Assignment # 04

Aimen Yaseen

Question 2 - TIME COMPLEXITY (Each part has 1 point) (Estimated Time 100 mins)

If you are having any difficulty in finding out the complexity of the codes - read this document. It has many practice problems regarding time complexity with their solution mentioned in the comments. Also if you need further explanation(of the above sample practice problems), please watch my following VIDEO-LECTURE, which I delivered in my discrete mathematics course.

(35*2 = 70 Points)

1) What is the algorithm's complexity of the following piece of code - Sample Solution is in RED

```
int Sum=0;                    // O(1) Time
for(int i=0; i<N; i++)        //(1+1+1+...+1 - - - N Times =O(N)
 for(int j=0; j<N; j++)  Sum++;
 // (1+1+1+...+1) + (1+1 +... +1)+... + (1+1 +... +1) added N times
   // N          +      N    +... +      N          = O(N^2)
Overall Complexity : O(1) +O(N) + O(N^2) + O(N^2) = O(N^2)
```

2) What is the algorithm's complexity of the following piece of code

```
int Sum=0;                 # O(1)
for(int i=0; i<N; i++)     # (1+1+ ... +1) = O(N)
 Sum++;                    O(N)
for(int j=0; j<N; j++)     # O(N)
 Sum++;
```

$T_N = \boxed{O(N)}$

3) What is the algorithm's complexity of the following piece of code

```
int Sum=0;                   # O(1)
for(int i=0; i<N; i++)       # O(N)
 for(int j=0; j<N; j++)      # N+N+... +N = N.N = O(N^2)
  for(int k=0; k<N; k++)     # N+N+...+N = N^2.N = O(N^3)
   Sum++;

for(int i=0; i<N; i++)       # O(N)
 for(int j=0; j<N; j++)      # O(N^2)
  for(int k=0; k<N; k++)     # O(N^3)
   Sum++;
```

$T_N = 2[O(N) + O(N^2) + O(N^3)] = \boxed{O(N^3)}$

4) What is the algorithm's complexity of the following piece of code

```
int Sum=0;                 # O(1)
for(int i=0; i<N; i++)     # O(N)
 Sum++;
for(int j=0; j<N; j++)     # O(N)
 Sum++;
for(int k=0; k<N; k++)     # O(N)
 Sum++;
for(int m=0; m<N; m++)     # O(N)
 Sum++;
for(int n=0; n<N; n++)     # O(N)
 Sum++;
for(int p=0; p<N; p++)     # O(N)
 Sum++;
```

$T_N = \cancel{O(6N)}\ O(6N) = \boxed{O(N)}$

5)

```
int Sum=0;                   # O(1)
for(int i=0; i<N; i++)       # O(N)                    N.N
 for(int j=0; j<i; j++)      # 1+2+3+... +N = O(N^2)
  for(int k=0; k<j; k++)     # 1+2+3+...+N = N.N^2
   Sum++;                                   O(N^3)
```

$T_n = O(N) + O(N^2) + O(N^3) = \boxed{O(N^3)}$

6)

```
int Sum=0;                     # O(1)
for(int i=0; i<N; i+=2)        # O(N/2)                     = N^2
 for(int j=0; j<i; j+=2)       # 1+2+3+ - - +N/2 = N^2/4
  for(int k=0; k<j; k+=2)      # N/2 . N^2/4
   Sum++;
```

$T_n = O(N/2) + O(N^2/4) + O(\frac{N^3}{8}) = \boxed{O(N^3)}$

7

```
int Sum=0;
for(int i=1; i<N; i*=2)    # log_2 N
 for(int j=1; j<N; j*=2)   # N+N+...+N = N log_2 N
    Sum++;
```

$T_N = O(\log_2 N) + O(\cancel{\log} N\log_2 N) = \boxed{O(N\log_2 N)}$

8

```
int Sum=0;                 # O(1)
for(int i=1; i<N; i*=2)    # log_2 N
 Sum++;
for(int j=1; j<N; j*=2)    # log_2 N
 Sum++;
```

