KAZAKH BRITISH TECHNICAL UNIVERSITY

LECTURE 3 – BINARY SEARCH

Prepared by: Izbassar Assylzhan

Course: Algorithms & Data Structures – Fall 2025

LECTURE 3

In this lecture, we'll learn **binary search** algorithm for finding target element or positions in given data, and **compare it** with naive **linear search** approach

TOPICS WE'LL COVER:

- Introduction to Searching
- Binary Search
- Common Pitfalls
- Problem Solving
- Applications & Takeaways

GOALS FOR THIS LECTURE:

- Understand the concept of Binary Search and why it is more efficient than Linear Search
- Learn how to implement Binary Search in C++ (iterative and recursive approaches)
- Apply Binary Search to solve typical problems, including handling duplicates and finding insertion positions

IDEAS BEHIND SEARCHING

Searching = locating an element in a collection.

Linear Search: check elements one by one → O(n).

Binary Search: works only on sorted arrays.

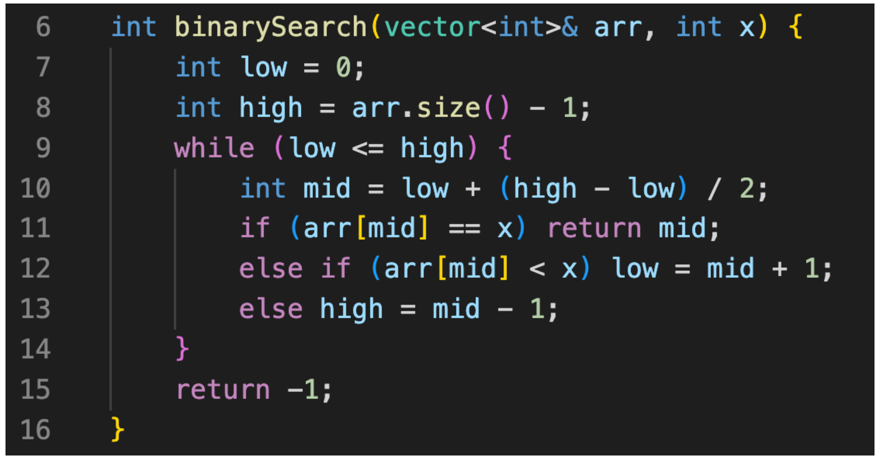
Idea: divide the search space in half each step.

Motivation: Faster search for large datasets.

BINARY SEARCH

The binary search have 5 target operations:

- Find the middle index
- Compare middle element with target
- If equal → found
- If target < mid → search left half
- If target > mid → search right half



```cpp
int binarySearch(vector<int>& arr, int x) {
    int low = 0;
    int high = arr.size() - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == x) return mid;
        else if (arr[mid] < x) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}
```

Time complexity: O(log n)

Space complexity: O(1) iterative, O(log n) recursive

Example: Find 7 in {1, 3, 5, 7, 9, 11}.

COMMON PITFALLS & VARIANTS

Forgetting to handle low <= high condition.
Infinite loop due to wrong mid calculation.

Mid calculation overflow:
- Wrong: (low + high) / 2
- Correct: low + (high - low) / 2

Handling duplicates (first/last occurrence).

Searching in an unsorted array → incorrect results.

KAZAKH BRITISH TECHNICAL UNIVERSITY

PROBLEM SOLVING

- Easy: Standard Binary Search in a sorted array.
- Medium 1: Find the first occurrence of a number in sorted array (with duplicates).
- Medium 2: Find the position to insert a number (lower bound).

APPLICATIONS & TAKEAWAYS

Binary Search is a Divide & Conquer strategy.

Used in:

- Searching in sorted arrays.
- std::lower_bound / std::upper_bound in C++.
- Optimization problems (binary search on answer).
- Finding square roots, peak elements, rotated arrays.

Key takeaway:

- Always ensure sorted data.
- Carefully handle edge cases.
- O(log n) efficiency makes it powerful for large datasets.

Q & A