Lab Assignment - 1

Rohan Verma (1510110508)

February 15, 2017

1 Solve the following recurrences using Masters method

1.1 $T[n] = 4T[n/2] + n^2$

Comparing the recurrence with the form used for master method,
$a = 4, b = 2, c = 2$
and $log_b a = log_2(2^2) = 2 = c$.
Therefore, $T(n) = \theta(n^c logn) = \theta(n^2 log(n))$

1.2 $T[n] = 2T[n/2] + c$

Comparing the recurrence with the form used for master method,
$a = 2, b = 2, c = 0$
and $log_b a = log_2(2) = 1 > c$.
Therefore, $T(n) = \theta(n^{log_b a}) = \theta(n^1)$

1.3 $T[n] = T[n/2] + T[n/4] + n^2$

The given relation is not in a form on which master method can be applied.

1.4 $T[n] = 2T[n/4] + logn$

Comparing the relation with forms used in master method,
$a = 2, b = 4, f(n) = logn$
$n^{log_b a} = n^{log_4 2} = \sqrt{n}$
For large n,
$\sqrt{n} > logn$

Therefore, $f(n) = logn = O(n^{1/2-\epsilon})$
Here, $\epsilon > 0$
Hence, using master method, $T(n) = \theta(\sqrt{n})$

1.5 $T[n] = 2T[n/4] + n!$

$a = 2, b = 4, f(n) = n!$
$n! = \prod_{i=0}^{n-1}(n-i)$
$log_b a = \frac{1}{2} \implies n^{log_b a} = \sqrt{n}$
Since, $n! > n^{1/2} \implies n! = n^{1/2+\epsilon}$
Thus, using master method, $f(n) = \Omega(n^{1/2+\epsilon})$
$\implies af(n/b) \leq cf(n)$
$\implies 2(n/n)! \leq cn!$
$\implies T(n) = \Theta(n!)$

2 Solve the following recurrences using Recursion tree method

2.1 $T[n] = T[n/3] + 2T[n/4] + n$

2.2 $T[n] = T[n-5] + 1/n$

2.3 $T[n] = T[\sqrt{n}] + 1$

2.4 $T[n] = 2T[n/2] + n/logn]$

2.5 $T[n] = T[n-1] + T[n/2] + n$

[handwritten]

3 Give an algorithm to solve the following problem: Given n, a positive integer, determine whether n is the sum of all its divisors. Analyze the asymptotic runtime complexity of this algorithm. Write the code in C and plot the observations.

3.1 Algorithm

Algorithm 1 Check if n is the sum of all its divisors

begin
take input n
initialize sumOfFactors:=0 and i:=1
while $i < n$ **do**
  if n mod i == 0 **then**
    sumOfFactors = sumOfFactors + 1
  end if
  i = i + 1
end while
if n == sumOfFactors **then**
  return true
else
  return false
end if

3.2 Code

```
#include <stdio.h>
#include <time.h>
#include <math.h>

int main(){

        unsigned long long n;

        scanf("%llu", &n);

        unsigned long long sof = 0, i = 1;

        clock_t t = clock();

        while(i < n){
                if (n % i  == 0)
                        sof += i;
                i+=1;
        }
```

```
        t = clock() - t;

        printf("%llu %f\n", n, ((double)t)/CLOCKS_PER_SEC);

        if (n == sof)
                printf("True\n");
        else
                printf("False\n");

        return 0;
}
```

```
#usr/bin/bash
for i in 10000000 20000000 30000000 40000000 50000000
do
        echo $i | ./perfect_number
done
```

3.3 Complexity

The runtime complexity of this algorithm will be $O(n)$ since, we have to check all numbers between 1 and n. The space complexity will be $O(1)$ since we don't need any additional memory.

3.4 Results

The results were found to be increasing linearly with n.

Table 1: Increasing n by times 10

1	0.000001
10	0.000000
100	0.000002
1000	0.000028
10000	0.000113
100000	0.001085
1000000	0.010230
10000000	0.093336
100000000	0.916945
1000000000	9.189980
10000000000	92.604395

