Lab Assignment 1

Name: Muhtasim Mahmud

ID: 19101652

Section: 01

Primality Testing

1. Naive approach:

outer for loop:

$$time = \frac{(higher\ limit - lower\ limit)+1}{increment}$$

$$= \frac{(n-2)+1}{1}$$

inner for loop:

i = 2	j = 2
i = 3	j = 2
i = 4	j = 2+3
i = 5	j = 2+3+4
⋮	⋮

sum = $\frac{n}{2}$ (first num. + last num)

here, for inner for loop i used the formula of sum of certain consecutive num of a range which is : $\frac{n}{2}$ (first num + last num)

here that range of sum = 2 to (n-1)

```
bool prime[n] = {0};

void isPrime (int n)
{

   for (int i=2; i<=n; i++)          ——  (n-2+1)/1 → n
   {
      int cnt = 0;
      for (int j= 2; j<i; j++)        ——  n/2 (first num.+ last num) → n
      {
         if (i%j == 0)
         {
            cnt ++;
         }
      }
   if (cnt == 0)
   {  prime[i] = 1;
   }
   }
   for (int i= 2; i<= n; i++)         ——  (n-2+1)/1 → n
   { if (prime[i])
        S.O.P (i+ " ");
   }
}
```

total time complexity $= O\left((n*n)+n\right)$

$= O(n^2)$

2. Optimal Sieve

```
void  sieveofEratosthenes (int n)
{
    boolean prime[] = new boolean[n+1];

    for (int i=0; i<n; i++)                          —————— {(n-1)-0}+1 / 1 → n
    {
        prime[i] = true;
    }
    for (int p = 2; p<= sqrt(n); p++)                —————— (√n-2)+1 / 1 → √n
    {
        if (prime[p] == true)
        {
            for (int i = p*p; i<=n; i=i+p)           —————— log(log n)
            {   prime[i] = false;
            }
        }
    }
    for (int i=2; i<=n; i++)                         —————— n-2+1 / 1 → n
    {
        if (prime[i] == true)
            S.O.P (i+ " ");
    }
```

for loop 1:

$$time = \frac{(higher\ limit - low\ limit)+1}{increment}$$

$$= \frac{\{(n-1)-0\}+1}{1}$$

outer for loop:

$$time = \frac{(\sqrt{n}-2)+1}{1}$$

inner for loop:

from harmonic progression series we got,

time = log (log n)

$$\text{total time complexity} = O\left(n + \sqrt{n}\log(\log n) + n\right)$$

$$= O\left(\sqrt{n}\log(\log n)\right)$$

Comparison:

Comparing Naive approach and optimal sieve we can say that Optimal Sieve Algorithm has better time complexity because $O(\sqrt{n}\log(\log n))$ takes less time than $O(n^2)$.

Scanned with CamScanner

Recursion tree time complexity

1. $T(n) = T(n/2) + n-1$

$$T(n) = \begin{cases} 0 & n = 1 \\ T(n/2) + n-1 & n > 1 \end{cases}$$

$$T(n) = T(n/2) + n-1$$

$$T(n) = \left[T\left(\frac{n}{2^2}\right) + \frac{n}{2} - 1\right] + (n-1) \qquad \left[T(n/2) = T(n/2^2) + \frac{n}{2} - 1\right]$$

$$T(n) = T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2} - 1\right) + (n-1) \qquad \left[T(n/2^2) = T(n/2^3) + \frac{n}{2^2} - 1\right]$$

$$T(n) = \left[T\left(\frac{n}{2^3}\right) + \frac{n}{2^2} - 1\right] + \left(\frac{n}{2} - 1\right) + (n-1)$$

$$T(n) = T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2} - 1\right) + \left(\frac{n}{2} - 1\right) + (n-1)$$

$$\vdots \qquad \vdots$$

$$T(n) = T\left(\frac{n}{2^k}\right) + \left(\frac{n}{2^{k-1}} - 1\right) + \left(\frac{n}{2} - 1\right) + (n-1)$$

As our base case is, $n = 1$. So, assume that,

$$\frac{n}{2^k} = 1$$

$$\Rightarrow n = 2^k$$

$$\Rightarrow \log n = k \log 2$$

$$\Rightarrow k = \frac{\log n}{\log 2}$$

$$\therefore k = \log_2 n$$

$$T(n) = T\left(\frac{n}{2^k}\right) + (n - 2) + 2 \quad \left[\sum_{n=0}^{\infty} \left(\frac{1}{2}\right)^n = 2\right]$$

$$\Rightarrow T(n) = T(1) + 2^k - 2^{k-1} + 2$$

$$\Rightarrow T(n) = 1 + 2^{\log n} - 2^{\log n - 1} + 2$$

$$= 2^{\log n}$$

$\therefore$ time complexity $= O\left(2^{\log n}\right)$

2. $T(n) = T(n-1) + n-1$

$$T(n) = \begin{cases} 0 & n=1 \\ T(n-1) + n-1 & n>1 \end{cases}$$

$T(n) = T(n-1) + n-1$

$T(n) = [T(n-1-1) + (n-1)-1] + n-1 \qquad [T(n-1) = T(n-1-1) + (n-1)-1]$

$T(n) = T(n-2) + (n-2) + (n-1)$

$T(n) = [T(n-2-1) + (n-2)-1] + (n-2) + (n-1) \qquad [T(n-2) = T(n-2-1) + (n-2)-1]$

