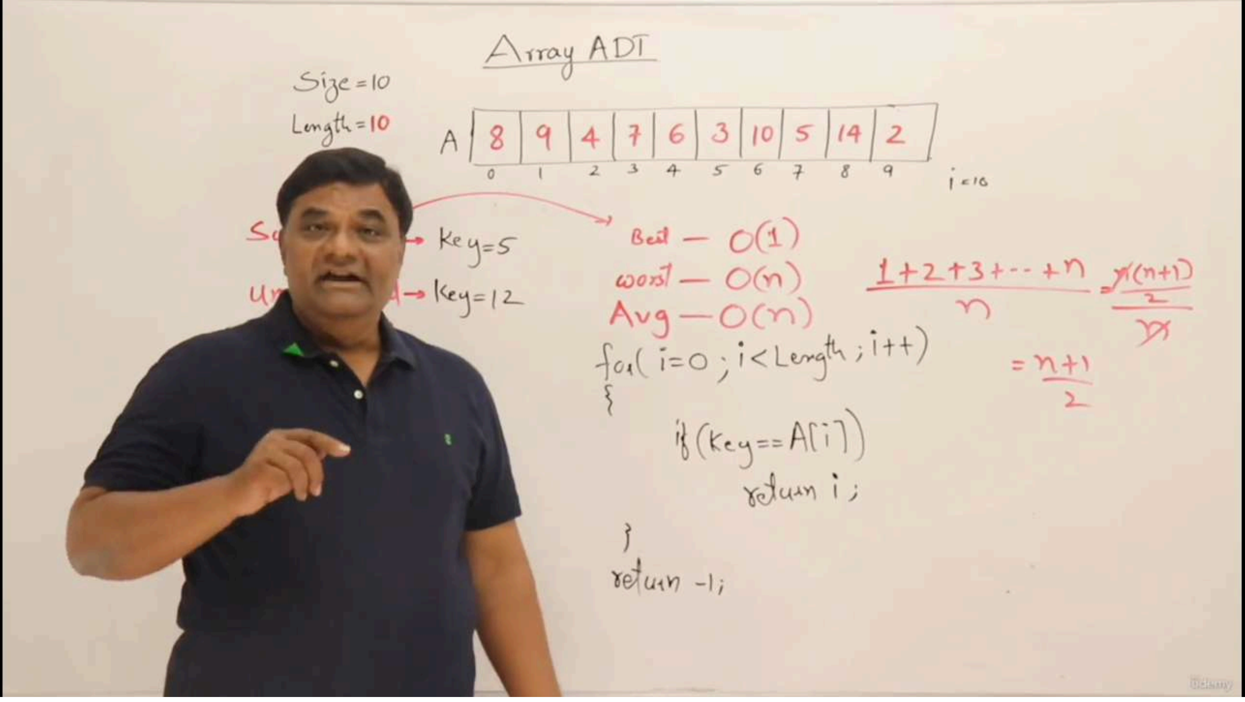


Array ADT

Size = 10

Length = 10

A	8	9	4	7	6	3	10	5	14	2
	0	1	2	3	4	5	6	7	8	9

$i = 10$

Su → Key = 5

Un d → Key = 12

Best — $O(1)$

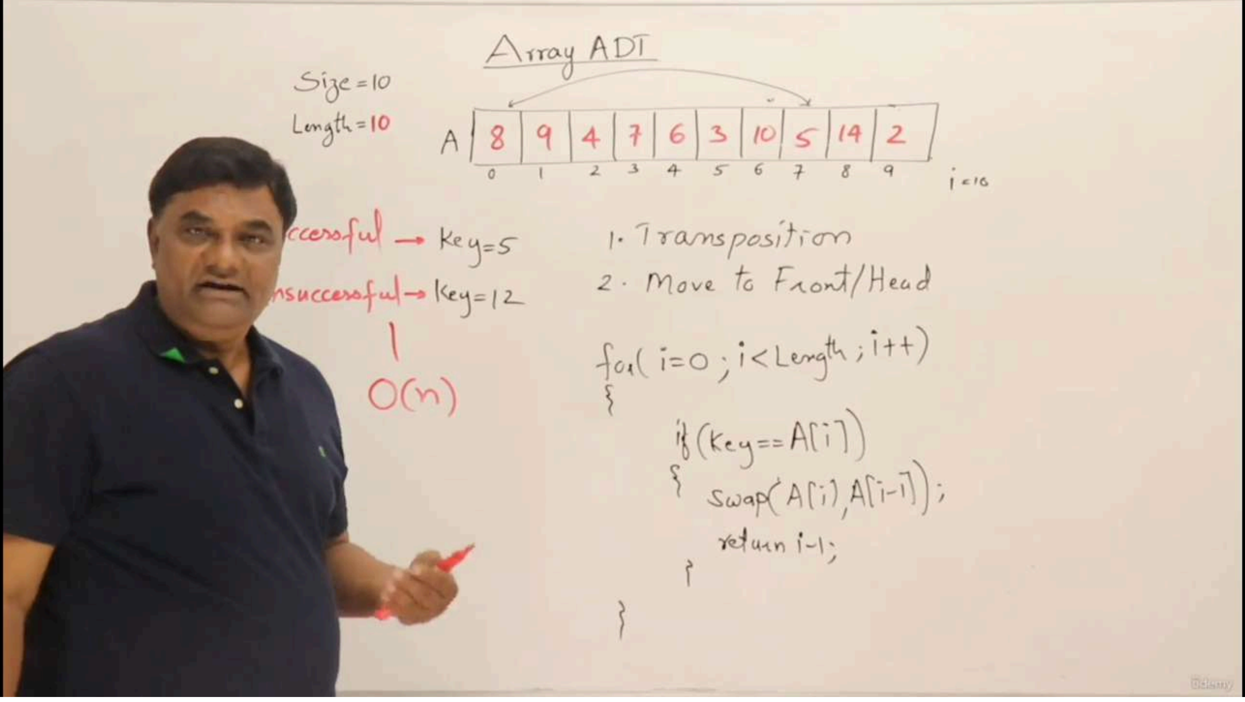
worst — $O(n)$

Avg — $O(n)$

$$\frac{1+2+3+\cdots+n}{n} = \frac{\frac{\cancel{n}(n+1)}{2}}{\cancel{n}} = \frac{n+1}{2}$$

```
for( i=0 ; i<Length ; i++)
{
      if(Key==A[i])
          return i;

}
return -1;
```



