Mario Bolivar

Mjb160330

September 5, 2018

CE 3345.001

Assignment 2

1.) Least -> Greatest (1st pair of functions and second pair of functions have the same growth rate)

2/N, 37, $\sqrt{N}$, N, N log log N, **N log N**, **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.) For each of the following code fragments give running time analysis (Big Oh)

A.)
```
sum =0;
for ( i=0; i < n ; i++)
        sum++;
```
f(n) = c1 + n + c2 = **O(n)**

B.)
```
sum =0;
for( i = 0; i < n; i++)
        for(j = 0; j < i ; j++)
                sum++;
```
*f(n) = c1 + n*n + c2 =* **O(n^2)**

C.)
```
sum =0;
for( i = 0; i < n; i++)
        for( j = 0; j < i *i ; j++)

                for( k = 0; k<j; k++)
                        }sum++;
```
f(n) = c1 + nn^2*n^2+ c2 =* **O(n^5)**

3.) What is the time complexity of the below function?

```
void fun(int n, int arr[])

{
        int i = 0, j = 0;
        for(; i < n; ++i)
                while(j < n && arr[i] < arr[j])
                        j++;
```
*f(n) = c1 + c2 + n*n =* **O(n^2) -> Assuming for all i, and j :** arr[i] < arr[j] since worst

case.

4.) An algorithm takes o.5 ms for input size 100. How long will it take for input size 500 if the running time is the following (assume lower order terms are negligible)

A.) Linear (n) → **2.5 ms**

B.) O(N log N) → **6.75 ms**

C.) Quadratic → **1,250 ms**

D.) Cubic → **625,000 ms**

5.) Show that N^{62} can be computed with only eight multiplication

$n^{62} = n^{60} * n^2$

$n^{60} = n^{48} * n^{12}$

$n^{48} = n^{24} * n^{24}$

$n^{24} = n^{12} * n^{12}$

$n^{12} = n^8 * n^4$

$n^8 = n^4 * n^4$

$n^4 = n^2 * n^2$

$n^2 = n * n$ → ***Total of 8 multiplications***

6.) Give an efficient algorithm to determine if there exists an integer i such that $A_i = i$ in an array of N distinct integers, sorted in ascending order. What is the running time of given algorithm?

Because is it sorted, we can use a binary search which will find the integer i in the array in O(log N) time. Pseudocode below:

- Compare integer i to the middle element in the array
- If integer i == middle element, return the middle index
- Else if integer i > middle element, then i can only be in the right half of the array after the mid element, so we recall the method using the right half of the array.
- Else, x is smaller than the middle element therefore we recall the method recursively using the left half of the array.

7.) Give an efficient algorithm along with running time analysis to find the minimum subsequence sum (Assume the minimum sum is either 0 or a negative value)

```
static int minSubSum(int arr[], int n){


        int min_max = 0;

        int min_current = 0;

        for (int i = 0; i < n; i++){

                if (min_max > 0)

                        min_max = arr[i];

                else

                        min_max += arr[i];

                min_current = Math.min(min_current, min_max);

        }

        return min_current;

}
```

f(n) = c1 + c2 + n + c2 + c3 + c4 → **O(n)**