Sequential Search

SCSJ2013 Data Structures & Algorithms

Nor Bahiah Hj Ahmad & Dayang Norhayati A. Jawawi

Faculty of Computing

Objectives

At the end of the class, students are expected to

1. understand the implementation of sequential search on unsorted data and sorted data
2. analyze the efficiency of the searching techniques

Basic Sequential Search

- to search item from unsorted list or array.
- For small size of list
  - because the efficiency of sequential search is low
- In a sequential search:

1. Every element in the array will be examine sequentially, starting from the first element.
2. The process will be repeated until the last element of the array or until the searched data is found.

Basic Sequential (BS) Search

- Used for searching that involves records stored in the main memory (RAM)
- The simplest search algorithm, but is also the slowest
- Searching strategy:

Examines each element in the array one by one and compares its value with the key

Successful if the search key matches

else, if no matches is found, the search process is continued to the last element of the array

Basic Sequential Search Function

```
int SequenceSearch( int   search_key,
                    const int array [ ]
                    int   array_size )
{   int p;
    int index =-1; //-1 means record is not found
    for ( p = 0; p < array_size; p++ ){
        if ( search_key == array[p] ){
            indeks = p;//assign current array index
            break;
        }//end if
     } //end for
   return index;
} //end function
```

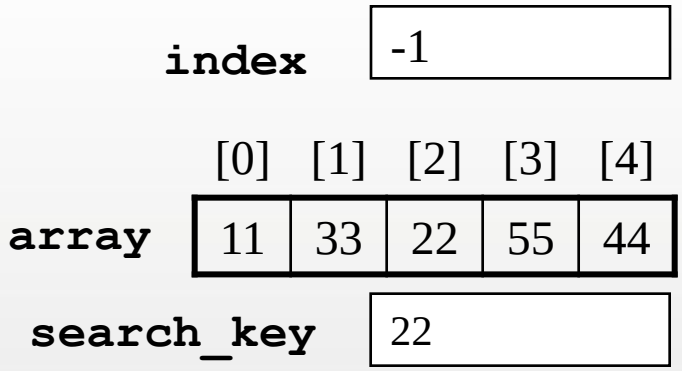
Every element in the array will be examined until the search key is found

or until the search process has reached the last element of the array

BS Search implementation – Search key = 22

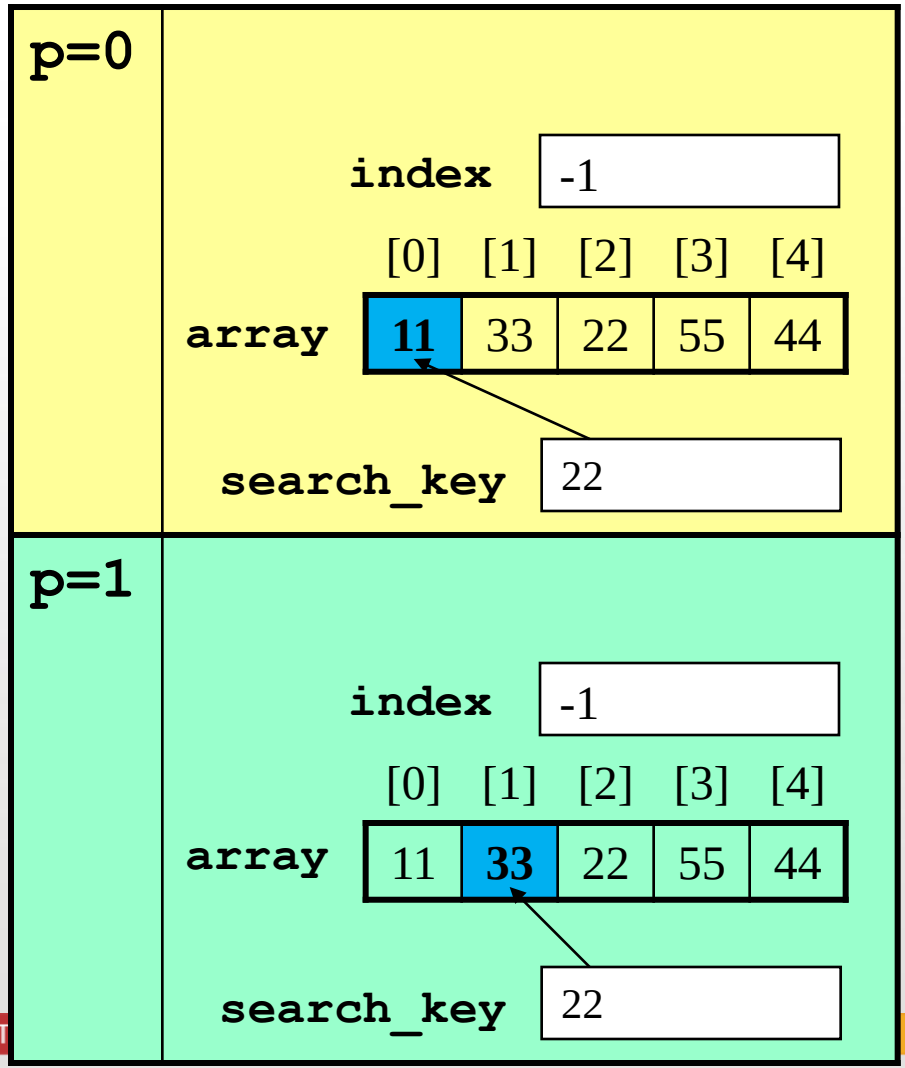
```
int SequenceSearch( int   search_key,
                    const int array [ ],
                    int   array_size )
{   int p;
    int index =-1;
    //-1 means record is not found
    for ( p = 0; p < array_size; p++ ){
       if ( search_key == array[p] ){

           indeks = p;
          //assign current array index
           break;
       }//end if
     } //end for
   return index;
} //end function
```



