RULES FOR CALCULATING TIME COMPLEXITY AND BIG-OH

Rule 00

Normally these formulas are very handy :

If $x^y = z$ then $y = \log_x z$

Also,

$$\sum_{i=1}^{n} a_i = \frac{n}{2}(a_1 + a_n) \qquad \sum_{i=1}^{n} i = \frac{n}{2}(n+1) \qquad \sum_{k=0}^{m} r^k = \frac{1-r^{m+1}}{1-r} \qquad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \text{ (for n >=1)}$$

Rule 0

The condition that stops a loop executes ONE MORE time than the loop itself (the last time is when it is evaluated false)

Rule 1

for (i=0;i<n;i=i+k) Anything inside the loop will run approximately **n/k** times

Rule 2

for (i=n;i>0;i=i-k) Anything inside the loop will run approximately **n/k** times

Rule 3

for (i=1;i<n;i=i*k) Anything inside the loop will run approximately **$\log_k n$** times

Rule 4

```
for(i=1;i<=n;++i)
        for (j=1;j<=i;++j)
```

The above nested loop approximately runs **½ n(n+1)** times.
The variable j depends upon the value of i

Rule 5

```
for(i=1;i<=n;i=i*2)
        for (j=1;j<=i;++j)
```

The statements in the above nested loop approximately run **2n-1** times.
The variable j depends upon the value of i

Rule 6

If the loop variables are independent then the total times a statement inside a nested loop is executed is equal to the product of the times the individual loops run

e.g.

```
for (i=0;i<n;++i)
        for (j=0;j<m;++j)
```

A statement inside the above nested loop will run **n*m** times

CS 2002 Data Structures and Algorithms
Exercises
Time Complexity and Big-Oh

Assume that basic operations and input output take single time units to complete. Calculate the time complexity function T(n) and O(n) for the following program fragments:

1.
```
int sum,i;
sum = 0;
for (i=0;i<n;++i)
        sum++;
```
Ans: T(n) = 3n+3,T(n) = O(n)

2.
```
int sum,i,j;
sum = 0;
for (i=0;i<n;++i)
{       for (j=0;j<n;++j)
        {       sum++;
        }
}
```
Ans: T(n) = $3n^2$+4n+3, T(n) = O(n^2)

3.
```
int sum,i;
sum = 0;
for (i=0;i<n;i=i+2)
        sum++;
```
Ans: T(n)=3n/2+3, T(n) = O(n)

4.
```
int sum,i,j;
sum = 0;
for (i=0;i<n;i=i+3)
{       for (j=0;j<n;j=j+2)
        {       sum++;
        }
}
```
Ans: T(n) = n^2/2 + 4n/3 + 3, T(n) = O(n^2)

5.
```
int sum,i,j;
sum = 0;
for (i=n; i>=1; i=i-3)
{       for (j=n;j>0;j--)
        {       sum++;
        }
}
```
Ans: T(n) = n^2+4n/3+3, T(n) = O(n^2)

6.
```
int sum,i,j;
sum = 0;
for (i=1;i<n;i=i*2)
        sum++;
```
Ans: T(n) = 3log n + 3, T(n)=O(log n)

7.
```
int sum,i,j;
sum = 0;
for (i=1;i<n;i=i*2)
{       for (j=0;j<n;++j)
        {       sum++;
        }
}
```
Ans: T(n) = 3nlog n + 4 log n + 3 , T(n)=O(nlog n)

8.

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

        }
        cout << Sum;
}
```

Ans: $T(n) = 2n^2 + 9n + 2$, $T(n) = O(n^2)$

9.

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

Ans: $T(n) = 10n+7\log_2 n-2$, $T(n) = O(n)$

10.

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

Ans: $T(n) = 2n\log_4 n+6\log_4 n+2$,
$T(n) = O(n \log n)$

11.

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

Ans: $T(n) = 4/3n\log_3 n+2n+2$,
$T(n) = O(n \log n)$

Question 12.

```
 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=1	1
i <= n	$\log_2 n$ + 2
i = i * 2	$\log_2 n$ + 1
j = 1	$\log_2 n$ + 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($(\log_2 n)^2$)

The above material was prepared from material provided by Dr. Mehreen Saeed for Data Structures and Algorithms course.

Question 1) Find time and space complexity of the following functions in terms of Big-Oh:

(i)
```
void func(int n)
 {  if (n>0)
    { func(n/k); //where k is some constant }
 }
```

(ii)
```
public void code1(int n)
{
  for (int i=n; i>=0; i--)
  {
    for (int j=0; j<=999; j++)
    {
      int num = i*j;
      int* array = new  int[num];
    }
  }}
```

Question 2) While doing worst case analysis of an algorithm, I found it takes 2^{n+1} steps to complete the job, where n is the input size. Can I say it is $O(2^n)$? Why or why not?

Question 3) While doing worst case analysis of an algorithm, I found it takes 2^{2n} steps to complete the job, where n is the input size. Can I say it is $O(2^n)$? Why or why not?