**Searching in C**

Searching is a fundamental operation in computer science, involving finding a specific element within a dataset. In C programming, we commonly employ two primary searching techniques:

**1. Linear Search:**

* **Simple and straightforward:** Iterates through the array sequentially, comparing each element with the target value.
* **Time complexity:** O(n) in the worst case, where n is the number of elements.
* **Suitable for unsorted arrays:** Can be used on any array, regardless of order.

**2. Binary Search:**

* **Efficient for sorted arrays:** Requires the array to be sorted in ascending or descending order.
* **Divide and conquer approach:** Repeatedly divides the search space in half.
* **Time complexity:** O(log n), significantly faster than linear search for large datasets.

**Example: Linear Search**

**#include <stdio.h>  
  
int linearSearch(int arr[], int n, int x) {  
 for (int i = 0; i < n; i++) {  
 if (arr[i] == x) {  
 return i;  
 }  
 }  
 return -1;  
}  
  
int main() {  
 int arr[] = {2, 3, 4, 10, 40};  
 int x = 10;  
 int n = sizeof(arr) / sizeof(arr[0]);  
 int result = linearSearch(arr, n, x);  
 (result == -1) ? printf("Element is not present in array")  
 : printf("Element is present at index %d", result);  
 return 0;  
}**

**Example: Binary Search**

**#include <stdio.h>  
  
int binarySearch(int arr[], int l, int r, int x) {  
 if (r >= l) {  
 int mid = l + (r - l) / 2;  
 if (arr[mid] == x)  
 return mid;  
 if (arr[mid] > x)  
 return binarySearch(arr, l, mid - 1, x);  
 return binarySearch(arr, mid + 1, r, x);  
 }  
 return -1;  
}  
  
int main() {  
 int arr[] = {2, 3, 4, 10, 40};  
 int x = 10;  
 int n = sizeof(arr) / sizeof(arr[0]);  
 int result = binarySearch(arr, 0, n - 1, x);  
 (result == -1) ? printf("Element is not present in array")  
 : printf("Element is present at index %d", result);  
 return 0;  
}**

**Choosing the Right Search:**

* **Unsorted arrays:** Linear search is the only option.
* **Sorted arrays:** Binary search is significantly more efficient.