Matthew Barlow

2.1 Order the following functions by growth rate: N, √N, N1.5, N2, NlogN, N log log N, N log2 N, N log(N2), 2/N, 2N, 2N/2, 37, N2 log N, N3. Indicate which functions grow at the same rate.

Lowest to Highest Growth: None of the functions have the exact same growth, but they are tiered by their level of growth. Functions on the same line grow with similar trends.

2/N

37

sqrt(N)

N

NloglogN NlogN   N $\log(N^2)$

$N\log^2 N$

$N^{1.5}$

N^2 $N^2 \log N$

N^3

$2^{(N/2)}$ 2^N

. **2.6 In a recent court case, a judge cited a city for contempt and ordered a fine of $2 for the first day. Each subsequent day, until the city followed the judge’s order, the fine was squared (that is, the fine progressed as follows: $2, $4, $16, $256, $65, 536, . . .).**

a. **What would be the fine on day N?**

   F(N)=2^$2^{(N-1)}$

b. **How many days would it take the fine to reach D dollars? (A Big-Oh answer will do.)**

N={(log[logD])/(log[2log2])} +1

(10 pts) Give an analysis of the Big-Oh running time for each of the following program fragments:

```
int sum = 0;
for ( int i = 0; i < 23; i ++)
    for ( int j = 0; j < n ; j ++)
        sum = sum + 1;
```

1 for initialization of sum : 1
1 to set int i, 23+1=24 tests,23 increments: 48
1 to set int j, n+1 tests, n increments, executed 23 times: 23(2n+2)
1 addition, 1 initialization, executed n times:2n
TOTAL=1+48+23(2n+2)+2n=95+48n -> 48n->n

```
int sum = 0;
for ( int i = 0; i < n ; i ++)
    for ( int k = i ; k < n ; k ++)
        sum = sum + 1;
```

1 for initialization of sum:1
1 to set int i, n+1 tests, n increments:2n+2
1 to set k, n+1 checks, n increments, n executions: n*(2n+2)
1 addition, 1 initialization, n executions: 2n
TOTAL=1+(2n+2)+n*(2n+2)+2n=2n2 +6n +3 -> 2n2-> n^2

```
public int foo(int n, int k) {
    if(n<=k)
        return 1;
    else
        return foo(n/k,k) + 1;
}
```

else loop is worse case:
calls foo log(n)/log(k) times
each call of foo has 1 check and 1 return:2

Total=2*[log(n)/log(k)]->log(n)/log(k)-> log(n)

- **2.11 An algorithm takes 0.5 ms for input size 100. How long will it take for input size 500 if the running time is the following (assume low-order terms are negligible):**

  a. linear

  (500/100)=O(500)/O(100)

  O(500)=5*O(100)=5*(.5e-3s)=2.5ms=O(500)

  b. O(NlogN)

  [500log(500)]/[100log(100)]=O(500)/O(100)

  O(500)=O(100)*[500log(500)]/[100log(100)]=(.5ms)*6.74=3.374ms=O(500)

  c. quadratic

  O(100)*(5002)/(1002)=O(500)

  (.5ms)*(250000/1000)=(.5ms)(25)=12.5ms=O(500)

  d. cubic

  O(100)*(5003)/(1003)=O(500)

  (.5ms)*(125)=62.5ms=O(500)

2.15 Give an efficient algorithm to determine if there exists an integer i such that Ai = i in an array of integersA1 <A2 <A3 <···<AN.What is the running time of your algorithm?

```
Public boolean findIt(int[] arr){

 int upperbound=arr.length-1;

 int lowerbound=0;

 while(upperbound>=lowerbound){

     index=(upper_bound + lower_bound)/2;
```

```
    if(arr[index]==index+1){ return True;}

    else if(arr[index]>index+1){upperbound=index-1;}

    else if(arr[index]<index+1){lowerbound=index+1;}

 return False;
```

Run Time:

O(findIt)=4[log(N)/log(2)]+7-> log_2(N)->log(N)