**Binary Search**

If we have an unsorted list from which we are trying to retrieve a particular element, performing a linear search will take linear time. However, if the list is sorted, and we have to perform searches over the list multiple times, binary search is a far superior option. For a single search, a linear search is still better.

A linear search is of the order O(n), which a binary search is of the order O(log2 n). If we want to sort an unsorted list and then perform a binary search over it, the order becomes O(nlog2 n).

#include<stdio.h>  
  
int main()  
{  
 int n = 11;  
 int arr[11] = {1, 1, 1, 35, 47, 47, 55, 100, 150, 150, 150};  
 int i, find, high = n-1, low = 0, mid, flag = 0;  
  
 for (i=0; i<n; i++) printf("%d ", arr[i]);  
  
 printf("\nEnter Number to Search: ");  
 scanf("%d", &find);  
  
 while (low <= high)  
 {  
 mid = (high + low)/2;  
 if (arr[mid] < find) low = mid + 1;  
 else if (arr[mid] == find)  
 {  
 printf("%d is in index %d.", find, mid);  
 high = mid;  
 low = mid;  
 while (arr[low-1] == arr[low]) low--;  
 while (arr[high+1] == arr[high]) high++;  
 if (low < 0) low = 0;  
 if (high > n-1) high = n;  
 printf("Lower Bound: %d. Upper Bound: %d.", low, high);  
 flag = 1;  
 break;  
 }  
 else high = mid - 1;  
 }  
  
 if (flag == 0)  
 printf("Not Found. Lower Bound = %d. Upper Bound = %d", arr[high], arr[low]);

}

C