Array ADT

Size=10
Length=10

A	8	9	4	7	6	3	10	5	14	2	
	0	1	2	3	4	5	6	7	8	9	i=10

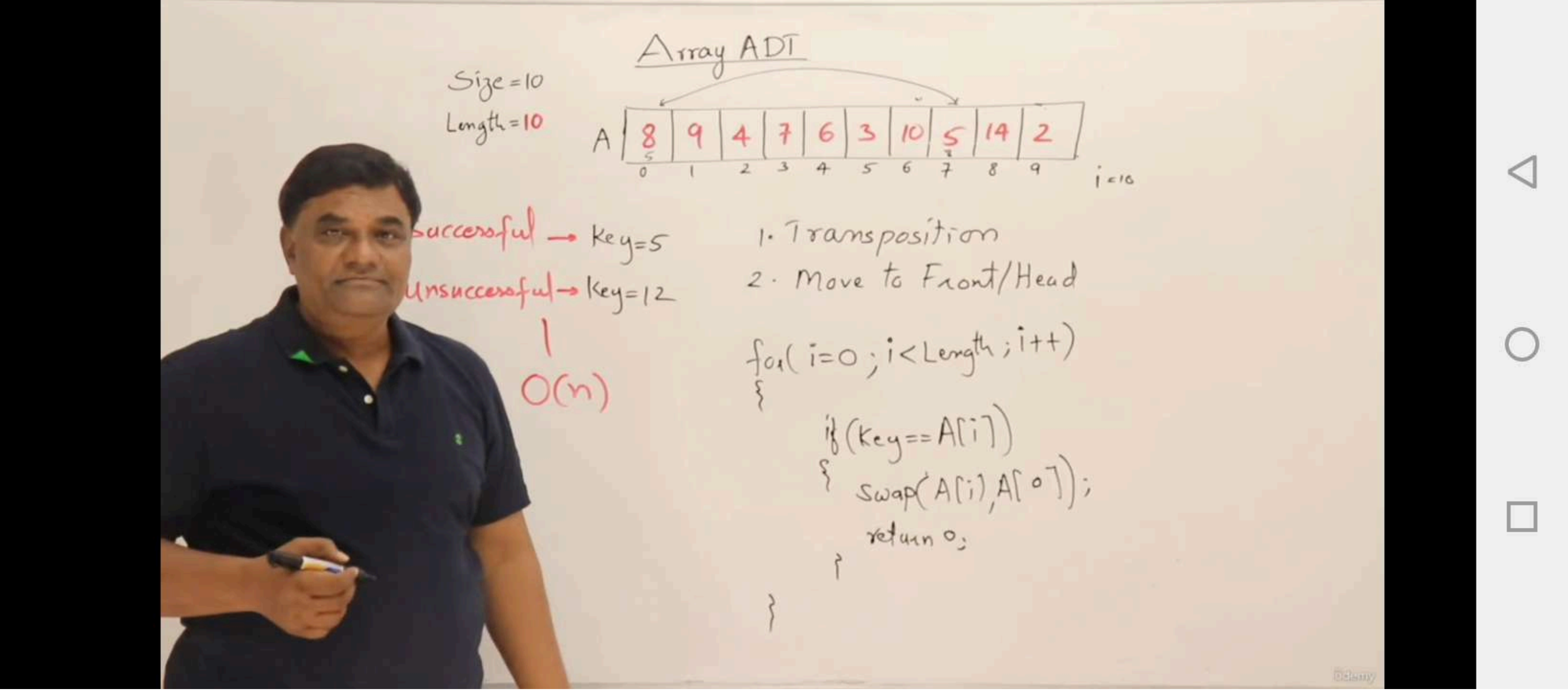
ccessful → Key=5

nsuccessful → Key=12

$O(n)$

1. Transposition
2. Move to Front/Head

```
for( i=0 ; i<Length ; i++)
{
     if (Key==A[i])
     {
       swap(A[i],A[i-1]);
       return i-1;
     }
}
```



Array ADT

Size=10

Length=10

A	8	9	4	7	6	3	10	5	14	2
	0	1	2	3	4	5	6	7	8	9

i=10

Successful → Key=5

Unsuccessful → Key=12

$O(n)$

1. Transposition
2. Move to Front/Head

```
for(i=0;i<Length;i++)
{
    if(Key==A[i])
    {
      swap(A[i],A[0]);
      return 0;
    }
}
```

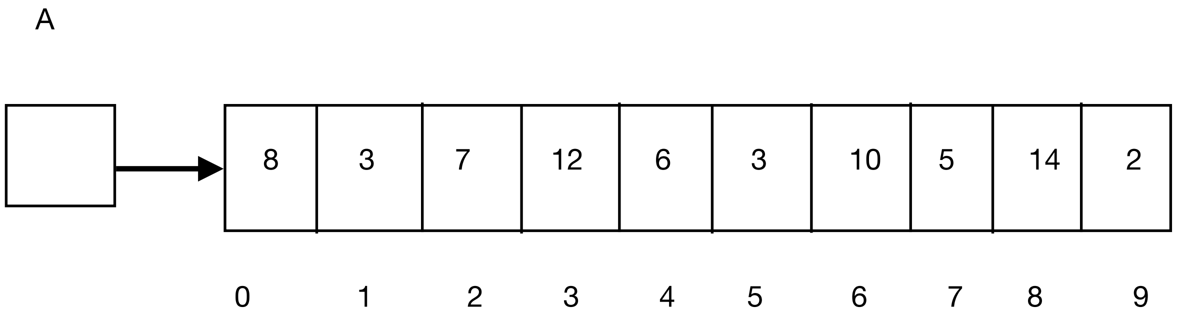
Linear Search

- They are 2 search method in an array

I. Linear search
II. Binary search

- Linear search :

Size = 10
Length = 10



Key = 5 (successful search)
Key = 12 (unsuccessful search)

- All the elements must be unique here
- The value you are searching is called key, In linear search we search the key element one by one linearly
- We search the element by comparing it with the key value

- The result of the search is the location of the element where its present (index number) , it is very useful in accessing the element in the list

- If the element is not found throughout the list that means it is not present in the list therefore search is unsuccessful

Syntax :

```
for( i = 0;  i < length ; i++ )
{
        if( key == A[ i ] )
        return i;                        //if search is successful it ends here
}

return -1;                               // if search unsuccessful returns -1
