# **SOME USEFUL INFORMATIONS (you must understand these bounds first. They are the crux of Time bounding)**

**Note: Print this assignment and solve on it. Make table for each loop.**

**Arithmetic Sequence Size**1, 2, 3, 4, 5, 6, .... , N O(N)  
1, 3, 5, 7, .... , N N/2 i.e O(N)  
1, 4, 7, 10, .... , N N/3 i.e. O(N)  
**1, 1+k, 1+2k, 1+3k, 1+4k, 1+5k, .... , N N/k i.e. O(N) if k is a constant  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Arithmetic Series** Applications of **1+2+3+4+...+N =**

1+2+3+4+5+6+ ....N-3+ N-2+ N-1+ N O(N2)  
1+2+3+4+5+6+ ....(N/2-3)+ (N/2-2)+ (N/2-1)+ N/2 O(N2)  
1+2+3+4+5+6+ ....(N/3-3)+ (N/3-2)+ (N/3-1)+ N/3 O(N2)  
1+2+3+4+5+6+ .... + <= O(()2) O(N)  
1+2+3+4+5+6+ .... + N2 O(N4)  
1+2+3+4+5+6+ .... + N3 O(N6)  
**1+2+3+4+5+6+ .... +Nk <= O(NkxNk)**1+22+32+42+52+62+ .... + N2 O(N3)  
1+23+33+43+53+63+ .... + N3 O(N4)  
**1k+2k+3k+4k+5k+6k+ .... +Nk <= O(Nk+1)  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Geometric Sequence Size**N, N/2, N/4, N/8, N/24, N/25 , N/26, …8, 4 , 2 , 1 <=   
  
N, N/3, N/9, N/27, N/34, N/35 , N/36, … , 33, 9, 3 , 1 <=   
  
N, N/5, N/25, N/125, N/54, N/55 , N/56, …, 53, 52, 5 , 1 <=   
  
**N, N/k, N/k2, N/k3, N/k4, N/k5 , N/k6, … , k3, k2, k, 1 <=   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Application

O(log N)

**for(int i=1; i\*i<=N; i++)** Complexity of this loop is **O(**)  
 **Sum++;   
N x N = N2 for(int i=1; i\*i<=N\*N; i++)** Complexity of this loop is **O(**)  
 **Sum++;**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
GEOMETRIC SERIES**

O(N)