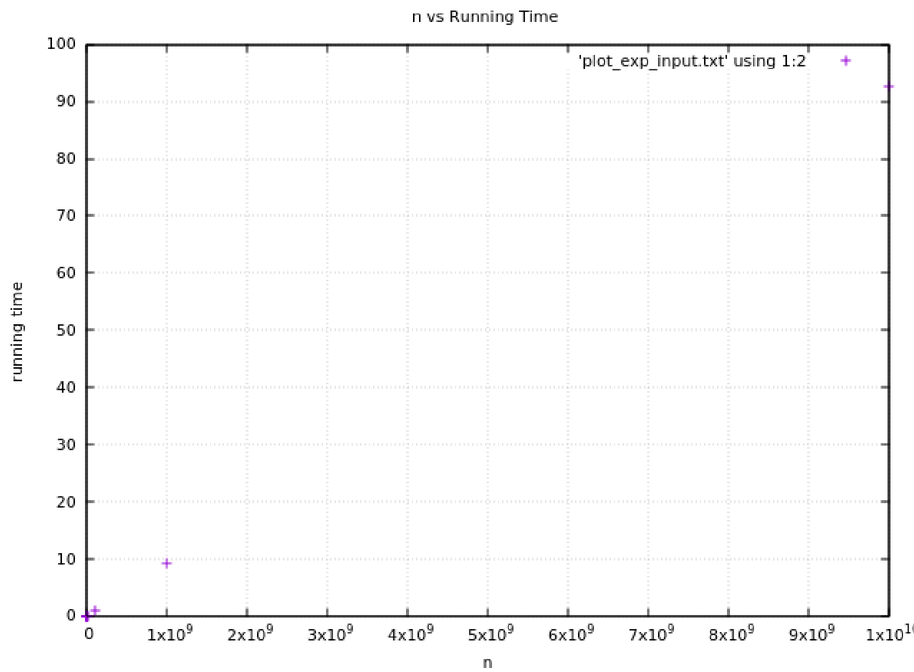
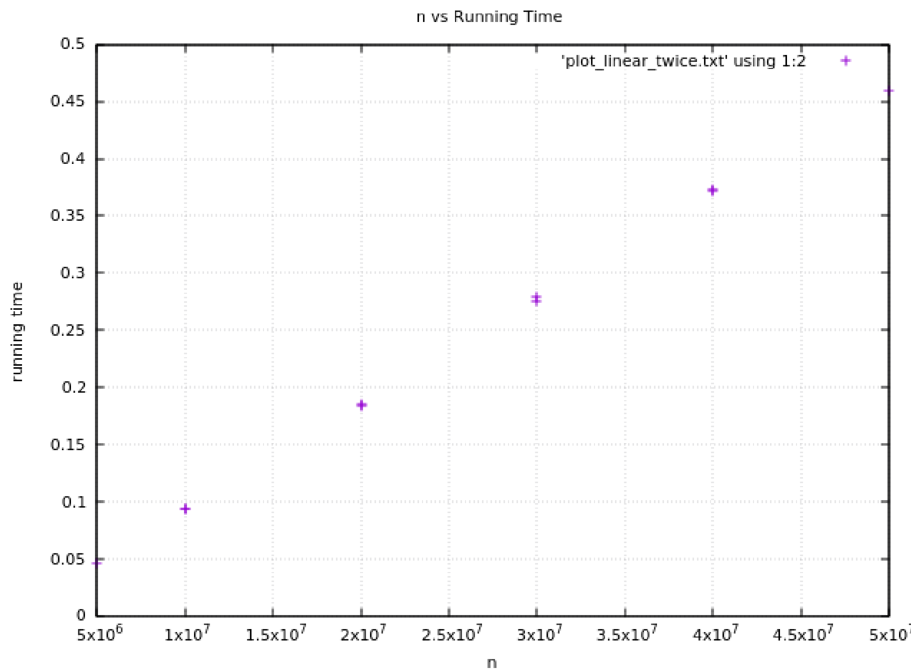


Table 2: Increasing n by times 2

10000000	0.094203
20000000	0.185652
30000000	0.275933
40000000	0.371852
50000000	0.459668


n vs Running Time
'plot_linear_twice.txt' using 1:2
running time
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
5x10^6
1x10^7
1.5x10^7
2x10^7
2.5x10^7
3x10^7
3.5x10^7
4x10^7
4.5x10^7
5x10^7
n

4 Consider the naive Monte Carlo algorithm for Primality testing presented in textbook, where Power(x, y) = x^y. What should be the value of t for the algorithms output to be correct with high probability? Write the code in C and analyze the observations.

```c
int primeMC(int num){
        int i, j, m;

        if (num == 1)
                return 1;
        for(i = 2; i < t; i++){
                m = pow(num, 0.5);
                j = rand() % m + 2;
                if (num % j == 0)
                        return 0;
        }
        return 1;
}
```

5 Show the complexity for T[n] = T[$n/2$ + 21] + $\theta(n)$ is $\theta(nlog(n))$

[Handwritten]