Practice Set: Loop Complexities

Takeaway from class-

Increment => $O(n)$ $(i = i + c)$
Doubling => $O(\log n)$ $(i = 2 * i)$
Exponentiation => $O(\log \log n)$ $(i = i * i)$

Question 1: What will be the time complexity of following nested loop ?

Consider the following program fragment:

```
for (int i=0; i * i < N; i++)
    for (int j=0; j * j < N; j++)
        printf("Hello!");
```

Which one of the following statements about the runtime $R(N)$ is true?

A $R(N) = \Theta(\log N)$

B $R(N) = \Theta(\sqrt{N})$

C $R(N) = \Theta(N)$

D $R(N) = \Theta(N^2)$

E $R(N) = \Theta(2^N)$

Answer:

$\boxed{\text{C}}$ Note that the inner loop will be executed $\sqrt{N}\sqrt{N}$ times.

Question 2

What will be the time complexity of following nested loop ?

```
for( int i = n; i > 0; i /= 2 ) {
    for( int j = 1; j < n; j *= 2 ) {
        for( int k = 0; k < n; k += 2 ) {
            ... // constant number of operations
        }
    }
}
```

Answer: $\theta(n(logn)^2)$

Question 3

What will be the time complexity of following nested loop ?

```
for ( int k = n; k >0; k /= 3 ) {
  for ( int i = 0; i < n; i += 2 ) {
     // constant number C of elementary operations
  }
  for ( int j = 2; j < n; j = (j*j)) {
      // constant number C of elementary operations
  }
}
```

Answer: $\theta(nlogn)$

Question 4

What will be the time complexity of following nested loop ?

```
int i=1;
while (i<= n) {
   int j = i;
   while (j > 0) {
      j = j/2;
   }
   i++;
}
```

Answer: $\theta(nlogn)$

Question 5

What will be the time complexity of following nested loop ?

```
for (k = 1; k <= n; k += 1)
{
    for (i = 1; i <= n; i *= 3)
    {
        j = i;
        while (j > 1)
        {
            sum += 1;
            j /= 3;
        }
    }
}
```

Answer: $\theta(n\ (logn)^2)$

Inner most while loop will run $\log i$ times.

For i = 1, inner most loop will run log 1 times
For i =3, inner loop will run log 3 times
For i $= 3^2$, inner loop will run log 3^2 times
For i $= 3^3$, inner loop will run log 3^3 times

.

.

For i = $3^k = n$, inner loop will run log 3^k times (here k = $\log n$)

(We will consider outermost for loop after a while, lets focus on inner for loop)
Inner for loop will run log1 + log3 + log 3^2 + log 3^3 + …..+ log 3^k

$$= \log 1.3.3^2.3^3.3 \ldots .3^k = \log 3^{1+2+3+\cdots k}$$

$$= \log 3^{k^2} = k^2 = (logn)^2$$

Question 6

What will be the time complexity of following nested loop ?

```
for(int i = 0; i < N*N; i++) {
    for(int j = 0; j < i; j++) {
        //something O(1)
    }
}
```

Answer: $\theta(n^4)$

Question 7

What will be the time complexity of following nested loop ?

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

Answer: $\theta(n^5)$

The most inner loop is executed in quadratic time (not constant), hence it should be `O(n) * O(n^2) * O(n^2) = O(n^5)`.

Here are all the costs:

- Most outer loop - `O(n)`
- The second loop - `O(n^2)` for each element for the outer loop
- The most inner loop - `O(n^2)` for each element for the second loop

```
for (int i = 0; i < n; i++) -> runs n times.
 for (int j = 0; j < n * n; j++) -> runs n² times.
  for (int k = 0; k < j; k++) -> runs n² times (k == j == n²)

n * n² * n² = n^5.
```

Question 8

What will be the time complexity of following loop ?

```
i = 1;
k = 1;

while(k<n){
    k = k+ i;
    i = i + 1;
}
```

Answer: $\theta(\sqrt{n})$

Let m be the number of the iteration, then the values of k and i evolve as follows:

m	k	i
0	1	1
1	1+1	2
2	1+1+2	3
3	1+1+2+3	4
4	1+1+2+3+4	5

So we see that k is $1 + \sum_{i=1}^{m} i$

This sum is a triangular number, and so:

$$k = 1 + m(m+1)/2$$

And if we fix m to the total number of iterations, then:

$$1 + m(m-1)/2 < n \leq 1 + m(m+1)/2.$$

So n is $O(m^2)$, and thus m is $O(\sqrt{n})$.