$T(n) = T(n-3) + (n-3) + (n-2) + (n-1)$

$\vdots \qquad \vdots$

$T(n) = T(n-k) + (n-k) + (n-(k-1)) + (n-1)$ -------- (i)

As, here the base case is $n=1$, so, assume that,

$n-k=1$

$\Rightarrow n=k$

apply $k=n$ in equation (i)

$T(n) = T(n-n) + (n-n) + (n-n+1) + \ldots (n-1)$

$T(n) = T(0) + 0 + 1 + \text{------} + (n-1)$

$T(n) = 0 + 0 + 1 + \text{------} + (n-1)$

$T(n) = \frac{n(n+1)}{2}$

$T(n) = \frac{n^2+n}{2} \qquad \therefore \text{time complexity} = O(n^2)$

Scanned with CamScanner

$$3.\ T(n) = T(n/3) + 2T(n/3) + n$$

$$T(n) = 3T(n/3) + n$$

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T(n/3) + n & n > 1 \end{cases}$$

$$T(n) = 3T(n/3) + n$$

$$T(n) = 3\left[3T\left(\frac{n}{3^2}\right) + \frac{n}{3}\right] + n \qquad \left[T\left(\frac{n}{3}\right) = 3T\left(\frac{n}{3^2}\right) + \frac{n}{3}\right]$$

$$T(n) = 3^2 T\left(\frac{n}{3^2}\right) + 3 \times \frac{n}{3} + n$$

$$T(n) = 3^2 T\left(\frac{n}{3^2}\right) + n + n$$

$$T(n) = 3^2\left[3T\left(\frac{n}{3^3}\right) + \frac{n}{3^2}\right] + 2n \qquad \left[T\left(\frac{n}{3^2}\right) = 3T\left(\frac{n}{3^3}\right) + \frac{n}{3^2}\right]$$

$$T(n) = 3^3 T\left(\frac{n}{3^3}\right) + 3n$$

$$\vdots$$

$$T(n) = 3^k T\left(\frac{n}{3^k}\right) + kn \text{ ------------- (i)}$$

As, $n = 1$ is the base case. So, Assume that,

$$\frac{n}{3^k} = 1$$

$$\Rightarrow n = 3^k \text{ ---------------- (ii)}$$

$$\Rightarrow k = \log_3 n \text{ ---------------- (iii)}$$

Scanned with CamScanner

now, putting the value of k in equation (i).

$$T(n) = 3^k T(1) + kn$$

$$\Rightarrow T(n) = n \times 1 + (\log_3 n) \times n$$

[from equation (ii), $3^k = n$
from eqation (iii), $K = \log_3 n$
from base case, $T(1) = 1$]

$$\Rightarrow T(n) = n \log_3 n$$

$\therefore$ time complexity $= O(n \log_3 n)$

(Ans.)

Scanned with CamScanner

4. $T(n) = 2T(\frac{n}{2}) + n^2$

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(n/2) + n^2 & n > 1 \end{cases}$$

$$T(n) = 2T(\frac{n}{2}) + n^2$$

$$T(n) = 2\left[2T(\frac{n}{2^2}) + (\frac{n}{2})^2\right] + n^2$$

$$T(n) = 2^2 T(n/2^2) + \frac{n^2}{2} + n^2$$

$$T(n) = 2^2\left[2T(n/2^3) + (\frac{n}{2^2})^2\right] + \frac{n^2}{2} + n^2$$

$$T(n) = 2^3 T(n/2^3) + \frac{n^2}{4} + \frac{n^2}{2} + n^2$$

$$\vdots \qquad \vdots$$

$$T(n) = 2^K T(n/2^K) + n^2(\frac{1}{4} + \frac{1}{2} + \cdots + 1)$$

As the base case $n = 1$ so, Assume that, $\frac{n}{2^K} = 1$

$$\Rightarrow n = 2^K$$

$$\Rightarrow K = \log_2 n$$

$$T(n) = 2^K T(n/2^K) + n^2(\frac{1}{4} + \frac{1}{2} + \cdots + 1)$$

$$\Rightarrow T(n) = n\,T(1) + n^2 \times 2 \quad \left[\sum_{n=0}^{\infty} (\frac{1}{2})^n = 2\right]$$

$$\Rightarrow T(n) = n + 2n^2$$

$\therefore$ time complexity $= O(n^2)$ [proved]

Scanned with CamScanner

◻ Pseudocode to coding

```
import java.util.Scanner;

public class code

{
    public static void main (String [] args)
    {
        Scanner sc = new Scanner (System.in);

         int a, n, sum, r = 0;

          n= sc.nextInt ();

           a = n;
           sum = 0;

           while (n>0)
           {
              r = n%10;
              sum= sum+ (r*r*r);

              n = n/10;

           }
         if (a == sum)
                System.out.println ("Armstrong Number");

         else
               System.out. println ("Not an Armstrong Number");

    }
}
```

Scanned with CamScanner