Worst case time complexity of Selection sort is ?

O(N)
O(N^2)
O(NLogN)
O(LogN)

What will be the Time Complexity of the following code in terms of 'n' ?
Note : Assume k to be a constant value
Refer the code in C++ -

```
 for(int i = 0; i < n; i++){
   for(int j = 1 ; j < k; j++){
     cout << (i + j ) << endl;
    }
 }
```

Refer the same code in Java -

```
for(int i = 0; i < n; i++){
     for(int j = 1 ; j < k; j++){
       System.out.println(i + j);
     }
}
```

Refer the same code in Python -

```
for i in range(n):
   for j in range(k):
      print(i+j)
```

O(n^2)
O(n)
O(logn)
None of these

What will be the Time Complexity of the following code in terms of 'n' '?

```
for(int i = 0; i < n ; i++){
   int k = n;
   while(k > 0){
        k/=2;
   }
}
```

Same code is Python is:

```
for i in range(n):
   k=n
   while k>0:
      k //= 2
```

O(nk)
O(nlogn)
O(n^2)
O(n)

What will be the Time Complexity of the following code in terms of 'n' ?

```
while(n > 0){
  n = n / 4;
}
```

Same code in Python is

```
while n>0:
    n = n//4
```

O(n)
O(logn to the base 4)
O(n^2)
None of these

What is the time complexity of following recursive code ?

```
def multiplyRec(m, n):
    if n==1:
        return m
    return m + multiplyRec(m,  n - 1)
```

O(m*n)
O(n)
O(n^2)
O(m)

What is the time complexity of following recursive code ?

```
def sumOfDigits(n):
    if n < 10:
        return n
    sum = (n % 10) + sumOfDigits(n//10)
    return sum
```

O(logn) - log is to the base 10
O(n)
O(n^2)
None of these

What is the recurrence relation for merge sort :

T(n) = 2T(n/2)
T(n) = 2T(n/2) + k
T(n) = 2T(n/2) + O(n)
T(n) = 2T(n/2) + O(log n)

For merging two sorted arrays of size m and n into a sorted array of size m+n, we require operations -

O(m * n)
O(m + n)
O(m) if m >= n
O(n) if n > m

What is the time complexity of merge sort :

O(n)
O(n^2)
O(nlogn)
O(log n)

What is the time complexity of following recursive code ?

```
def fib(n):
   if n == 0 or n == 1:
   return n
return fib(n - 1) + fib(n - 2)
```

O(n)
O(n^2)
O(2^n)
None of these
