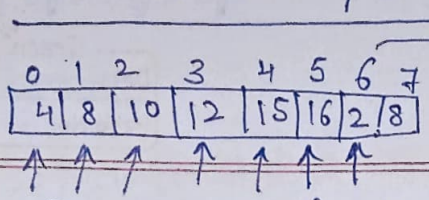
Lec-12 Linear VS Binary Search + Code in C Language:



Linear Search
→ Array. Traversal

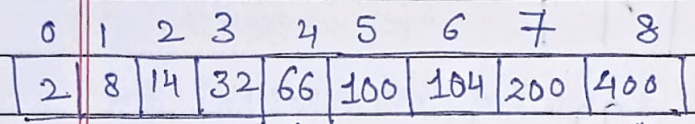
→ sorted + unsorted

WC O(n)

Binary search:

Condition
→ Array must be sorted

WC
O(log n)



To search 200

Low, Mid, High

Low	High	Mid	Elem found
0	8	4	No
4	8	6	No
6	8	7	yes

B.C = O(1)
W.C = log(n)

$\frac{6+8}{2} = 7$

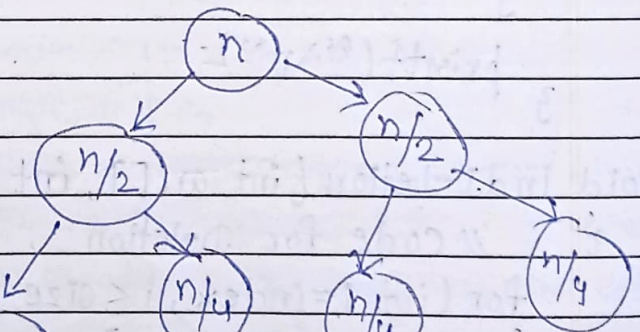
$\log(n) = \log(2^K)$

$\log(n) = K \log(2)$

$K = \frac{1}{\log 2} (\log(n))$

$K = K_2 \log(n)$

$\boxed{K = \log(n)}$



$n + \frac{n}{2} + \frac{n}{4} + \dots$

```
#include <stdio.h>
int linearSearch(int arr[], int size, int element){
    for (int i=0; i<size; i++)
    { if(arr[i]=element){
            return i;
    }
}
return -1;
}
int main(){
    int arr[]={1,3,5,56,4,3,23,5,4,56634,56,34};
    int size= sizeof(arr)/sizeof(int);
    int element= 4;
    int searchIndex= linearSearch(arr,size,element);
    printf("The element %d was found at index %d \n",
            element, searchIndex);
    return 0;
}
```

Linear Search

Green House
Date
Page No.

```
#include <stdio.h>
int binarySearch(int arr[], int size, int element){
    int low, mid, high;
    low=0;
    high= size-1;
    // start searching
    while(low<=high){
        mid= (low+high)/2;
        if (arr[mid]==element){
            return mid;
        }
        if(arr[mid]<element){
        low= mid+1;
    }
    else{ high= mid-1;
    }
    }
    // searching ends
    return -1;
}
```

Binary Search

```
int main()
{
    int arr[]={1,3,5,56,64,73,123};
    int size= sizeof(arr)/sizeof(int);
    int element= 56;
    int searchIndex= binarysearch
            (arr,size,element);
    printf("The element %d is found at index
            %d \n", element, searchIndex);
    return 0;
}
```