Conder Shou

Homework 1 – Written Part

**2.1**

The following functions are ordered by growth rate in increasing order:

2/N, 37, N0.5, N, N log logN, N logN is equal in growth rate to N log(N2), N log2N, , N1.5, N2, N2 logN, N3,

2N/2, 2N

**2.6**

a.

On day N, the fine would be 22^(N-1)

b.

D = 22^(N-1)

log2D = 2N-1

log2(log2D) = N -1

N = log2(log2D) + 1

It would take O(log log D) days to reach D dollars.

3.)

a. The first “for” loop only runs 23 times. The second for loop runes “n” times.

Running time: O(N)

b. A nested for loop. The running time is the running time of the statement multiplied by the product of the sizes of all loops. Each for loop here has a Big-Oh running time of O(N).

Running time: O(N2)

c. The problem is cutting its size by a fraction each time.

Running time: O(logN)

**2.11**

a. If linear, 0.5ms \* 5 = 2.5 ms

b. If O(NlogN), the new running time is similar to a linear relationship, but slightly more. So somewhere around, but above 2.5 ms. (0.5\*5\*log2(5))

c. If quadratic, 0.5\*52 = 12.5 ms

d. If cubic, 0.5\*53 = 62.5 ms

**2.15**

Algorithm used: Binary search

Running time: O(logN)

An efficient algorithm for determining if there exists an integer *i* such that in an array of integers, Ai = *i,* would be binary search, whose running time is O(logN). The pseudocode below explains it as thus:

Get the values of the array to be searched through

Get the desired value x to look for

Initialize the lower bound of the array and the upper bound of the array. To begin, this will mean that the lower bound is equal to zero, and the upper bound is the last element in the array.

While the lower bound is less than the higher bound, the program will find the middle of the two bounds by adding the upper and lower and dividing the sum by two. We should ensure it is integer division here.

If the middle element found is less than the desired value x, then we will make the lower bound, the middle index plus one.

If the middle element found is greater than the desired value x, then we will make the upper bound, the middle index minus one.

Otherwise, if neither of these cases are true, then the middle element must either be equal to the desired value x, or the array does not contain the desired value x.

If the middle element is equal, then the position of the middle element AKA the position of the x value, is returned.

At this point, the while loop will end.

If by this time the program still hasn’t returned the desired x value, then we return an error message saying that the value x was not found in the array.