**Complexity Analysis – Practice questions**

*Data Structures (Section A & D)*

**Question 1.** Write the tight big-O for the following expressions and **find c and n0**

1. **9n3+400n2**

# Solution:

The order of T(n) is O(n3)

**Proof:**

Since,

T(n) <= cf(n)

9n3 +400n2<= c(n3) ----------------------------------------------------------------------- (1)

(9n3 +400n2)/(n3) <= c

Put n=2:

(9\*23 +400\*22)/(23) <= c

(72 +1600)/(8) <= c

1672/8 <=c

209 <= c

c >= 209

Putting c=209 and n=2 in Eq(1):

9\*23 +400\*22 <= 209(23)

1672<= 1672 (hold true)

C=209 n0=2

Hence,proved.

1. **n32n + n23n**

# Solution:

The order of T(n) is O(n23n)

**Proof:**

Since,

T(n) <= cf(n)

n32n+n23n<= c(n23n) ----------------------------------------------------------------------- (1)

(n32n+n23n)/(n23n) <= c

Put n=2:

(2322+2232)/(2232) <= c

(32 +36)/36) <= c

68/36 <=c

1.89 <= c

c >= 1.89 ~2

Putting c=2 and n=2 in Eq(1):

2322+2232<= 2(2232)

68<= 72 (hold true)

The above equation holds true for all n>=1.

C=2 n0=1

Hence,proved.

1. **n2/logn + n**

# Solution:

The order of T(n) is O(n2/logn)

**Proof:**

Since,

T(n) <= cf(n)

n2/logn + n<= c(n2/logn) ----------------------------------------------------------------------- (1)

(n2/logn + n)/( n2/logn) <= c

Put n=2:

(22/log2 + 2)/( 22/log2) <= c

15.29/13.29 <= c

1.15 <=c

1.15 <= c

c >= 1.15 ~2

Putting c=2 and n=2 in Eq(1):

(22/log2 + 2) <= 2(22/log2)

15.29<= 26.58 (hold true)

C=2 n0=2

Hence,proved.

1. **nk+a+2n**

# Solution:

The order of T(n) is O(2n)

**Proof:**

Since,

T(n) <= cf(n)

nk+a+2n <= c(2n) --------------------------------------------------------------- (1)

(nk+a+2n) /2n <= c

Put n=2;

(2k+a+22) /22 <= c

(2k+a-2+1) <= c

Putting c=2k+a-2+1 and n=2 in Eq. (1):

(2k+a+22) <= 2k+a-2 +1\*(22)

(2k+a+22) <= 2k+a-2+2 +22

(2k+a+22) <= 2k+a +22 (holds true)

C=(2k+a-2+1) n0=2

Hence,proved.

1. **nk+a+nk log n**

# Solution:

The order of T(n) is O(nk+a)

**Proof:**

Since,

T(n) <= cf(n)

nk+a+nklogn <= c(nk+a) --------------------------------------------------------------- (1)

(nk+a+nklogn) / nk+a <= c

Put n=2;

(2k+a+2klog2) / 2k+a <= c

(2k+a-k-a+2k-k-alog2) <= c

(1+2-alog2) <= c

Putting c=(1+2-alog2) and n=2 in Eq. (1):

2k+a+2klog2 <= (1+2-alog2)(2k+a)

2k+a+2klog2 <= (2k+a +(2-alog2)\* 2k+a)

2k+a+2klog2 <= 2k+a +2klog2 (holds true)

C=(1+2-alog2) n0=2

Hence,proved.

1. **5n3+logn+ n\*(2n+n log n)**

# Solution:

The order of T(n) is O(n3)

**Proof:**

Since,

T(n) <= cf(n)

5n3 + n1/2 \*logn + 2n2 + n2logn <= c(n3) ------------------------------- (1)

(5n3 + n1/2 \*logn + 2n2 + n2logn) / n3<= c

Put n=2;

(40+1.41 \*1 + 8 + 4\*1)/8 <= c

6.67 <=c

c>= 6.67 ~7

Putting c=9 and n=2 in Eq.(1):

40+ (2)^1/2 + 8 + 4 <= 9\*8

53.41 <= 72 (holds true)

The above equation holds true for all n>=1.

C=9 n0=1

Hence,proved.

1. **(100n+logn) \* (25n+log n)**

# Solution:

T(n) = (100n + logn) \* (25n + logn)

T(n) = 2500n2 + 100nlogn + 25nlogn + (logn)2

The order of T(n) is O(n2).

