

Beirut Arab University Course: Data Structure

Faculty of Science Instructor: Shayma Hicham Obeid

Mathematics and Computer Science Name: Moustafa Haydar

Semester: Fall 2024-2025 Id: 202403077

**Question1**:

[**https://github.com/PG-Moustafa/CMPS347\_DataStructure/tree/main/Lab\_3/Exercise\_1**](https://github.com/PG-Moustafa/CMPS347_DataStructure/tree/main/Lab_3/Exercise_1)

**Question 2: Time Complexity**

Find Big O for all examples:

***Example 1:***

|  |  |
| --- | --- |
| **Statement** |  |
| i=1; 1 |  |
| sum = 0; 1 |  |
| while (i <= n) { n+1 |  |
| j=1; n |  |
| while (j <= n) { n(n+1) |  |
| sum = sum + i; 2n^2 |  |
| j = j + 1; 2n^2 |  |
| } |  |
| i = i +1; 2n  } |  |

Total time = 1 + 1 + (n+1) + n + n(n+1) + 2n^2 + 2n^2 + 2n

= 5n^2 + 5n + 3

Then, O(n^2)

***Example 2:***

Here is the same code as above but using nested for loops instead of while loops:

|  |  |  |
| --- | --- | --- |
| **Statement** |  |  |
| sum = 0; 1 |  |  |
| for(i= 1; i ≤ n; i++) 2n+2 |  |  |
| for(j= 1; j ≤ n; j++) n(2n+2) |  |  |
| sum = sum + i; n\*2n |  |  |

Total time = 1 + (2n+2) + n(2n+2) + 2n^2

= 4n^2 + 4n + 3

Then, O(n^2)

***Example 3:***

|  |  |  |
| --- | --- | --- |
|  | **Cost** | **Times** |
| for (i=1; i<=n; i++) | C1 | (n+1) |
| for (j=1; j<=i; j++) | C2 | n(n+1)/2 |
| for (k=1; k<=j; k++) | C3 | |  | | --- | |  |   n(n+1)(n+2)/6 |
| x=x+1; | C4 | n(n+1)(n+2)/6 |

Then, O(n^3)

***Example 4: Selection sort using for-loops***

|  |  |  |
| --- | --- | --- |
|  | **Cost** | **Times** |
| for (i= n -1; i>=1; i--) | C1 | n+1 |
| Index=0 | C2 | n |
| for (j=1; j<=i; j++) | C3 | n(n+1)/2 |
| if(x[j] > x[index])  Index=j; | C4 | n(n+1)/2 |
| Exchange (x[index], x[i]) | C5 | n |

Then, O(n^2)

***Example 5: Bubble sort using for-loops***

|  |  |  |
| --- | --- | --- |
|  | **Cost** | **Times** |
| for (i= n -1; i>=1; i--) | C1 | n+1 |
| for (j=1; j<=i; j++) | C2 | n(n+1)/2 |
| if(x[j-1] > x[j]) | C3 | n(n+1)/2 |
| swap(x[j], x[j-1]) | C4 | n(n+1)/2 |

Then, O(n^2)

***Example 6:***

What is the time complexity of fun()?

int fun(int n)

{

int count = 0; 1

for (int i = n; i > 0; i /= 2) log2(n)

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

count += 1; n

return count; 1

}

Then, O(n)

***Example 7:***

All the following functions use a single for loop and within the for loop, the same set of statements are executed. Consider the following for loops:

A) for(i = 0; i < n; i++) O(n)

B) for(i = 0; i < n; i += 2) O(n)

C) for(i = 1; i < n; i \*= 2) O(log2(n))

D) for(i = n; i <= n; i /= 2) O(1)

If n is the size of input(positive), which function is most efficient?

Conclusion: D is the most efficient with a time complexity of O(1) (this loop only runs once)

Thank You! ☺