$T_N = O(2\log_2 N) = \boxed{O(\log_2 N)}$

9

```
for(int i=1; i<=N*N; i+=2)    # N^2/2
  for(int j=1; j<N*N; j*=2)   # log_2 N^2 = (2 log_2 N)(N^2/2)
    Sum++;
```

$T_N = N^2/2 + 2N^2/2 \log_2 N = \boxed{O(N^2\log_2 N)}$

10

```
for(int i=1; i<=N*N; i+=2)    # N^2/2
  Sum++;
for(int j=1; j<N*N; j*=2)     # log_2 N^2 = 2 log N
    Sum++;
```

$T_N = N^2/2 + 2\log N = \boxed{O(N^2)}$

11

```
for(int i=1; i<=N*N; i*=2)    # log_2 N^2
  for(int j=1; j<N*N; j*=2)   # (log_2 N^2 + log_2 N^2 + ... + log_2 N^2) = log_2^2 N^2
    Sum++;
```

$T_N = 2\log N + 2\log^2 N = \boxed{O(\log^2 N)}$

12

```
for(int i=1; i<=N*N; i*=2)    # log_2 N^2 = 2 log N
    Sum++;
for(int j=1; j<N*N; j*=2)     # log_2 N^2 = 2 log N
    Sum++;
```

$T_N = 4\log N = \boxed{O(\log N)}$

13

```
int Sum=0;
for(int i=1; i<=N; i*=2)        # log_2 N
 for(int j=1; j<=N; j*=2)       # log_2 N . log_2 N = log_2^2 N
   for(int k=1; k<=N; k*=2)     # log_2^2 N (log_2 N) = log_2^3 N
     Sum++;
```

$T_n = O(\log N) + O(\cancel{2}\log N) + O(\cancel{3}\log^3 N) = \boxed{O(\log^3 N)}$

14

```
int Sum=0;
for(int i=1; i<=N; i*=2)    # log_2 N
Sum++;
for(int j=1; j<=N; j*=2)    # log_2 N
    Sum++;
for(int k=1; k<=N; k*=2)    # log_2 N
    Sum++;
```

$T_N = 3\log_2 N = \boxed{O(\log_2 N)}$

15

```
int sum,i,j;
sum = 0;
for (i=1;i<n;i=i*2)     # log_2 N
{
  for (j=0;j<n;++j)     # N log_2 N
  {
     sum++;
  }
}
```

$T_N = \log_2 N + N\log_2 N = \boxed{O(N\log_2 N)}$

16

BE CAREFUL GEOMETRIC SERIES

```
int sum,i,j;
sum = 0;
for (i=1; i<n; i=i*2)     # log_2 N
{
  for (j=0; j<i; ++j)     # 1+2+4+8+ ... N <= 2N-1
  {
     sum++;
  }
}
```

$T_N = O(\log_2 N) + O(2N-1) = \boxed{O(N)}$

17

BE CAREFUL GEOMETRIC SERIES

```
int sum,i,j;
sum = 0;
for (i=1; i<n; i=i*5)      # O(log_5 n)
{
  for (j=0; j<i; j+=2)     # (1+5+25+...+N)/2 ≤ 2N/2
  {
      sum++;
  }
}
```

$T_N = O(\log_5 n) + O(N) = \boxed{O(N)}$

18

```
int sum,i,j;
sum = 0;
for (i=1; i<n; i=i*4)      # O(log_4 n)
{
  for (j=0 ; j<n ; j+=3)   # (n/3 + n/3 + ... + n/3) log_2 n
  {
      sum++;
  }
}
```

$O(n/3 \log_2 n)$

$T_N = O(\log_2 n) + O(n/3 \log_2 n) = \boxed{O(n \log_2 n)}$

19 What will be the output (the value of Sum) of the program asymptotically in BIG-O notation, I am not asking here the complexity of loop rather the asymptotic bound on the value of Sum:

```
int Sum = 0;
for(int i=1; i<=n; i+=1)   # O(n)
{
 Sum+=i;   # 1+2+3+4+...+n = O(n^2)
}
cout<<Sum<<endl;
```