The number of iterations m is a measure of the complexity, so it is $O(\sqrt{n})$.

Question 9

What will be the time complexity of following loop ?

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

Answer: $\theta(nlogn)$

Inner loop will run $\frac{n}{2i}$ times.

For i = 1, inner loop will run n/2 times
For i =2, inner loop will run n/(2x2) times
For i =3, inner loop will run n/(2x3) times
For i =4, inner loop will run n/(2x4) times

.

.

For i =n, inner loop will run n/(2xn) times

Time Complexity :

$$= \frac{n}{2} + \frac{n}{2\times 2} + \frac{n}{2\times 3} + \frac{n}{2\times 4} + \frac{n}{2\times 5} + \dots \dots .. + \frac{n}{2\times n} =$$
$$= n/2\,[1/2 + 1/3 + 1/4 + \cdots ..1/n] = nlogn$$

Question 10

What will be the time complexity of following loop ?

```
p = 0
for( i=1; i<n; i=i*2 ) {
    p++
}

for( j=1; j<p; j=j*2 ) {
    some_statement
}
```

Answer: $\theta(log n)$

Because they are not independent: The first computes *p = O(log n)* and the second depends on it. The time complexity of the second loop would be *O(log p) = O(log log n)*.

However, the first loop is still *O(log n)*, which indeed makes the time complexity of the entire program *O(log n + log log n) = O(log n)*, as you say.

Question 11

What will be the time complexity of following loop ?

```
for (int j = 2; j < N; j++) {
    for (int k = 2*j; k <= N; k += j) {
        some_statement
    }
}
```

Answer: $\theta(nlogn)$

Inner loop will run $\frac{n-2j}{j} = \frac{n}{j} - 2 = \frac{n}{j}$ times.

For j = 1, inner loop will run n/1 times
For j =2, inner loop will run n/2 times
For j =3, inner loop will run n/3 times
For j =4, inner loop will run n/4 times

.

.

For j =n, inner loop will run n/n =1 time

Time Complexity :

$$= \frac{n}{1} + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \frac{n}{5} + \ \ldots\ldots.. + \frac{n}{n}$$
$$= \ n\ [1 + 1/2 + 1/3 + 1/4 + \cdots..1/n] \ = \ nlogn$$

Question 12

What will be the time complexity of following loop ?

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

Answer: $\theta(nlogn)$

Question 13

What will be the time complexity of following loop ?

```
for (i=1;i<=n;i*=2){
  for (j=1;j<=i;j++) {
    // some O(1) operation
  }
}
```

Answer: $\theta(n)$

For i = 1, inner most loop will run 1 times
For i =2, inner loop will run log 2 times
For i $= 2^2$, inner loop will run 2^2 times
For i $= 2^3$, inner loop will run 2^3times

.
.
For i = $2^k = n$, inner loop will run 2^k times (here k = $\log n$)

Question 14

What will be the time complexity of following loop ?

```
for (int i = 1; i < n; i*=2)
  for (int j = 0; j < i; j +=2)
  {
    // some contstant time operations
  }
```

Answer: $\theta(n)$

Question 15

What will be the time complexity of following loop ?

```
for ( int i = 1; i < n*n*n; i *= n ) {
    for ( int j = 0; j < n; j += 2 ) {
        for ( int k = 1; k < n; k *= 3 ) {
            // some contstant time operations
        }
    }
}
```

Answer: $\theta(n \log n)$

Question 16

What will be the time complexity of following loop ?

```
for(int i = 0; i < n^3; i++){
    if(i % 3 == 0){
        break;
    }
    else{
        print ":D"
    }
}
```

Answer: $\theta(1)$

Question 17

What will be the time complexity of following loop ?

