Complexity Analysis – Practice questions

Data Structures

Question 1. Write the tight big-O for the following expressions and **find c and n_0**

1. **$9n^3+400n^2$**

Solution:

Proof:

Since,

T(n) <= cf(n)

$9n^3 +400n^2 <= c(n^3)$ -- (1)

$(9n^3 +400n^2)/(n^3) <= c$

Put n=2:

$(9*2^3 +400*2^2)/(2^3) <= c$

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

1672/8 <=c

209 <= c

c >= 209

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

$9*2^3 +400*2^2 <= 209(2^3)$

1672<= 1672 (hold true)

C=209 n_0=2

Hence,proved.

2. **$n^3 2^n + n^2 3^n$**

Solution:

Proof:

Since,

T(n) <= cf(n)

$n^3 2^n+n^2 3^n <= c(n^2 3^n)$ -- (1)

$(n^3 2^n+n^2 3^n)/(n^2 3^n) <= c$

Put n=2:

$(2^3 2^2+2^2 3^2)/(2^2 3^2)$ <= c

(32 +36)/36) <= c

68/36 <=c

1.89 <= c

c >= 1.89 ~2

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

$2^3 2^2+2^2 3^2$<= $2(2^2 3^2)$

68<= 72 (hold true)

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

C=2 n_0=1

Hence,proved.

3. **n^2/logn + n**

Solution:

<u>Proof:</u>

Since,

T(n) <= cf(n)

n^2/logn + n<= c(n^2/logn) -- (1)

(n^2/logn + n)/(n^2/logn) <= c

Put n=2:

(2^2/log2 + 2)/(2^2/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):

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

15.29<= 26.58 (hold true)

C=2 n_0=2

Hence,proved.

4. **n^{k+a}+2^n**

Solution:

Proof:

Since,

$T(n) <= cf(n)$

$n^{k+a}+2^n <= c(2^n)$ -- (1)

$(n^{k+a}+2^n) /2^n <= c$

Put n=2;

$(2^{k+a}+2^2) /2^2 <= c$

$(2^{k+a-2}+1) <= c$

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

$(2^{k+a}+2^2) <= 2^{k+a-2} +1*(2^2)$

$(2^{k+a}+2^2) <= 2^{k+a-2+2} +2^2$

$(2^{k+a}+2^2) <= 2^{k+a} +2^2$ (holds true)

$C=(2^{k+a-2}+1)$ $n_0=2$

Hence,proved.

5. **$n^{k+a}+n^k$ log n**

Solution:

Proof:

Since,

$T(n) <= cf(n)$

$n^{k+a}+n^k logn <= c(n^{k+a})$ -- (1)

$(n^{k+a}+n^k logn) / n^{k+a} <= c$

Put n=2;

$(2^{k+a}+2^k log2) / 2^{k+a} <= c$

$(2^{k+a-k-a}+2^{k-k-a} log2) <= c$

$(1+2^{-a} log2) <= c$

Putting $c=(1+2^{-a} log2)$ and n=2 in Eq. (1):

$2^{k+a}+2^k log2 <= (1+2^{-a} log2)(2^{k+a})$

$2^{k+a}+2^k log2 <= (2^{k+a} +(2^{-a} log2)* 2^{k+a})$

$2^{k+a}+2^k log2 <= 2^{k+a} +2^k log2$ (holds true)

$C=(1+2^{-a} log2)$ $n_0=2$

Hence,proved.

6. **$5n^3 + \sqrt{n}$ logn+ n*(2n+n log n)**

Solution:

Proof:

Since,

T(n) <= cf(n)

5n^3 + n^1/2*logn + 5n^2 + n^2logn <= c(n^3) ------------------------------ (1)

(5n^3 + n^1/2*logn + 5n^2 + n^2logn) / n^3 <= c

Put n=2;

(40+1.41 *1 + 20 + 4*1)/8 <= c

8.17 <=c

c>= 8.17 ~9

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

40+ (2)^1/2 + 20 + 4 <= 9*8

65.41 <= 72 (holds true)

C=9 n_0=2

Hence,proved.

7. **(100n+logn) * (25n+log n)**

Solution:

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

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

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

Proof:

Since,

T(n) <= cf(n)

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

(2500n^2 + 100nlogn + 25nlogn + (logn)^2) /n^2<= 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 n_0=1

Hence,proved.

8. **n^3+ 4n+ 2^n**

Solution:

Proof:

Since,

T(n) <= cf(n)

n^3+ 4n+ 2^n <= c(2^n) -- (1)

(n^3+ 4n+ 2^n)/(2^n) <= c

Put n=2:

(2^3+ 4*2+ 2^2)/(2^2) <= c

28/4 <=c

7 <= c

c >= 7

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

(2^3+ 4*2+ 2^2) <= 7(2^2)

28<= 28 (hold true)