$T_{sum} = \boxed{O(n^2)}$

20 What will be the output(the value of Sum) of the program asymptotically in BIG-O notation:

```
int Sum = 0;
for(int i=1; i<=n; i*=2)   # log_2 n
{
 Sum+=i;   # 1+2+4+8+16+...+n ≤ 2N
}
cout<<Sum<<endl;
```

$T_{sum} = O(2N) = \boxed{O(N)}$

21 What is the time complexity of the algorithm:

```
int Sum = 0;
for(int i=1; i<=n; i+=1)   # O(N)
{
 for(int j=1; j<=i; j++)   # 1+2+3+...+N = O(N^2)
 {
    Sum++;
 }
}
cout<<Sum<<endl;
```

$T_N = O(N) + O(N^2) = \boxed{O(N^2)}$

22 What is the time complexity of the algorithm:

```
int Sum = 0;   # O(1)
for(int i=1; i<n; i*=2)    # O(log_2 N)
{
 for(int j=1; j<=i; j++)   # 1+2+4+...+N ≤ 2N
 {
    Sum++;
 }
}
cout<<Sum<<endl;
```

$T_N = O(\log N) + O(2N) = \boxed{O(N)}$

40 Complexity of primeNumber function.

```
int sqrt(int N)
{
 int d;
 for(d=0; d*d<=N; d++) { }   # √N
 return d-1;   -> √N - 1 = √N
}
bool primeNumber(int n)
{
     bool isPrime = true;
     int lmt = (sqrt(n));     # √N
     for (int d=2; d <=lmt ;++d)    # O(√N)
```

41 Complexity of primeNumber function.

```
int sqrt(int N)
{
 int d;
 for(d=0; d*d<=N; d++){ }   # √N
 return d-1;   -> √N - 1
}
bool primeNumber(int n)
{
     bool isPrime = true;    -> √N · √N
     for (int d=2; d <= sqrt(n) ;++d)    = (N)
     {    √N + √N + √N = √N(√N)
```

```
{
        if (n%d==0)
                return false;
}
return true;
}
```

$T_N = \boxed{O(\sqrt{N})}$

```
        if (n%d==0)
                return false;
    }
    return true;
}
```

$T_N = \boxed{O(N)}$

23 What is the time complexity of the algorithm:

```
int f1(int n)
{
    int K=0;
    for(int j=0; j*j<=n*n; j++)  K++;     # O(N)
    return K; -> N
}
int main()
{
    int Sum = 0,  n;
    cin>>n;
    for(int i=1; i<=f1(n); i+=1)   # N+N+...+N = N.N = N^2
      for(int j=1; j<=i; j++)   Sum++;   #  1+2+3+...+N = N^2
    cout<<Sum<<endl;
}
```

$T_N = O(N^2) + O(N^2) = O(2N^2)$
$= \boxed{O(N^2)}$

24 What is the time complexity of the algorithm:

```
int f1(int n)
{
  int K=0;                             # log_2 √N = 1/2 log N
  for(int j=1; j*j<=n; j*=2)    K++;               = log N
  return K;   -> log N
}
int main()
{
int Sum = 0;
int n;                              # log N + log N + log N + ... + log N
cin>>n;                               log N(log N) = log^2 N
for(int i=1; i<=f1(n); i+=1)
    for(int j=1; j<=i; j++)         Sum++;
cout<<Sum<<endl;    # 1+2+3+ ... + log N = log^2 N
}
```

