Kent Mark

3/9/2021

Com S 311 – Exam 1

Exam 1

1. The outer loop i does n, n/2, n/(2\*2), n/(2\*2\*2) and so on until the value becomes 1. The condition to break the outer loop is n/2k < 1. So n < 2k, taking log on both sides equals log(n) < k meaning k is nearly equal to log(n). Therefore, the runtime of the outer loop is O = (log(n)).

The inner loop j does 1, 2, 3, 4, … n + 1. In this loop j will increase until it reaches n + 1, which is the condition for the loop to break. This means there is an n operation for each outer loop. This would give a runtime of O(n) for the inner loop.

Multiplying the runtimes of both loops give me a final runtime of O(n log(n)).

1. This method was written in Java. Assume that there is a main method that accepts an array of number from the user.

public class Question3{

static int divNconq(int[] arr,int lower,int upper){

int middle = (lower + upper) / 2;

int token = -1;

if(arr[upper] < lower || arr[lower] > upper)

return -1;

if(upper < lower)

return -1;

if(middle < arr[middle]){

if(arr[middle] <= upper)

token = divNconq(arr, arr[middle], upper);

if(token == -1)

token = divNconq(arr, lower, middle - 1);

}

else if(arr[middle] < middle){

if(arr[middle] >= lower)

token = divNconq(arr, lower, arr[middle]);

if(token == -1)

token = divNconq(arr, middle + 1, upper);

}

else{

token = middle;

}

return token;

1. Using the Master’s Theorem we have a = 2, b = 4, and f(n) = .

As such we can use Case 2 of the Master Theorem to get:

(

1. I do not know how to solve this problem.

}