Searching Algorithm

Lecture 4

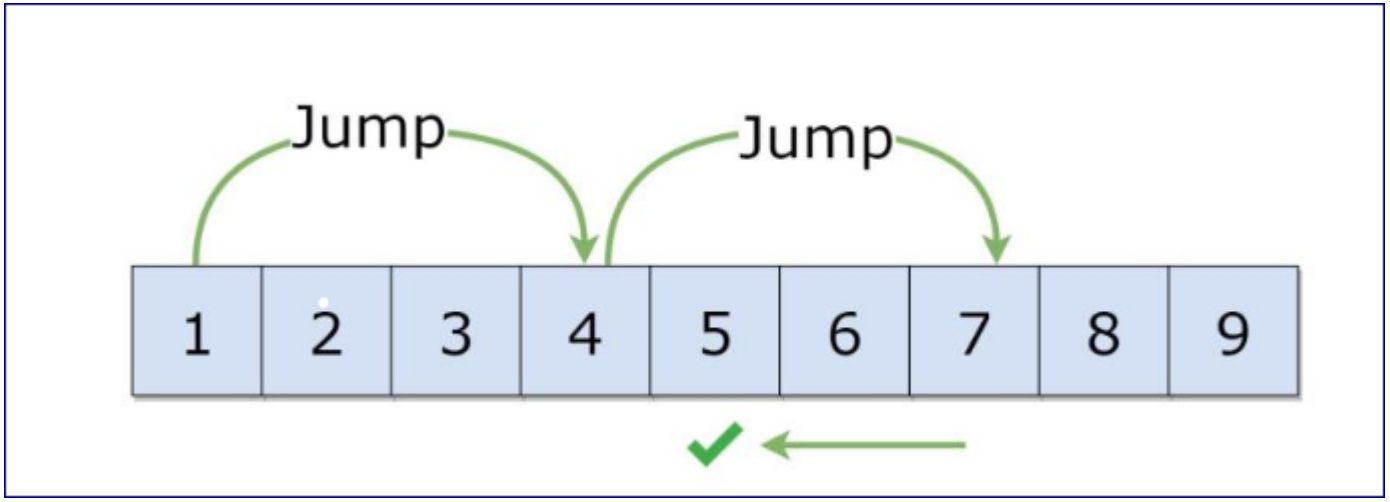
Outlines

- **Jump Search**
- **Exponential Search**

Jump Search

- Like Binary Search, Jump Search is a searching algorithm for sorted arrays.
- The basic idea is to check fewer elements (than linear search) by jumping ahead by fixed steps or skipping some elements in place of searching all elements. For example, suppose we have an array arr[] of size n and block (to be jumped) size m. Then we search at the indexes arr[0], arr[m], arr[2m]…..arr[km] and so on. Once we find the interval (arr[km] < x < arr[(k+1)m]), we perform a linear search operation from the index km to find the element x.

Jump Search



the optimal jump size is:

