

DSA Assignment-1 : Sorting and Binary Search

General note: You may use either C (with `scanf/printf`) or C++ (with `cin/cout`). Follow the sample code given in the drive/eLMS for how to take input and print output.

In all problems below, when the program runs, it should:

- First take the size of the array as input.
- Then take the elements of the array.
- Then take the element to search for.

Print the position of the element if found otherwise -1. While uploading to eLMS Kindly do not upload .zip file rather upload multiple files .

1. Binary search after Bubble Sort

Write a program that:

- (a) Reads an integer n (size of the array).
- (b) Reads n integers into an array.
- (c) Reads one more integer x , which is the element to search for.
- (d) Sorts the array in **ascending order** using **Bubble Sort**.
- (e) Performs **binary search** on the sorted array to search for x .
- (f) Prints whether x is found or not (and you may print its index if you want).

Example (just to show the input order): **Input:** 5 3 1 4 2 5 (Here, $n = 5$, array = {3, 1, 4, 2, 5}, $x = 5$)

2. Binary search after Insertion Sort

Copy your code from Problem 1. Now change only the sorting part as follows:

- (a) Instead of Bubble Sort, use **Insertion Sort** to sort the array in ascending order.
- (b) Keep the input format and the binary search part exactly the same.

So the program should:

- (a) Read n , then the array, then x .
- (b) Sort the array using **Insertion Sort**.
- (c) Do binary search to find x .
- (d) Print whether x is found.

3. Binary search after Selection Sort

Again, copy your code from Problem 1 (or Problem 2). Now change only the sorting algorithm:

- (a) Use **Selection Sort** to sort the array in ascending order.
- (b) Keep the input format and the binary search part exactly the same.

So the program should:

- (a) Read n , then the array, then x .
- (b) Sort the array using **Selection Sort**.
- (c) Do binary search to find x .
- (d) Print whether x is found.

4. Binary search on a descending sorted array

In this problem, the array is already sorted in **descending order**. Do **not** sort it again.

Write a program that:

- (a) Reads n .
- (b) Reads n integers into an array that is already sorted from **largest to smallest**.
- (c) Reads an integer x to search for.
- (d) Performs **binary search** on this **descending** sorted array to search for x . (Hint: The usual binary search condition changes slightly because the order is reversed.)
- (e) Prints whether x is found.

Example input order (values are just an example): **Input:** 5 9 7 5 3 1 7 Here, $n = 5$, array = {9, 7, 5, 3, 1} (already in descending order), $x = 7$.

5. Find the k -th smallest element in an array

In this problem, you will find the **k -th smallest element** from an unsorted array.

Write a program that:

- (a) Reads n , the size of the array.
- (b) Reads n integers into an array.
- (c) Reads an integer k .
- (d) Sorts the array in **ascending order** (you may use any sorting method).
- (e) After sorting, prints the element that is the **k -th smallest**. (Reminder: the 1st smallest is the minimum element, the 2nd smallest is the next one, and so on.)
- (f) If k is out of range (for example, $k > n$ or $k < 1$), print a suitable message.

Example: **Input:** 5 7 2 9 1 4 3 Here, $n = 5$, array = {7, 2, 9, 1, 4}, $k = 3$. Sorted array = {1, 2, 4, 7, 9}. The 3rd smallest element is **4**.