**N < 1+2+4+8+16+32+... +N/4+N/2+N < 2N**N < 1+3+9+34+35+... +N/32+N/3+N < 2N  
  
N < 1+5+52+53+54+55+... +N/52+N/5+N < 2N

# **TIME COMPLEXITY**

|  |  |
| --- | --- |
| 1) What is the algorithm’s complexity of the following piece of code  int Sum=0; for(int i=0; i<N; i++)   for(int j=0; j<N; j++)  Sum++;  **Total:** | 2) What is the algorithm’s complexity of the following piece of code  int Sum=0; for(int i=0; i<N; i++)   Sum++;  for(int j=0; j<N; j++)  Sum++;  **Total:** |
| 3)  int Sum=0; for(int i=0; i<10; i++)   for(int j=0; j<20; j++)   for(int k=0; k<30; k++)  Sum++; | 4)  int Sum=0; for(int i=0; i<10; i++)   for(int j=0; j<N; j++)   for(int k=0; k<N; k++)  Sum++; |
| 5)  int Sum=0; for(int i=0; i<N; i++)   for(int j=0; j<N; j++)   for(int k=0; k<N; k++)  Sum++;    for(int i=0; i<N; i++)   for(int j=0; j<N; j++)   for(int k=0; k<N; k++)  Sum++;    **Total =** | 6)  int Sum=0; for(int i=0; i<N; i++)   Sum++;  for(int j=0; j<N; j++)  Sum++; for(int k=0; k<N; k++)  Sum++; for(int m=0; m<N; m++)   Sum++;  for(int n=0; n<N; n++)  Sum++; for(int p=0; p<N; p++)  Sum++;  **Total =** |
| 7)  int Sum=0; for(int i=0; i<N; i++)   for(int j=0; j<i; j++)   for(int k=0; k<j; k++)  Sum++;  **Total =** | 8)  int Sum=0; for(int i=0; i<N; i+=2)   for(int j=0; j<i; j+=2)   for(int k=0; k<j; k+=2)  Sum++;  **Total =** |
| 9)  int Sum=0; for(int i=1; i<N; i\*=2)   Sum++; for(int j=1; j<N; j\*=2)   Sum++; | 10)  int Sum=0; for(int i=1; i<N; i\*=2)   for(int j=1; j<N; j\*=2)   Sum++; |
| 11) for(int i=1; i<=N\*N; **i+=2**)   for(int j=1; j<N\*N; j\*=2)   Sum++; | 12) for(int i=1; i<=N\*N; **i+=2)**   Sum++;  for(int j=1; j<N\*N; j\*=2)   Sum++; |
| 13) for(int i=1; i<=N\*N; **i\*=2)**   for(int j=1; j<N\*N; j\*=2)   Sum++; | 14) for(int i=1; i<=N\*N; i\*=2)   Sum++;  for(int j=1; j<N\*N; j\*=2)   Sum++; |
| 15)  int Sum=0; for(int i=1; i<=N; i\*=2)   for(int j=1; j<=N; j\*=2)   for(int k=1; k<=N; k\*=2)   Sum++; | 16) int Sum=0; for(int i=1; i<=N; i\*=2)   Sum++; for(int j=1; j<=N; j\*=2)   Sum++; for(int k=1; k<=N; k\*=2)   Sum++; |
| 17) int sum, i, j;  sum = 0;  for (i=1; i<n; i=i\*2)  {   for (j=0; j<n;++j)  {   sum++;  }  } | 18) **BE CAREFUL GEOMETRIC SERIES** int sum,i,j;  sum = 0;  for (i=1; i<n; i=i\*2)  {   for (j=0; j <i ; ++j)  {   sum++;  }  } |
| 19) **BE CAREFUL GEOMETRIC SERIES** int sum,i,j;  sum = 0;  for (i=1; i<n; i=i\*5)  {   for (j=0; j<i; j+=2)  {   sum++;  }  } | 20) int sum,i,j;  sum = 0;  for (i=1; i<n; i=i\*4)  {   for (j=0 ; j<n ; j+=3)  {   sum++;  }  } |
| 21 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)  {  Sum+=i;  } cout<<Sum<<endl; | 22 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)  {  Sum+=i; } cout<<Sum<<endl; |
| 23) What is the time complexity of the algorithm: int Sum = 0; for(int i=1; i<=n; i+=1)  {  for(int j=1; j<=i; j++)   {  Sum++;  } } cout<<Sum<<endl; | 24) What is the time complexity of the algorithm: int Sum = 0; for(int i=1; i<n; i\*=2)  {  for(int j=1; j<=i; j++)   {  Sum++;  } } cout<<Sum<<endl; |
| 25) What is the time complexity of the algorithm: void f1(int n) {  for(int j=0**; j\*j<=n\*n;** j++)  K++;  return K;  } int main() { int Sum = 0; int n; cin>>n;  for(int i=1; **i<=f1(n);** i+=1)  {  for(int j=1; j<=i; j++)   {  Sum++;  } } cout<<Sum<<endl; } | 26) What is the time complexity of the algorithm: void f1(int n) {  for(int j=1; **j\*j<=n;** j\*=2)  K++;  return K;  } int main() { int Sum = 0; int n; cin>>n; for(int i=1; i<=f1(n); i+=1)  {  for(int j=1; j<=i; j++)   {  Sum++;  } } cout<<Sum<<endl; } |
| 27)  What is the time complexity of the algorithm:   void f1(int n) {  for(int j=1**; j\*j<=n;** j++)  K++;    return **K\*K;**  } int main() { int Sum = 0; int n; cin>>n;  int **Terminator** **= f1(n);** for(int i=1; i<= **Terminator;** i+=1)  {  for(int j=1; j<=i; j++)   {  Sum++;  } } cout<<Sum<<endl; } | 28)  What is the time complexity of the algorithm:  void f1(int n) {  for(int j=0; **j\*j<=n;** j++)  K++;    return K;  } int main() { int Sum = 0; int n; cin>>n; int **Terminator** **= f1(n);** for(int i=1; i<=**Terminator**; i+=1)  {  for(int j=1; j<=i; j++)   {  Sum++;  } } cout<<Sum<<endl; } |
| 29)  for (i=1;i<n;i=i\*4)  {  cout << i;  for (j=0;j<n;j=j+2)  {  cout << j;  sum++  }  cout << sum;  } | 30)  for (i=1;i<n;i=i\*4)  {  cout << i;  for (j=0;j<i; j=j+2)  {  cout << j;  sum++  }  cout << sum;  } |
| 31) for (i=1; i<=n\*n; ++i)  {   cout << i;  Sum=0;  for (j=1; j<=i; ++j)  {  Sum++;  cout << i;  }  cout << Sum;  } | 32) for (i=1; i<=n\*n\*n; ++i)  {   cout << i;  Sum=0;  for (j=1; j<=i; ++j)  {  Sum++;  cout << i;  }  cout << Sum;  } |
| 33) for (i=1; i<=n\*n\*n; i\*=2)  {   cout << i;  Sum=0;  for (j=1;j<=i; j++)  {  Sum++;  cout << i;  }  cout << Sum;  } | 34) for (i=1; i<=n\*n\*n; i\*=2)  {   cout << i;  Sum=0;  for (j=1;j<=n; j++)  {  Sum++;  cout << i;  } for (k=1; k<=n; k++)  {  Sum++;  cout << i;  }  cout << Sum;  } |
| 35) 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;  } | 36) 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;  } |
| 37-38   |  | | --- | | for (int i=1; i <= n ; i = i \* 2)  {  for ( j = 1 ; j <= i ; j = j \* 2)  cout<<”\*”;  } | | for (int i=1; i <= n ; i = i \* 2)  for ( j = 1 ; j <= i ; j = j \* 2)  cout<<”\*”;  for (int i=1; i <= n ; i = i \* 2)  for ( j = 1 ; j <= i ; j = j \* 2)  cout<<”\*”; | | 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;  } |
| 39) for (int i=1; i <= n ; i = i \* 2)  {  for ( j = 1 ; j <= i ; j = j \* 2)  {  cout<<”\*”;  }  }  for(int i=0; i<=N; i++)  {  Sum++; } | 40) for (i=0; i<n; i=i+3)  {  cout << i;  for (j=1; j<n; j=j\*3)  {  sum++  }  }  for (k=1; k<n; k=k\*3)  {  cout << j;  sum++  }  cout << sum; |
| 41) Complexity of **primeNumber function.**  int sqrt(int N) {  int d;  **for(d=0; d\*d<=N; d++)** { }   return d-1; } bool **primeNumber**(int n)  {  bool isPrime = true;  int lmt = **(sqrt(n));**  for (int d=2; d <=lmt ;++d)  {  if (n%d==0)  return false;  }  return true; } | 42) Complexity of **primeNumber function.** int sqrt(int N) {  int d;  **for(d=0; d\*d<=N; d++)**{ }   return d-1; } bool **primeNumber**(int n)  {  bool isPrime = true;  for (int d=2; d <= **sqrt(n)** ;++d)  {  if (n%d==0)  return false;  }  return true;  } |
|  |  |