**Proof:**

Since,

T(n) <= cf(n)

2500n2 + 100nlogn + 25nlogn + (logn)2 <= c(n2) --------------------------------------------------------------- (1)

(2500n2 + 100nlogn + 25nlogn + (logn)2) /n2<= c

Put n=2;

(2500\*4 + 100\*2\*1 + 25\*2\*1 +1)/4 <= c

2562.5 <= c

c >= 2563

Putting c=2563 and n=2 in Eq. (1):

(2500\*4 + 100\*2\*1 + 25\*2\*1 +1) <= 2563 \* 4

10251 <= 10252 (holds true)

The above equation holds true for all n>=1.

C=2563 n0=1

Hence,proved.

1. **n3+ 4n+ 2*n***

# Solution:

**Proof:**

Since,

T(n) <= cf(n)

n3+ 4n+ 2n <= c(2n) ----------------------------------------------------------------------- (1)

(n3+ 4n+ 2n)/( 2n) <= c

Put n=2:

(23+ 4\*2+ 22)/( 22) <= c

28/4 <=c

7 <= c

c >= 7

Putting c=7 and n=2 in Eq(1):

(23+ 4\*2+ 22) <= 7( 22)

28<= 28 (hold true)

The above equation holds true for all n>=1.

C=7 n0=1

Hence,proved.

**Question 2. For each of the following program fragments give an analysis of the running time in *T(N)* and as well as in *tight* *Big-O*.**

**TO DO:** Dry run the code for different values of N in rough before estimating. Assume cost of cout<< is 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| for (int i=1; i <= n ; i = i \* 2)  { for ( j = 1 ; j <= i ; j = j \* 2)  { cout<<”\*”;  }  } Solution:  |  |  |  |  | | --- | --- | --- | --- | | **n** | **i** | **possible values of j** | **Frequency of cout stmt** | | 1 | 1 | 1 | 1 | | 2 | 1,2 | 1, 1,2 | 1+2 =3 | | 4 | 1,2,4 | 1, 1,2, 1,2,4 | 1+2+3=6 | | 8 | 1,2,4,8 | 1, 1,2, 1,2,4, 1,2,4,8 | 1+2+3+4=10 | | 16 | 1,2,4,8,16 | 1 ,1,2, 1,2,4, 1,2,4,8, 1,2,4,8,16 | 1+2+3+4+5=15 | | 32 | 1,2,4,8,16,32 | 1, 1,2, 1,2,4, 1,2,4,8, 1,2,4,8,16, 1,2,4,8,16,32 | 1+2+3+4+5+6=21 | | n | 1,2,4,8,…,n | 1, 1,2, 1,2,4, ….  1,2,4,8…(n) | 1+2+3…+() =  ()(2 + )/2 |  |  |  | | --- | --- | | Steps | Step Counts | | Int i=0 | 1 | | i <= n | + 2 | | i = i \* 2 | + 1 | | j = 1 | + 1 | | j <= i | ()(2 + )/2 + + 1 | | j = j \* 2 | ()(2 + )/2 | | Cout << ”\*” | ()(2 + )/2 |   T(n) = sum of all terms in the last column = (compute this yourself)  T(n) = O(()2) |
| for (i=n; i>1; i=i\4){  cout << i;  for (j=0; j<n; j=j+2)  sum++  } Solution:  |  |  | | --- | --- | | **Statement** | **Number of times executed** | | I=n | 1 | | i>1 | log4n+1 | | I=i\4 | log4n | | cout<<i | log4n | | j=0 | log4n | | J<n | log4n(n/2+1) = log4n(n/2) + log4n | | J=j+2 | log4n(n/2) | | sum++ | log4n(n/2) | | **Total** | **5 log4n +3 log4n(n/2) +2** | |  | **T(n) = 5 log4n +3 log4n(n/2) +2, T(n) = O( log4n(n/2))** | |
| int sum, i, j;  sum = 0;  for (i=n;i>=1;i=i-3)  for (j=n;j>0;j--)  sum++; Solution:  |  |  | | --- | --- | | **Statement** | **Number of times executed** | | sum=0 | 1 | | i=n | 1 | | i>=1 | n/3+1 | | i=i-3 | n/3 | | j=n | n/3 | | j>0 | n/3(n+1) =n2/3+n/3 | | j-- | n/3(n) =n2/3 | | sum++ | n/3(n) =n2/3 | | **Total** | **n2+4n/3+3** | |  | **T(n) = n2+4n/3+3, T(n) = O(n2)** | |
| sum = 0;  for( i = 1; i < n; ++i )  for( j = 1; j < i \* i; ++j )  for( k = 0; k <n; ++k )  ++sum; Solution: values of i: 1,2,3,4,5,6,7,……………….,n  iterations of j: 12+22+32+42+52+62+72+……………+n2  iterations of k: n\*12+n\*22+n\*32+ n\*42+ n\*52+ n\*62+ n\*72+……………+n\*n2  **Series for j**: 12+22+32+42+52+62+72+……………+n2  **Formula:**    12+22+32+42+52+62+72+……………+n2 = (n(n+1)(2n+1))/6 = (2n3 + 3n2 + n)/6  **Series for k**: n\*12+n\*22+n\*32+ n\*42+ n\*52+ n\*62+ n\*72+……………+n\*n2  = n(12+22+32+42+52+62+72+……………+n2)  **Formula:**    n(12+22+32+42+52+62+72+……………+n2)= n(n(n+1)(2n+1))/6 = n(2n3 + 3n2 + n)/6 = 2n4+3n3+n2/6   |  |  | | --- | --- | | **Statement** | **Number of times executed** | | sum=0 | 1 | | i=1 | 1 | | I<n | n+1 | | i=i++ | n | | j=1 | n | | J<i\*i | ((2n3 + 3n2 + n)/6)+n+1) | | J++ | (2n3 + 3n2 + n)/6 | | if( j % i == 0 ) | (2n3 + 3n2 + n)/6) | | K=0 | (2n3 + 3n2 + n)/6) | | K<j | 2n4+3n3+n2/6 +(2n3 + 3n2 + n)/6) +1 | | K++ | 2n4+3n3+n2/6 | | Sum++ | 2n4+3n3+n2/6 | | **Total** | **(6n4 + 19n3 + 18n2 + 5n)/6) + 4n +5** | |  | **T(n) = (6n4 + 19n3 + 18n2 + 5n)/6) + 4n +5 = O(n4)** | |