$T_N = O(\log^2 N) + O(\log^2 N) = \boxed{O(\log^2 N)}$

25

What is the time complexity of the algorithm:

```
int f1(int n)
{        int K=0;
  for(int j=1; j*j<=n; j++)   # √N
   K++;

 return K*K;  -> √N.√N = N
}
int main()
{
int Sum = 0;
int n;
cin>>n;
int Terminator = f1(n);    # O(1)
for(int i=1; i<= Terminator; i+=1)  -> O(N)
{
 for(int j=1; j<=i; j++)   #  1+2+3+ ... + N = N^2
  {
    Sum++;
  }
}
```

$T_N = O(N) + O(N^2) = \boxed{O(N^2)}$

26

What is the time complexity of the algorithm:

```
int f1(int n)
{        int K=0;
  for(int j=0; j*j<=n; j++)   #  √N
   K++;

 return K;   √N
}
int main()
{
int Sum = 0;
int n;
cin>>n;
int Terminator = f1(n);   #  O(1)
for(int i=1; i<=Terminator; i+=1)    # O(√N)
{
 for(int j=1; j<=i; j++)   #  1+2+3+ ... + √N = (√N)^2 = N
 {
    Sum++;
}
```

$T_N = O(N) + O(\sqrt{N}) = \boxed{O(N)}$

```
cout<<Sum<<endl;
}
```

```
}
cout<<Sum<<endl;
}
```

27

```
for (i=1;i<n;i=i*4)       # O(log_4 N)
{
    cout << i;
    for (j=0;j<n;j=j+2)   #O(N/2 log_4 N)
    {
        cout << j;
        sum++
    }
    cout << sum;
}
```

$\boxed{O(N \log N)}$

$T_N = O(\log_4 N) + O(N/2 \log_4 N) =$

28

```
for (i=1;i<n;i=i*4)       # log_4 N
{
    cout << i;
    for (j=0;j<i; j=j+2)  # (1+4+16+...+N ≤ 2N)/2
    {
        cout << j;
        sum++
    }
    cout << sum;
}
```

$T_N = O(\log N) + O(2N/2)$

$= \boxed{O(N)}$

29

```
for (i=1;i<=n*n;++i)      # O(N^2)
{
    cout << i;
    Sum=0;
    for (j=1; j<=i; ++j)  # 1+2+3+...+N^2 = O(N^4)
    {
        Sum++;
        cout << i;
    }
    cout << Sum;
}
```

$T_N = O(N^2) + O(N^4)$

$= \boxed{O(N^4)}$

30

```
for (i=1;i<=n*n;++i)      # O(n^3)
{
    cout << i;
    Sum=0;
    for (j=1; j<=i; ++j)  # 1+2+3+...+n^3 = O(n^6)
    {
        Sum++;
        cout << i;
    }
    cout << Sum;
}
```

$T_n = O(n^3) + O(n^6)$

$= \boxed{O(n^6)}$

31

```
for (i=1;i<=n*n*n; i*=2)  # O(log_2 n^3)
{                         O(3 log_2 n) =
                          O(log n)
    cout << i;
    Sum=0;
    for (j=1;j<=i; j++)   1+2+4+8+...+n^3 ≤ 2n^3
    {
        Sum++;
        cout << i;
    }
    cout << Sum;
}
```

$T_N = O(\log n) + O(2n^3)$

$= \boxed{O(n^3)}$

32

```
for (i=1;i<=n*n*n; i*=2)  # O(log_2 n^3)
{                         O(3 log_2 n) = O(log n)
    cout << i;
    Sum=0;
    for (j=1;j<=n; j++)   # ~~log_2 n^3 · log_2 n^3 = O(log^2 n^3) = O(3 log^2 n) = O(log n)~~
    {
        Sum++;
        cout << i;
    }
    for (k=1;k<=n; k++)   # ~~log_2 n^3 · log_2 n^3 = O(log^2 n)~~
    {
        Sum++;
        cout << i;
    }
    cout << Sum;          ~~T_n = O(log n) + O(log^2 n) = O(log^2 n)~~
}
```