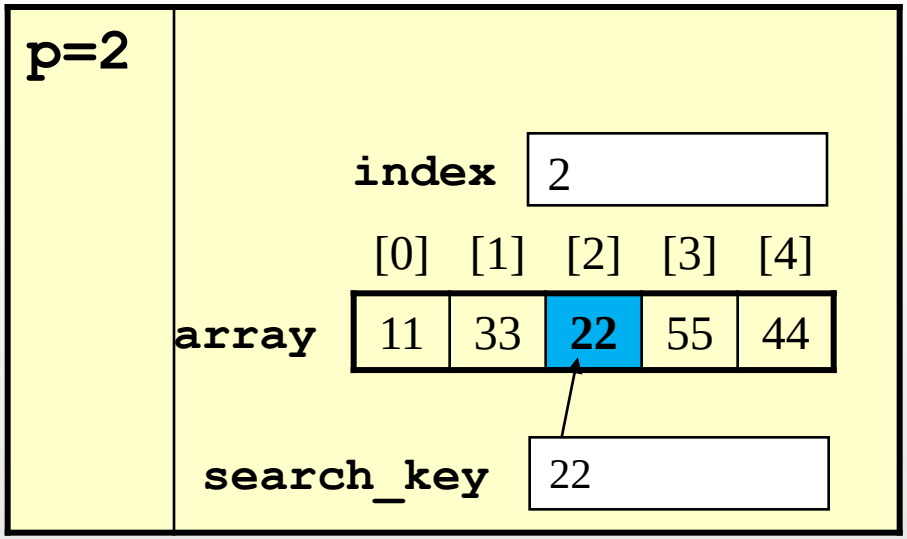
BS Search implementation – Search key = 22

```
int SequenceSearch( int   search_key,
                    const int array [ ],
                    int   array_size )
{ int p;
  int index =-1;
  //-1 means record is not found
  for ( p = 0; p < array_size; p++ ){
     if ( search_key == array[p] ){
          indeks = p;
         //assign current array index
          break;
    }//end if
  } //end for
   return index;
} //end function
```



BS Search implementation – Search key = 22

```
int SequenceSearch( int   search_key,
                    const int array [ ],
                    int   array_size )
{ int p;
  int index =-1;
  //-1 means record is not found
  for ( p = 0; p < array_size; p++ ){
     if ( search_key == array[p] ){
          indeks = p;
         //assign current array index
          break;
    }//end if
  } //end for
   return index;
} //end function
```