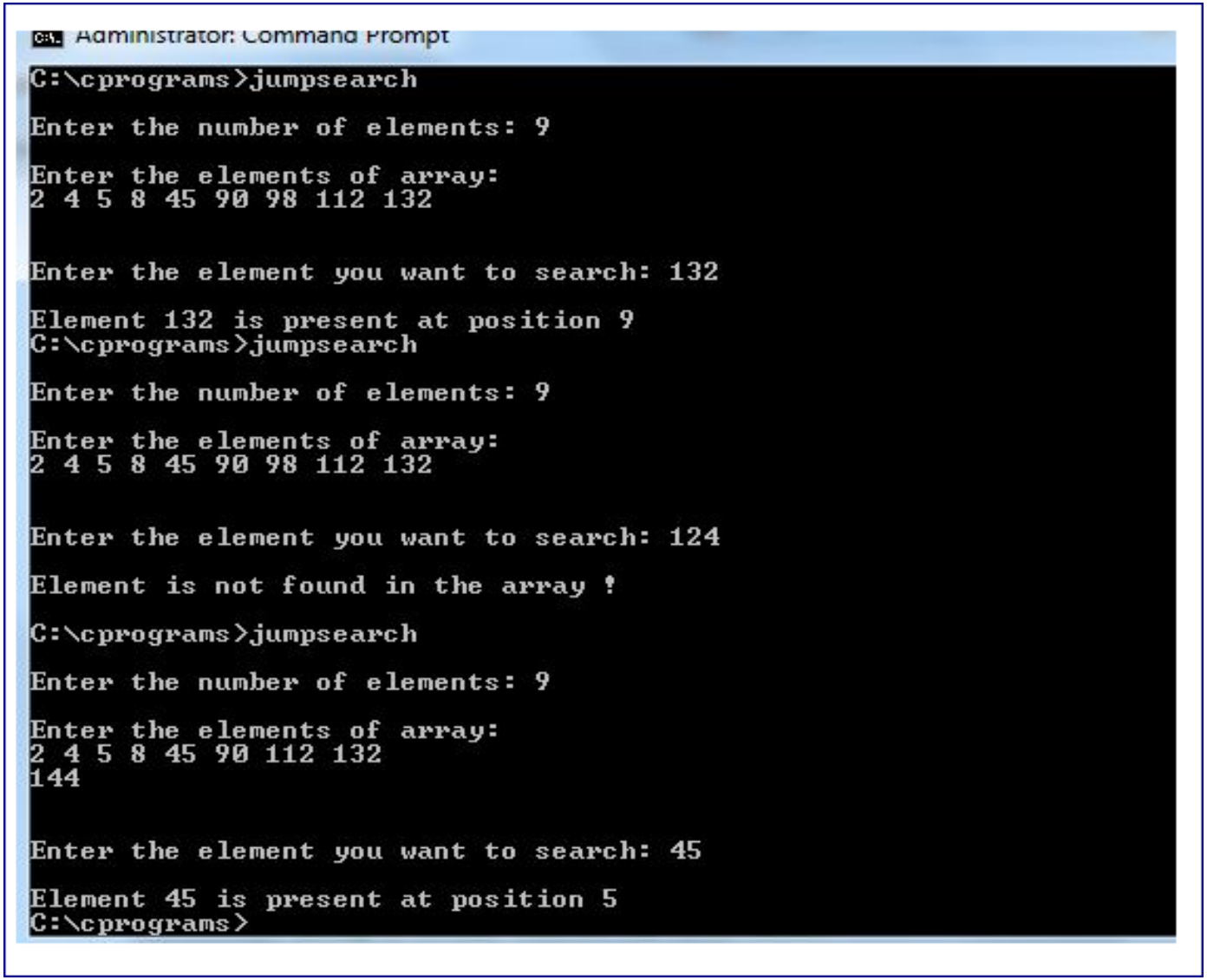
$$m = \sqrt{n}$$

```
#include <stdio.h>
#include <math.h>
#define MAX 100
int find_element(int element);
int arr[MAX],n;
int main()
{
    int i,element,result;
  printf("\nEnter the number of elements: ");
  scanf("%d",&n);
  printf("\nEnter the elements of array: \n");
  for(i=0;i<n;i++)
  {       scanf("%d",&arr[i]);
  }
    printf("\n\nEnter the element you want to search: ");
    scanf("%d",&element);
    result=find_element(element);
    if(result==-1)
    {           printf("\nElement is not found in the array !\n");
    }     else
    {           printf("\nElement %d is present at position %d",element,result);
    }
    return 0;
}
```

```
int find_element(int element)
{   int jump_step,prev=0;
   jump_step=floor(sqrt(n));

   /* Finding block in which element lies, if it is present */
    while(arr[prev]<element)
   {       if(arr[jump_step]>element || jump_step>=n)
       {
            break;
       }       else
       {                prev=jump_step;
             jump_step=jump_step+floor(sqrt(n));
       }
   }
    /*Finding the element in the identified block */

   while(arr[prev]<element)
   {
        prev++;
   }
    if(arr[prev]==element)
   {       return prev+1;
   }   else
   {       return -1;   }}
```


Administrator: Command Prompt
C:\cprograms>jumpsearch
Enter the number of elements: 9
Enter the elements of array:
2 4 5 8 45 90 98 112 132
Enter the element you want to search: 132
Element 132 is present at position 9
C:\cprograms>jumpsearch
Enter the number of elements: 9
Enter the elements of array:
2 4 5 8 45 90 98 112 132
Enter the element you want to search: 124
Element is not found in the array !
C:\cprograms>jumpsearch
Enter the number of elements: 9
Enter the elements of array:
2 4 5 8 45 90 112 132
144
Enter the element you want to search: 45
Element 45 is present at position 5
C:\cprograms>

Exponential Search

- Exponential search involves two steps:
  - Find range where element is present
  - Do Binary Search in above found range
- How to find the range where element may be present?
  - The idea is to start with subarray size 1 compare its last element with x, then try size 2, then 4 and so on until last element of a subarray is not greater. Once we find an index i (after repeated doubling of i), we know that the element must be present between i/2 and i.

Exponential Search

- Exponential Binary Search is particularly useful for unbounded searches, where size of array is infinite. Please refer Unbounded Binary Search for an example.
- It works better than Binary Search for bounded arrays also when the element to be searched is closer to the first element.
  - .

Computational Complexity

- linear search -O(n)
- Binary search has complexity O(log n).
- Jump Search has complexity O(√N).
- Exponential Binary Search O(Log n).


LTE