Data Structure, Algorithms and Programming

GATEBOOK

April 2018

1 Asymptotic Notations

1. Assign truth value to each of following statements:

   (a) If $f(n) = 100 * n^4 + 2 * n^5 + 1$ then $f(n) = O(n^4)$

   (b) If $f(n) = O(g(n))$ and $g(n) = \Omega(f(n))$ then $f(n) = \theta(g(n))$

   (c) $f(n) = \Omega(n)$ ensures that at best case some algorithm whose running time is given by f(n), can process input in linear time.

   (d) If an algorithm A can processes input of size n in A(n) time units and another algorithm B processes the same input in B(n) time units then A(n)≥B(n) provided algorithm A processes inputs faster than algorithm B.

   (e) Big-O is a representative of time requirement in worst case scenario while Ω always represents the same in Best case scenario.

   (f) If $f(n) = 2^{log_2 n}, g(n) = 2^{\sqrt{n}}, h(n) = nlogn^{logn}, k(n) = n^{log_2 2}$, then these time complexities can be arranged in non-decreasing order as follows:
   $f(n) = k(n) \leq h(n) \leq g(n)$

   (g) In a program if there is a loop inside another loop then the asymptotic time complexity for this program can be shown by $\Omega(n^2)$

   (h) If both of the algorithms A and B need $O(nlogn)$ time then they both are equally efficient and finish in same amount of time.

   (i) It is known that for three algorithms, all of which solve the same problem, the time requirements can be given as follows:
   $f(n) = O(n^2)$,
   $g(n) = O(nlogn)$ and
   $h(n) = O(n^3)$.
   Since g(n) has the least time requirements among all, it must always be the preferred for all inputs.

   (j) If $f(n) = O(g(n))$ and $g(n) = O(f(n))$ then $f(n) = g(n)$

(k) If $f(n) = \sum_{1 \leq k \leq 10000} n$, then $f(n) = \Theta(n^2)$

2. Given any two functions f(n) and g(n), show that:
   $f(n) + g(n) = \Theta(max(f(n), g(n)))$.

3. Arrange the following functions in their increasing order of complexities.
   $f(n) = n^{0.999999} logn$
   $g(n) = 10^6 n$
   $h(n) = n^2$
   $k(n) = 1.000001^n$
   $p(n) = \frac{2^{\sqrt{n}}}{n^2}$
   $q(n) = \frac{n^{1.000001}}{logn}$

4. If $f(n) = O(g(n))$, $f(n) = \Omega(n^3)$, where $g(n)$ returns the fifth root of unity. Which of the following is correct?
   (A) $f(n)$ could possibly return square root of unity.
   (B) $f(n) = \Theta(n^4)$
   (C) $f(n)$ could possibly return fourth root of unity.
   (D) $f(n) = \Omega(n^3)$ and $f(n) * f(n = o(n^{10})$

5. Arrange the following in increasing order of complexity:
   $n^2, logn^{logn}, n^{\sqrt{logn}}, (n+1)^2, n!, \frac{n!}{10^{10}}, n^{2^n}, 2^{log_2 n} * n^{100}, 1.5^{n^2}, (n * n!)^{log_n n}$

6. Find the complexity of the following code fragment:
   for($i = 0; i < n * n; i* = i$)
   {
   printf(“*”);
   }

7. Find the complexity of the following code fragment:
   for($i = 1; ; i++$)
   {
   printf(“*”);
   }

8. Find the complexity of the following code fragment:
   int $i = 1$;
   for($; i <= n * logn; i++$)
   {
   for($i++; i <= n; i++$)
   {
   $\Theta(1)$
   }

   }

9. Find the complexity of the following code fragment:

```
int i=1,sum=0;
for(; i <= n; n = n + n, i = i * i)
{
for(i + +; i <= n; i + +)
{
sum + +;
}

}
```

10. Find the complexity of the following code fragment:

```
int i=1,j;
for(; i <= n; i + +)
{
for(j = i; j <= nlogn; j* = 2)
{
sum + +;
}

}
```

11. Find the complexity of the following code fragment:

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

12. Find the complexity of the following code fragment:

```
int i=1,j;
for(; i <= n; i + +)
{
for(j = n^2; j >= 1; j/ = 2)
{
sum + +;
```

```
}
}
```

13. Find the complexity of the following code fragment:

```
int i, j,k, count 0;
for (i = n/3; i <= n; i + +)
{
for (j = 1; j <= n/2; j = 2 * j)
{
for (k = 1; k * k <= n; k + +)
{
count++;
}
}
}
```

14. Find the complexity of the following function when called with some integer n:

```
void foo(n)
{
int i,j,k,x=0;
for (i=1 ; i ≤ n ; i++)
for (j=1 ; j ≤ i * i ; j++)
{
for ( k = 1 ; k ≤ j ; k++)
x=x+10;
}
}
```

15. Find the complexity of the following code fragment:
a[] *is an array storing either 1 or 0 and f(n) has time complexity as* $\Theta(n)$

```
int c = 0;
for (i = 1; i <= n; i + +)
{ if (a[i] == 1) c++;
else f(c); c = 0;
}
```

16. Find the complexity of the following function when called with some integer n:

```
void foo(n)
{
int i,j,k,x=0;
```

```
for (i=1 ; i ≤ n ; i++)
for (j=1 ; j ≤ i * i ; j++)
{
if ( j mod i == 0 )
for ( k = 1 ; k ≤ j ; k++)
x=x+10;
}
}
```

17. Consider the function kth_root() defined as follows:
    kth_root(n)
    {
    return k^{th} root of n in $\Theta(1)$ time.
    }
    Find the complexity of the following code fragment:
    for (int i = n; i > 0; i = kth_root(i))
    {
    // O(1)
    }

18. Find the complexity of the following code fragment:

```
int i, j,k, count 0;
for (i = n; i > 1; i = log2(i))
{
for (j = 2^i; j >= 1; j − −)
{
for (k = 1; k <= log2 j; k + +)
{
count++;
}
}
}
```

19. Find the complexity of the following code fragment:

```
int i, j,k, count 0;
for (i = n; i > 1; i − −)
{
for (j = i; j <= n!; j = j ∗ j)
{
for (k = n; k >= 1; k/ = 2)
{
```

```
count++;
}
}
}
```

20. Find the complexity of the following code fragment

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

21. Find the complexity of the following code fragment while the function foo() is defined as follows:

```
foo(n)
{
//returns closest integer to n/2 in Θ(logn) time
}

int i, j,k, count 0;
for (i = n; i >= 1; i − −)
{
for (j = i; j <= n!; j = j ∗ j)
{
for (k = n; k >= 1; k = foo(k))
{
count++;
}
}
}
```

2 C Programming

1. What does the following C program do? Assume all necessary libraries are included.

```
int main()
{
int x, i;
scanf("%d", &x);
for(i = 1; i <= 100; i + +)
{
if((i%x) == 3)
{
```