Determining the time and cost of the algorithms

Time Complexity

Worst-Case Time Complexity:

- It refers to the maximum number of operations/steps required for an algorithm to complete execution across all possible legal inputs of size n.
- The worst-case is determined by the input that causes the algorithm to perform the maximum number of operations out of all possible inputs of size n.
- It gives an upper bound on running time to complete the algorithm and hence represents the maximum time required.
- For example, for linear search to find an element in an array, the worst case occurs when the element to be searched is not present in the array or is present at the last position. This causes the search to traverse the complete array resulting in a worst-case complexity of O(n).

Time Complexity

Average-Case Time Complexity:

- It refers to the average number of operations performed by an algorithm taken across all possible legal inputs of size n.
- Finding an exact average case requires integrating over all possible inputs and calculating the arithmetic mean of steps taken for each input instance.
- For most algorithms this calculation is complicated, so average case is generally estimated through analysis, simulations, probabilistic methods, assuming uniform random distribution of inputs.
- Average case gives a typical run time that can be expected for an algorithm. For the linear search example, if the elements are uniformly randomly distributed, then on average half the array will be traversed giving an average case complexity of O(n/2) which is still O(n).

Calculation of time complexity

```
if(Condition){          C1
    printf(“Hi”);       C2
}
else{                   C3
    printf(“Hello”);    C4
}
```

Breaking down the code

- IF condition cost = C1 (Executes once)
- IF statement cost = C2 (Executes if IF is true)
- ELSE condition cost = C3 (Executes once)
- ELSE statement cost = C4 (Executes if IF is false)

Now there are two possibilities:

1. If IF condition is true
    - IF condition cost = C1
    - IF statement cost = C2
2. Total cost = C1 + C2
3. If IF condition is false
    - IF condition cost = C1
    - ELSE condition cost = C3
    - ELSE statement cost = C4
4. Total cost = C1 + C3 + C4

Since the if condition can only be either true or false, the total cost will be:

Total Cost = Max(C1 + C2, C1 + C3 + C4)

By taking the max, we consider the costlier of the two possibilities (IF true or IF false).

Simplified, the total cost is:
Total Cost = Max(IF Cost, ELSE Cost)
Where:
IF Cost = C1 + C2

ELSE Cost = C1 + C3 + C4

So we take the maximum between the IF section cost or the ELSE section cost as the total cost.

It has constant time complexity O(1), as the costs C1, C2, C3, C4 are fixed, not dependent on any input size parameter.

Calculate the time complexity for this snippet

```
for(int i=1;i<5;i++) {        C1
printf("%d",i);               C2
}
```

Given the assumptions:

- Cost of the for loop overhead = C1 (This includes initialization, condition check, update statements)
- Cost of each iteration inside the loop body = C2
- Let's assume the loop runs N iterations.

So the costs will be:

1. For loop overhead
    - Runs only once
    - Cost = C1
2. Instructions inside body
    - Runs N iterations
    - Cost per iteration = C2
    - Total Cost = N * C2

Therefore, Total Cost = Cost of for loop + Cost of N executions of loop body = C1 + (N * C2) Where:

C1 = Constant cost of for loop overhead

C2 = Constant cost per loop iteration

N = Number of iterations

So we can model total cost as:

Total Cost = C1 + N*C2

Which shows that it is a linear function in terms of N with base costs C1 for the loop and C2 per iteration inside the loop body.

So the overall order is O(N) where N is number of iterations.

Calculate the time complexity for this snippet

```
for(int i=1;i<=M;i++){                    C1
        for(int j=1;j<=N;j++){            C2
                printf("Hello");          C3
}
}
```

Analyzing step by step

- Outer loop cost per iteration = C1
- Inner loop cost per iteration = C2
- Instruction inside inner loop cost per iteration = C3
- Let's assume:
  - Outer loop runs M iterations
  - Inner loop runs N iterations per each outer iteration

Then, costs will be:

1. Outer loop
   - Runs M iterations
   - Cost = M * C1
2. Inner loop
   - Runs N iterations
   - Runs M times (once per outer loop iteration)
   - Cost per inner loop = N * C2
   - Total inner loops = M (outer iterations) So total cost = M * (N * C2) = M * N * C2
3. Instructions inside inner loop
   - Have cost C3 per iteration
   - Run N times per inner loop
   - With M outer loops
   - So total times instructions execute = M * N
   - Hence total cost = M * N * C3

Total Cost = Outer Loop Cost + Inner Loop Cost + Instruction Cost

= M * C1 + M * N * C2 + M * N * C3

Therefore, the total cost is:
Total Cost = M(C1 + N*C2 + N*C3)
Where,

M = number of outer loop iterations

N = number of inner loop iterations

C1 = outer loop overhead per iteration

C2 = inner loop overhead per iteration

C3 = cost per iteration of instructions inside inner loop

This shows a polynomial time complexity of O(M*N) for nested loops.

Calculate time complexity

```
for(int i=1;i<=N;i++){                C
    if(condition){                    C1
        Statement;                    C2
    }
    else{                             C3
    Statement;                        C4
}
}
```

- Cost of for loop overhead = C (initialization, condition, update statements)
- If-else costs:
  - IF condition cost = C1
  - IF statement cost = C2
  - ELSE condition cost = C3
  - ELSE statement cost = C4
- Let the for loop run N iterations

Then, cost per iteration =
Max(C1 + C2, C1 + C3 + C4) (If-else cost)
And the for loop runs N iterations.
Therefore, total cost = Cost of N for loop iterations + N times (If-else cost per iteration)
Total Cost = C + N * Max(C1 + C2, C1 + C3 + C4)
Simplifying:
Total Cost = C + N * Max(IF Cost, ELSE Cost)
Where:
IF Cost = C1 + C2
ELSE Cost = C1 + C3 + C4
N = number of iterations
So overall the total cost is linear in terms of N (number of iterations).
Hence, the overall time complexity with if-else inside loop is O(N).

Question

```
m=1    C1 cost
for(i=1;i<=n;i++){   C2 cost
        for(j=1;j<=n;j++){  C3 cost
                m=i+j;     C4 cost
        }
}
return m; C5 cost
```

Counting executions per statement:

1. m = 1 (once) : Cost = C1
2. Outer loop initialization (i = 1): Cost = C2
3. Outer loop test (i <= n): Runs n+1 times Cost = (n+1)*C2
4. Inner loop initialization (j = 1): Runs n times (once per outer iteration) Cost = n*C3
5. Inner loop test (j <= n):
   Runs n^2 times Cost = n^2*C3
6. m = i + j: Runs n^2 times Cost = n^2*C4
7. Return statement: Runs once Cost = C5

Total Cost = C1 + (n+1)*C2 + n*C3 + n^2*C3 + n^2*C4 + C5*
= C1 + C2 + C3 + 2*n^2*C3 + n^2*C4 + C5

This is O(n^2) time complexity dominated by the nested loop iterations.

In the worst case, we consider the maximum number of iterations for each loop. For each iteration of the outer loop, the inner loop runs n times.

Total cost for worst case = C1 * n + C2 * n * C3 * n + C4 * n * n + C5

n the average case, we would typically consider the average number of operations. Since the code does not contain any conditional statements or variable input sizes, the average case is the same as the worst case.

Total cost for average case = C1 * n + C2 * n * C3 * n + C4 * n * n + C5

Therefore, both the worst case and average case time complexity for the given code are O(n^2).