```

Improving Linear Search

- When you are searching for a key element there is a possibility that you are searching the same element again
- To improve the speed of comparison , you can move a key element repeatedly search one step forward this method is called transposition

Syntax :

```
for( i = 0; i < length ; i++ )
        {
                if( key == A[ i ] )
                {
                        swap( A[i], A[i-1]);
                        return i-1;
                }
        }
```

- The second method is you can directly swap the key element to the first element this process is called move to head . The next search for the same element becomes faster.

```
for( i = 0; i < length ; i++ )
    {
        if( key == A[ i ] )
        {
            swap( A[i], A[0]);
            return 0;
        }
    }
```

Searching in a Array

```c
#include<stdio.h>
struct Array
{
    int A[10];
    int size;
    int length;
};
    void Display(struct Array arr)
    {
        int i;
        printf("\nElements are\n");
        for(i=0;i<arr.length;i++)
            printf("%d ",arr.A[i]);
    }
 void swap(int *x,int *y)
 {
     int temp=*x;
     *x=*y;
     *y=temp;
 }

int LinearSearch(struct Array *arr,int key)
{
    int i;
    for(i=0;i<arr->length;i++)
    {
        if(key==arr->A[i])
        {
            swap(&arr->A[i],&arr->A[0]);
            return i;
        }
    }
    return -1;
}


int main()
{

    struct Array arr1={{2,23,14,5,6,9,8,12},10,8};
    printf("%d",LinearSearch(&arr1,14));
    Display(arr1);
    return 0;
}
```