$\# \ O(n \log n)$

$\# \ O(n \log n)$

$T_n = O(\log n) + O(n \log n) + O(n \log n)$

$= \boxed{O(n \log n)}$

Scanned with CamScanner

33

```
for (i=1;i<=n*n*n; i*=2)
{
    cout << i;
    Sum=0;
    for (j=1;j<=i; j++)
    {
        Sum++;
        cout << i;
    }

    for (j=1;j<=n; j*=2)
    {
        Sum++;
        cout << i;
    }

    cout << Sum;
}
```

$O(\log_2 n^3) = O(\log n)$

$1+2+4+8+\cdots+n^3 \le 2n^3$

$O(\log n).(O\log n^3)$ $O(\log^2 n)$

$Tn = O(\log n) + O(2n^3) + O(\log^2 n) = \boxed{O(n^3)}$

34

```
for (i=1;i<=n*n*n; i*=2)
{
    cout << i;
    Sum=0;
    for (j=1;j<=i; j++)
    {
        Sum++;
        cout << i;
    }

    for (j=1;j<=n; j++)
    {
        Sum++;
        cout << i;
    }

    cout << Sum;
}
```

$O(\log_2 n^3) = O(\log n)$

$2n^3$ $1+2+4+\cdots+n^3$

$O(n).O(\log n)$

$Tn = O(\log n) + O(2n^3) + O(n\log n) = \boxed{O(n^3)}$

35-36

```
for (int i=1; i <= n ; i = i * 2)
{
    for ( j = 1 ; j <= i ; j = j * 2)
        cout<<"*";
}
```

#$O(\log_2 n)$

$1+2+4+\cdots+\log_2 n = O(\log^2 n)$

$Tn = O(\log n)+O(\log^2 n)$ ~~$+O(\log^2 n)$~~ $\boxed{O(\log^2 n)}$

```
for (int i=1; i <= n ; i = i * 2)
    for ( j = 1 ; j <= i ; j = j * 2)
        cout<<"*";
```

$O(\log n)$

$O(\log^2 n)$

```
for (int i=1; i <= n ; i = i * 2)
    for ( j = 1 ; j <= i ; j = j * 2)
        cout<<"*";
```

$O(\log^2 n)$

$Tn = O(\log^2 n) + O(\log^2 n)$ $\boxed{O(\log^2 n)}$

37

```
for (i=0; i<n; i=i+3)
{
    cout << i;
    for (j=1; j<n; j=j*3)
    {
        cout << j;
        sum++
    }
    for (k=1;k<n;k=k*3)
    {
        cout << j;
        sum++
    }

    cout << sum;
```

~~$O(\log n)$~~ $O(n/3)$

#$O(n.\log_3 n)$

$O(n.\log_3 n)$

$Tn = O(n/3) + O(n\log_3 n) + O(n\log_3 n) = \boxed{O(n\log n)}$

Scanned with CamScanner

38

```
for (int i=1; i <= n ; i = i * 2)     # O(log n)
{
      for ( j = 1 ; j <= i ; j = j * 2)     # O(log n) .
      {
            cout<<"*";
      }
}
```

$O(\log n) = O(\log^2 n)$

```
for(int i=0; i<=N; i++)     # O(n)
{
 Sum++;
}
```

$Tn = O(\log n) + O(\log^2 n) + O(n)$

$= \boxed{O(n)}$

39

```
for (i=0; i<n; i=i+3)     # O(n/3)
{
      cout << i;
      for (j=1; j<n; j=j*3)     # O(n log_3 n)
      {
            sum++
      }
}
for (k=1;k<n;k=k*3)     # O(log_3 n)
{
      cout << j;
      sum++
}
cout << sum;
```

$Tn = O(n/3) + O(n \log_3 n) + O(\log_3 n)$

$= \boxed{O(n \log n)}$

40. Challenge: Make a question of your own and ask your fellows (on whatsapp group, what is the time complexity of the problem, of course make that question which you know the answer of :))

```
O===O
 \O/
  |
 /\
```

Happy Coding... :)

Scanned with CamScanner