1. Determine how many times the output statement is executed in each of the following fragments. Indicate whether the algorithm is O(n) or O(n2).

A white text with black numbers

Description automatically generated with medium confidence

1. O(n)
2. O(n2)
3. O(n2)
4. O(n)

2. Trace the execution of the following:

A black text with white text

Description automatically generated with medium confidence

What are the contents of an array after the execution of each

loop? After the first loop it produce 0 , 1, 2, 3, 4,5,6

after the second loop it produce 0, 1,2,3,5,7

3. Please provide analysis to calculate O(n) and T(n) for the following algorithms:

*a. Sum of an Array*

public static int sumArray(int[] array) {

int sum = 0; // 1 operation

for(int i = 0; i < array.length; i++) { // n iterations

sum += array[i]; // 2 operations (access and addition)

}

return sum; // 1 operation

}

(O(n)): 1 (initialization) + n \* 2 (iteration operations) + 1 (return) = 2n + 2 and (T(n)): O(n) since the number of operations is linear with respect to the size of the input array.

*b. Matrix Multiplication*

public static int[][] multiplyMatrices(int[][] firstMatrix, int[][] secondMatrix,

int r1, int c1, int c2) {

int[][] product = new int[r1][c2];

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

for (int k = 0; k < c1; k++) {

product[i][j] += firstMatrix[i][k] \* secondMatrix[k][j];

}

}

}

return product;

}

Number of Operations (O(n)): r1 \* c2 \* (c1 + 2) (nested loop operations + product update) Time Complexity (T(n)): O(r1 \* c2 \* c1) since the number of operations is cubic in terms of the dimensions of the matrices.

Please provide analysis to calculate O(n) and T(n) for the following algorithms:

*c. For Looping*

int result = 0;

for (int i = 0; i < n; i++) {

result = i + i;

}

for (int j = 0; j < n; j++) {

result = j + j;

}

for (int k = 0; k < n; k++) {

result = k + k;

}

Number of Operations (O(n)): 3 \* n \* 2 (assignment and addition in each loop) Time Complexity (T(n)): O(n) since the number of operations is linear with respect to the value of n.

*d. While Looping*

int i = n;

while (i > 0) {

int k = 2 \* i;

i = i / 2;

Number of Operations (O(n)): Logarithmic, specifically O(log n), as the value of i is halved in each iteration. Time Complexity (T(n)): O(log n) since the loop iterates in logarithmic time with respect to the initial value of i.

4. Please provide the examples of some well-known algorithms and their time complexity in Big O notation. Find at least one for each degree complexity.

1. O(1) - Constant Time Complexity (Accessing an element by index in an array.)

2. O(log n) - Logarithmic Time Complexity: Example: Binary Search in a sorted array.

3. O(n) - Linear Time Complexity: Example: Finding the maximum element in an unsorted array.

4. O(n log n) - Linearithmic Time Complexity: Example: Merge Sort algorithm.

5. O(n^2) - Quadratic Time Complexity: Example: Bubble Sort algorithm.

5. What is Abstract Data Type in data structure? Show the examples of the implementation in Java.

Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of values and a set of operations.

6. Please provide a summary of the difference between List and ArrayList in Java. Show it in the form of a table.

|  |  |
| --- | --- |
| List | Arraylist |
| Interface | Class(implement list interface) |
| Resizable | Resizable |
| Slower due to generalization | Faster due to array implementation |
| More memory overhead | Less memory overhead |
| Slower for inserting or deleting an element | Faster for inserting or deleting an element |

7. Write a program that does the following

• Create an ArrayList of integers, add the elements [12, 25, 34, 46] to it

• Remove the number 25 from the ArrayList

• Print the final ArrayList