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

https://courses.cs.washington.edu/courses/cse332/21au/exams/oldExams/cse332-midterm-12wi-soln.pdf

Answer: $\theta(n)$

Question 18

What will be the time complexity of following nested loop ?

```
int n;
int sum;
for (int i = 1; i < n; i++)
{
  for (int j = 0; j < i*i; j++)
    {
      if (j % i == 0)
        {
          for (int k = 0; k < j; k++)
            {
              sum++;
            }
        }
    }
}
```

Answer: $\theta(n^4)$

The `if` condition will be true when `j` is a multiple of `i`; this happens `i` times as `j` goes from 0 to `i * i`, so the third `for` loop runs only `i` times. The overall complexity is O(n^4).

```
for (int i = 1; i < n; i++)
{
  for (int j = 0; j < i*i; j++)        // Runs O(n) times
    {
      if (j % i == 0)                  // Runs O(n) × O(n^2) = O(n^3) times
        {
          for (int k = 0; k < j; k++) // Runs O(n) × O(n) = O(n^2) times
            {
              sum++;                   // Runs O(n^2) × O(n^2) = O(n^4) times
            }
        }
    }
}
```

Question 19

What will be the time complexity of following loop ?

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

Answer: $\theta((logn)^2)$

```
for (i = 1; i <= N; i = i*2)
    for (j = 1; j <= i²; j=j*2)   ← log i²
        sum++;
```

$i = 1 \quad \log 1^2$

$i = 2 \quad \log 2^2$

$i = 2^2 \quad \log(2^2)^2 \rightarrow 2^4$

$i = 2^3 \quad \log(2^3)^2 \rightarrow 2^6$

$\vdots$

$i = 2^k \quad \log(2^k)^2 \rightarrow 2^{2k}$

$2^k = N \Rightarrow k = \log_2 N$

$\log 1^2 + \log 2^2 + \log 2^4 + \log 2^6 + \log 2^8 + \cdots \log 2^{2k}$

$= 2\log 2 + 4\log 2 + 6\log 2 + \cdots 2k\log 2$

$= 2 + 4 + 6 + \cdots 2k$

$= 2(1+2+3+\cdots k) = O(k^2)$

$= O((\log n)^2)$

Question 20

What will be the time complexity of following loop ?

```
for ( i=1; i < n; i *= 2 )
for ( j = n; j > 0; j /= 2 )
for ( k = j; k < n; k += 2 ) {
sum += (i + j * k );
}
```

Answer: $\theta(n(logn)^2)$

Running time of the inner, middle, and outer loop is proportional to n, $\log n$, and $\log n$, respectively. Thus the overall Big-Oh complexity is $O(n(\log n)^2)$

More detailed optional analysis gives the same value. Let $n = 2^k$. Then the outer loop is executed k times, the middle loop is executed $k+1$ times, and for each value $j = 2^k, 2^{k-1}, \ldots, 2, 1$, the inner loop has different execution times:

j	Inner iterations
2^k	1
2^{k-1}	$(2^k - 2^{k-1})\frac{1}{2}$
2^{k-2}	$(2^k - 2^{k-2})\frac{1}{2}$
...	...
2^1	$(2^k - 2^1)\frac{1}{2}$
2^0	$(2^k - 2^0)\frac{1}{2}$

In total, the number of inner/middle steps is

$$\begin{aligned} 1 + k \cdot 2^{k-1} - (1 + 2 + \ldots + 2^{k-1})\frac{1}{2} &= 1 + k \cdot 2^{k-1} - (2^k - 1)\frac{1}{2} \\ &= 1.5 + (k-1) \cdot 2^{k-1} \equiv (\log_2 n - 1)\frac{n}{2} \\ &= O(n \log n) \end{aligned}$$

Thus, the total complexity is $O(n(\log n)^2)$.

Thank you