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

C=7 n_0=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.

```
a)
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…+($1+\log_2 n$) = ($1+\log_2 n$)(2 + $\log_2 n$)/2

Steps	Step Counts
Int i=0	1
i <= n	$\log_2 n$ $\log_2 n$ + 2
i = i * 2	$\log_2 n$ $\log_2 n$ + 1
j = 1	$\log_2 n$ $\log_2 n$ + 1
j <= i	($1+\log_2 n$)(2 + $\log_2 n$)/2 + $\log_2 n$ + 1
j = j * 2	($1+\log_2 n$)(2 + $\log_2 n$)/2
Cout << "*"	($1+\log_2 n$)(2 + $\log_2 n$)/2

T(n) = sum of all terms in the last column = (compute this yourself)

T(n) = O(($\log_2 n$)^2)

b)

```
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	$\log_4 n+1$
I=i\4	$\log_4 n$
cout<<i	$\log_4 n$
j=0	$\log_4 n$
J<n	$\log_4 n(n/2+1) = \log_4 n(n/2) + \log_4 n$
J=j+2	$\log_4 n(n/2)$

sum++	$\log_4 n(n/2)$
Total	**$5 \log_4 n + 3 \log_4 n(n/2) + 2$**
	$T(n) = 5 \log_4 n + 3 \log_{4n}(n/2) + 2$, $T(n) = O(\log_4 n(n/2))$

c)

```
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	$n/3(n+1) = n^2/3 + n/3$
j--	$n/3(n) \quad = n^2/3$
sum++	$n/3(n) \quad = n^2/3$
Total	**$n^2+4n/3+3$**
	$T(n) = n^2+4n/3+3$, $T(n) = O(n^2)$

d)

```
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: $1^2+2^2+3^2+4^2+5^2+6^2+7^2+\ldots\ldots\ldots\ldots+n^2$

iterations of k: $n*1^2+n*2^2+n*3^2+ n*4^2+ n*5^2+ n*6^2+ n*7^2+\ldots\ldots\ldots\ldots+n*n^2$

<u>**Series for j**</u>: $1^2+2^2+3^2+4^2+5^2+6^2+7^2+\ldots\ldots\ldots\ldots+n^2$

<u>**Formula:**</u>

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \text{ (for n >= 1)}$$

$1^2+2^2+3^2+4^2+5^2+6^2+7^2+\ldots\ldots\ldots\ldots+n^2 = (n(n+1)(2n+1))/6 = (2n^3 + 3n^2 + n)/6$

<u>**Series for k**</u>: $n*1^2+n*2^2+n*3^2+ n*4^2+ n*5^2+ n*6^2+ n*7^2+\ldots\ldots\ldots\ldots+n*n^2 = n(1^2+2^2+3^2+4^2+5^2+6^2+7^2+\ldots\ldots\ldots\ldots+n^2)$

<u>**Formula:**</u>

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \text{ (for n >= 1)}$$

$n(1^2+2^2+3^2+4^2+5^2+6^2+7^2+\dots\dots\dots\dots+n^2) = n(n(n+1)(2n+1))/6 = n(2n^3+3n^2+n)/6 = 2n^4+3n^3+n^2/6$

Statement	**Number of times executed**
sum=0	1
i=1	1
I<n	n+1
i=i++	n
j=1	n
J<i*i	$((2n^3+3n^2+n)/6)+n+1)$
J++	$(2n^3+3n^2+n)/6$
if(j % i == 0)	$(2n^3+3n^2+n)/6)$
K=0	$(2n^3+3n^2+n)/6)$
K<j	$2n^4+3n^3+n^2/6+(2n^3+3n^2+n)/6)+1$
K++	$2n^4+3n^3+n^2/6$
Sum++	$2n^4+3n^3+n^2/6$
Total	$\mathbf{(6n^4+19n^3+18n^2+5n)/6)+4n+5}$
	$\mathbf{T(n) = (6n^4+19n^3+18n^2+5n)/6)+4n+5 = O(n^4)}$

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 O(1)
Worst case n is multiple of 3 … $\log_3 n$

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:

$$\sum_{i=1}^{n} i = \frac{n}{2}(n+1)$$

Series of j = 1+2+3+………………….+n-1 = (n-1)(n/2)= $n^2/2 - n/2$

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]	$n^2/2 - n/2 + n-1$
J++,i2++	$n^2/2 - n/2$
if (len < i2 - i1 + 1)	$n^2/2 - n/2$
length = i2 - i1 + 1;	$n^2/2 - n/2$
Total	**$2n^2 + 3n - 3$**
	$T(n) = 2n^2 + 3n - 3$, $T(n) = O(n^2)$

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 and 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:

$$\sum_{i=1}^{n} i = \frac{n}{2}(n+1)$$

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(asize^2)

Question 4. Write an algorithm for following problems and **derive tight Big-O of your algorithm**

- Reverse an array of size n: O(n)
- Find if the given array is a palindrome or not
- Sort array using bubble sort
- Sort array using selection sort
- Sort array using insertion sort
- Print a square matrix of size nxn: O(n^2)
- Sum two matrices of size nxn: O(n^2)
- Product of two matrices of size nxn: O(n^3)
- Transpose of a matrix
- Printing all numbers that can be represented by n bits: O(2^n)
- Printing all subsets of numbers in an array of size n: O(2^n)
- Printing all permutations of numbers in array of size n: O(n!)