Lecture 5 Algorithm Performance Analysis

1. What does Big-O notation represent?
A) The exact runtime of an algorithm
B) The upper bound of an algorithm's growth rate
C) The lower bound of an algorithm's growth rate
D) The average runtime of an algorithm
Answer: B

2. What does asymptotic analysis focus on?
A) Exact runtime values for specific inputs
B) Program behavior as input size becomes very large
C) Measuring time using a stopwatch
D) Hardware-specific performance metrics
Answer: B

3. What is the average-case complexity of searching for a letter in a word using linear search?
boolean hasLetter(String word, char letter);
A) O(1)
B) O(\log n)
C) O(n)
D) O(n \log n)
Answer: C

4. Which is true about an algorithm's average-case complexity?
A) It must be equal to best-case complexity.
B) It must be equal to worst-case complexity.
C) It lies between best-case and worst-case complexities.
D) It cannot be determined without more information.
Answer: C

5. Which of the following complexities is the fastest for large input sizes?
A) O(n^2)
B) O(n \log n)
C) O(n)
D) O(\log n)
Answer: D

6. If an algorithm has a runtime of f(n) = 3n + 5, what is its Big-O complexity?
A) O(1)
B) O(n)
C) O(n^2)
D) O(\log n)
Answer: B

7. What is the best-case complexity of a linear search in an array?
A) O(1)
B) O(n)
C) O(\log n)
D) O(n^2)
Answer: A

8. Which notation represents the exact bound of an algorithm's growth rate?
A) Big-O
B) Big-Omega (Ω)
C) Big-Theta (Θ)
D) None of the above
Answer: C

9. Given a function g(n) = 2^n + n^2 + 100, what is its Big-O complexity?
A) O(2^n)
B) O(n^2)
C) O(n \log n)
D) O(1)
Answer: A

10. Given a function g(n) = (n+100)^2 +100n + 100000 n log n, what is its Big-O complexity?
A) O(2^n)
B) O(n^2)
C) O(n \log n)
D) O(1)
Answer: B

11. For binary search on an array of sorted numbers, what is the worst-case time complexity?
A) O(1)
B) O(n)
C) O(\log n)
D) O(n^2)
Answer: C

12. Describe the worst-case running time of the following code in Big-O notation in terms of the variable n.

```
void f(int n) {
  int j = n;
  while (j > 2) {
      // O(1)
      j = j / 2;
  }
}
```

ANS: O(log n)

The function has a while loop that continues as long as j > 2, where j is initially set to n. Within each iteration of the loop, j is divided by 2 using integer division. This pattern of repeatedly halving j is characteristic of logarithmic behavior. Here's why:

- Loop Behavior: The loop halves the value of j in each iteration. This means that the number of times the loop runs is proportional to how many times you can divide n by 2 before it becomes less than or equal to 2.
- Logarithmic Complexity: The number of times you can divide a number by 2 before it becomes less than or equal to a constant (in this case, 2) is approximately the logarithm base 2 of that number. Thus, the number of iterations of the loop is roughly log_2(n)).

Therefore, the worst-case running time of the function f in terms of the variable n, is O(log n) (the base of 2 or 10 does not matter).

13. What is the time complexity of function f1(n) and function f2(n), respectively?

```
void f1(n){
  for (int i = 0; i < n; i+=5) {
        // O(1)
  }
}
void f2(n){
  for (int i = 1; i < n; i*=5) {
        // O(1)
  }
}
```

A) O(\log n), O(\log n)
B) O(\log n), O(n)
C) O(n), O(\log n)
D) O(n), O(n)
Answer: C

14. What is the time complexity of function f(n), which consists of two sequential loops?

```
void f(n){
  for (int i = 0; i < n; i++) {
        // O(1)
}
  for (int i = 1; i < n; i*=2) {
        // O(1)
  }
}
```

A) O(n \log n)
B) O(n^2)
C) O(\log n^2)
D) O(n)
Answer: D

15. What is the time complexity of function f1(n) and function f2(n), respectively?

```
void f1(n){
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         // O(1)
      }
   }
}
void f2(n){
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < i; j++) {
         // O(1)
      }
   }
}
```

A) O(n \log n), O(n \log n)
B) O(n^2), O(n^2)
C) O(n \log n), O(n \log i)
D) O(n^2), O(n*i)

Answer: B

For f2(n), outer loop executes n times, inner loop executes i times. Since i goes from 0 to n-1, inner loop has complexity O(n). Hence total complexity is O(n^2). Note that i is not a program input, but an internal temporary variable.

16. What is the time complexity of function f1(n) and function f2(n), respectively?

```
void f1(n){
   for (int i = 0; i < n; i++) {
      for (int j = 1; j < n; j *= 2) {
         // O(1)
      }
   }
}
void f2(n){
   for (int i = 0; i < n; i++) {
      for (int j = i; j >= 1; j /= 2) {
         // O(1)
      }
   }
}
```

A) O(n \log n), O(n \log n)
B) O(n^2), O(n^2)
C) O(n \log n), O(n \log i)
D) O(n^2), O(n*i)

Answer: A
(The outer loop runs in n, and the inner loop runs in \log n.)

17. What is the time complexity of function f(int[] arr) w.r.t. input array size n in Big-O notation?

```
int f(int[] arr) {
  int range = 100;
  int start = arr.length / 2 - range / 2;
  int sum = 0;
  for (int i = start; i < start + range; i++) {
    sum += arr[i];
  }
  return sum;
}
```

A) O(1)
B) O(\log n)
C) O(n)
D) O(n \log n)

Answer: A
(The loop executes a constant number of iterations (100), regardless of the size of the array arr. Therefore, the overall complexity of the method is O(1) (constant time), as it does not depend on the size of arr.)

18. Describe the worst-case running time of the following code in Big-O notation in terms of the variable n.

```
void f (int n) {
  for(int i=0; i < n; i++) {
    for(int j=0; j < 10; j++) {
      for(int k=0; k < n; k++) {
        for(int m=0; m < 10; m++) {
          System.out.println("!");
}}}}
}
```

Answer: O(n^2)
Outer Loop (i loop): executes n times.
Second Loop (j loop): Runs from 0 to 9, so it executes 10 times.
Third Loop (k loop): executes n times.
Innermost Loop (m loop): Runs from 0 to 9, so it executes 10 times.
To find the total number of iterations, we multiply the number of iterations of each loop:Total iterations = n×10×n×10=100n^2.
The dominant term in this expression is n^2, and constants are ignored in Big-O notation.
Therefore, the worst-case running time of the function in big-Oh notation is:O(n^2)

19. Describe the worst-case running time of the following code in Big-O notation in terms of the variable n.

```
int f(int n) {
int sum = 73;
for(int i=0; i < n; i++) {
   for(int j=i; j >= 5; j--) {
   //Alternative 1: for(int j=i; j >= 0; j--) {
   //Alternative 2: for(int j=0; j <= i; j++) {
   //Alternative 3: for(int j=0; j < 2i; j++) {
   //Alternative 4: for(int j=0; j < i2; j++) {
   //Alternative 5: for(int j=0; j < n2; j++) {
   //Alternative 6: for(int j=0; j < 1000000; j++) {
   sum--;
}}
return sum;
}
```

ANS: O(n^2) for Alternatives 1-3. O(n^3) for Alternatives 4-5. O(n) for Alternative 6, since total iterations is 1000000*n, and we drop all constants in big-O analysis.