**Question 3. Find out what does each of the following algorithm do. Then estimate the best-case and the worst-case running time in term of tight big Oh for each of the following codes**

|  |
| --- |
| a)  int Func(int n)  {  int i;  i = 0;  while (n%3 == 0) {  n = n/3;  i++;  }  return i;  } Solution: Best case: n is not multiple of 3 so O(1)  Worst case: n is multiple of 3 so O(log3n) |
| **b)**  len=1;  for (i = 0; i < n-1; i++) {  i1 = i2 = i;  for (j = i; j < n-1 && a[j] < a[j+1]; j++, i2++);  if ( len < i2 - i1 + 1)  length = i2 - i1 + 1;  } Solution: Best case is O(n)  **Worst Case:**  values of i: 0,1,2,3,4,5,6,7,……………….,n-1  iterations of j: n-1 + n-2 + n-3 + n-4 + ……..+ 2+1 = 1+2+3+………………….+n-1    **Formula:**    Series of j = 1+2+3+………………….+n-1 = (n-1)(n/2)= n2/2 – n/2   |  |  | | --- | --- | | **Statement** | **Number of times executed** | | Len=1 | 1 | | i=0 | 1 | | i<n-1 | n-1 + 1 =n | | i++ | n-1 | | I1 = i2=i | n-1 | | j=i | n-1 | | j<n-1 && a[j]<a[j+1] | n2/2 – n/2 + n-1 | | J++,i2++ | n2/2 – n/2 | | if ( len < i2 - i1 + 1) | n2/2 – n/2 | | length = i2 - i1 + 1; | n2/2 – n/2 | | **Total** | **2n2  + 3n - 3** | |  | **T(n) = 2n2  + 3n - 3, T(n) = O(n2)** | |
| **c)**  int Mystery( int a[], int asize ){  int mSum = 0;  for( int i = 0; i < asize; ++i ){  int thisSum = 0;  for( int j = i; j < asize; ++j ){  thisSum += a[ j ];  if( thisSum > mSum )  mSum = thisSum;  }  }  return mSum;  } Solution: Best case is O(asize)  **Worst Case:**  values of i: 0,1,2,3,4,5,6,7,………………., asize -1  iterations of j: asize -1 + asize -2 + asize -3 + asize -4 + ……..+ 2+1 = 1+2+3+………………….+ asize -1    **Formula:**    Series of j = 1+2+3+………………….+ asize -1 = ((asize -1)( asize ))/2= asize 2/2 – asize /2   |  |  | | --- | --- | | **Statement** | **Number of times executed** | | mSum = 0; | 1 | | i=0 | 1 | | i< asize -1 | asize -1 | | i++ | asize -2 | | thisSum = 0 | asize -2 | | j=i | asize -2 | | j < asize | asize 2/2 – asize /2+ asize -2 | | j++ | asize 2/2 – asize /2 | | thisSum += a[ j ]; | asize 2/2 – asize /2 | | if(thisSum>mSum) | asize 2/2 – asize /2 | | mSum = thisSum; | asize 2/2 – asize /2 | | **Total** | **2 asize 2 + asize 2 /2 +2 asize -6** | |  | **T(n)= 2 asize 2 + asize 2 /2 +2 asize -6= O(asize2)** | |