Data Structures Lab – Assignment 3

January 21, 2026

Assignment: Empirical Analysis of Time Complexity

Objective

The objective of this assignment is to experimentally analyze and compare the execution time of algorithms with different **asymptotic time complexities**. Implement the algorithms, measure their execution time for varying input sizes, and observe the relationship between input size and performance using graphical analysis.

Analyze the plotted graphs and compare the growth patterns of execution time for different algorithms. Comment on how the observed trends relate to the theoretical time complexities.

—

Input Sizes

Each program must be executed for the following values of N (Read the question to find out what is N):

$$N = \{5, 10, 20, 50, 100, 200, 500\}$$

—

Experimental Procedure

1. Implement each of the problems as separate programs.
2. Execute each program using the specified values of N.
3. Measure the execution time using an appropriate timing function.
4. Record the observed execution times in a table.
5. Plot graphs for each problem with:
   - X-axis: Input size (N)
   - Y-axis: Execution time

—

Q1: Binary Search ($O(\log N)$)

Given a sorted integer array of size N and a target element X, write a program to determine whether X exists in the array using binary search.

Task:

- Ensure the input array is sorted.
- Implement binary search to locate the target element.
- Record the execution time for each value of N.

Sample Case 1

Input

```
N = 5
Sorted Array = 1 3 5 7 9
X = 7
```

Output

```
Element found at index 3
```

Sample Case 2

Input

```
N = 10
Sorted Array = 2 4 6 8 10 12 14 16 18 20
X = 5
```

Output

```
Element not found
```

—

Q2: Linear Search ($O(N)$)

Given an integer array of size N and a target element X, write a program to determine whether X exists in the array using linear search.

Task:

- Search for the target element by examining elements sequentially.
- Record the execution time for each value of N.

Sample Case 1

Input

```
N = 5
Array = 4 7 1 9 3
X = 9
```

Output

```
Element found at index 3
```

Sample Case 2

Input

```
N = 10
Array = 2 5 8 1 6 9 3 7 4 10
X = 11
```

Output

```
Element not found
```

—

Q3: Pair Sum Problem ($O(N^2)$)

Given an integer array of size N and an integer value K, write a program to count the number of distinct index pairs (i, j) such that:

$$i < j \quad \text{and} \quad A[i] + A[j] = K$$

Task:

- Check all possible pairs in the array.
- Do not use additional data structures such as hash tables.
- Record the execution time for each value of N.

Sample Case 1

Input

```
N = 5
Array = 1 5 7 -1 5
K = 6
```

Output

```
Number of valid pairs = 3
```

Explanation: Valid pairs are (1,5), (1,5), and (7,-1).

Sample Case 2

Input

```
N = 10
Array = 2 4 3 5 7 8 9 1 6 0
K = 8
```

Output

```
Number of valid pairs = 4
```

—

Q4: Counting Common Elements Across Three Arrays ($O(N^3)$)

You are given three integer arrays A, B, and C, each of size N. Write a program to count the number of triplets (i, j, k) such that:

$$A[i] = B[j] = C[k]$$

Task:

- Compare elements across all three arrays.
- Do not use sorting or auxiliary data structures.
- Record the execution time for each value of N.

Sample Case 1

Input

```
N = 5
A = 1 2 3 4 5
B = 2 3 4 5 6
C = 3 4 5 6 7
```

Output

```
Number of common triplets = 3
```

Explanation: Common elements across all three arrays are 3, 4, and 5.

Sample Case 2

Input

```
N = 10
A = 1 2 3 4 5 6 7 8 9 10
B = 5 6 7 8 9 10 11 12 13 14
C = 7 8 9 10 15 16 17 18 19 20
```

Output

```
Number of common triplets = 4
```

—