

URBAN

## **Time Complexity – Competitive Practice Sheet**

1. Fine the time complexity of the func1 function in the program show in program1.c as follows:

#include <stdio.h>

void **func1**(int array[], int length)

{

    int sum = 0;

    int product = 1;

    for (int i = 0; i < length; i++)

    {

        sum += array[i];

    }

    for (int i = 0; i < length; i++)

    {

        product \*= array[i];

    }

}

int **main**()

{

    int arr[] = {3, 5, 66};

**func1**(arr, 3);

    return 0;

}

1. Fine the time complexity of the func function in the program from program2.c as follows:

void **func**(int n)

{

    int sum = 0;

    int product = 1;

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < n; j++)

        {

**printf**("%d , %d\n", i, j);

        }

    }

}

1. Consider the recursive algorithm below, where the random(int n) spends one unit of time to return a random integer which is evenly distributed within the range [0,n]. If the average processing time is T(n), what is the value of T(6)?

int **function**(int n)

{

    int i;

    if (n <= 0)

    {

        return 0;

    }

    else

    {

        i = **random**(n - 1);

**printf**("this\n");

        return **function**(i) + **function**(n - 1 - i);

    }

}

1. Which of the following are equivalent to O(N)? Why? (k is a constant)
2. O(N + P), where P < N/9
3. 0(9N-k)
4. O(N + 8log N)
5. O(N + M2)
6. The following simple code sums the values of all the nodes in a balanced binary search tree. What is its runtime?

int **sum**(Node node)

{

    if (node == NULL)

    {

        return 0;

    }

    return **sum**(node.left) + node.value + **sum**(node.right);

}

1. Find the complexity of the following code which tests whether a give number is prime or not?

int **isPrime**(int n){

    if (n == 1){

        return 0;

    }

    for (int i = 2; i \* i < n; i++) {

        if (n % i == 0)

            return 0;

    }

    return 1;

}

1. What is the time complexity of the following snippet of code?

int **isPrime**(int n){

    for (int i = 2; i \* i < 10000; i++) {

        if (n % i == 0)

            return 0;

    }

    return 1;

}

**isPrime();**