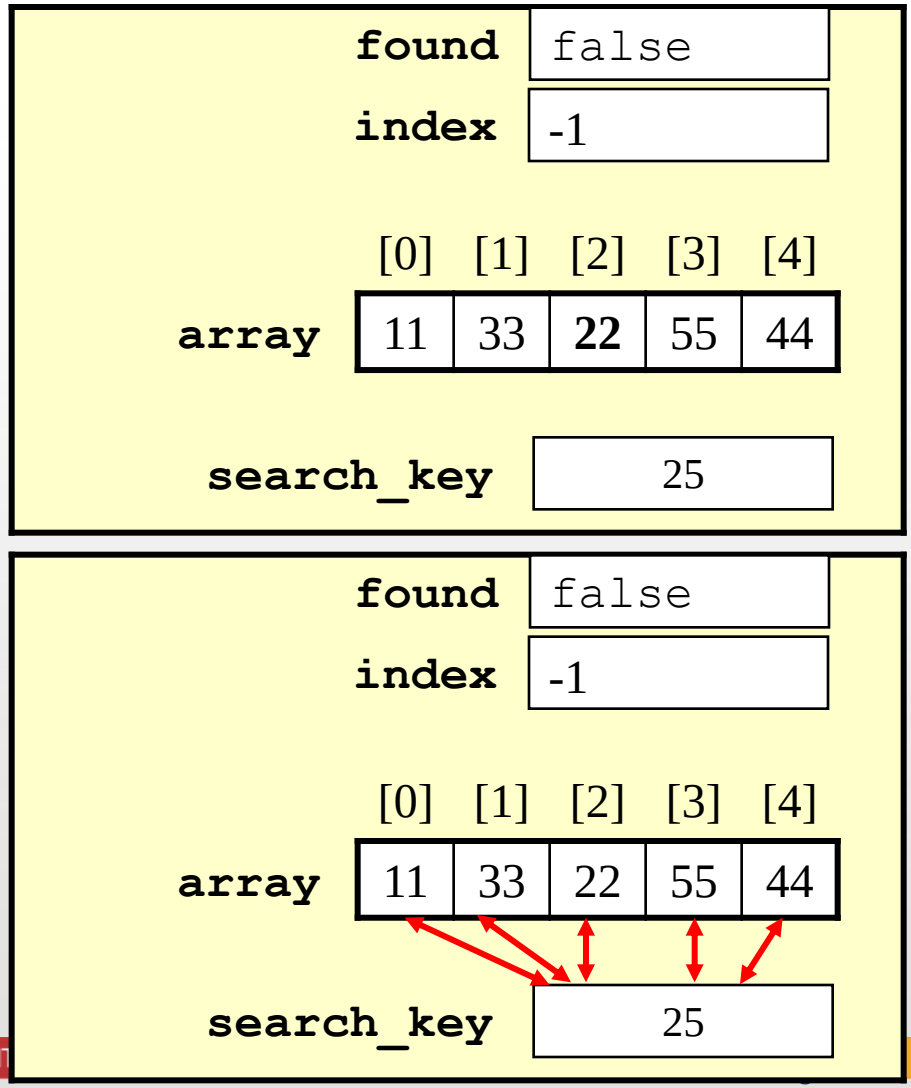


Search for key 22 is successful & return 2

BS Search implementation – Search key = 25

```
int SequenceSearch( int   search_key,
                    const int array [ ],
                    int   array_size )
{ int p;
  int index =-1;
  //-1 means record is not found
  for ( p = 0; p < array_size; p++ ){
     if ( search_key == array[p] ){
         indeks = p;
        //assign current array index
         break;
    }//end if
  } //end for
   return index;
} //end function
```

p=0,1,2,3,4 => search key is not matches
Search is unsuccessful



Sequential Search Analysis

- Searching time for sequential search is O(n).
  - the searched key is located at the end the list or
  - the key is not found

- If the list can be found at index 0, then searching time is, O(1).

Improvement of Basic Sequential Search Tech.

- **Problem:**
  - Search key is compared with all elements in the list, O(n) time consuming for large datasets.

- **Solution:**
  - Improved the efficiency –s earching on a sorted list.
  - For searching on ascending list, search key until :
    1. found.
    2. Or key value is smaller than the item compared

=> This will minimize the searching process.

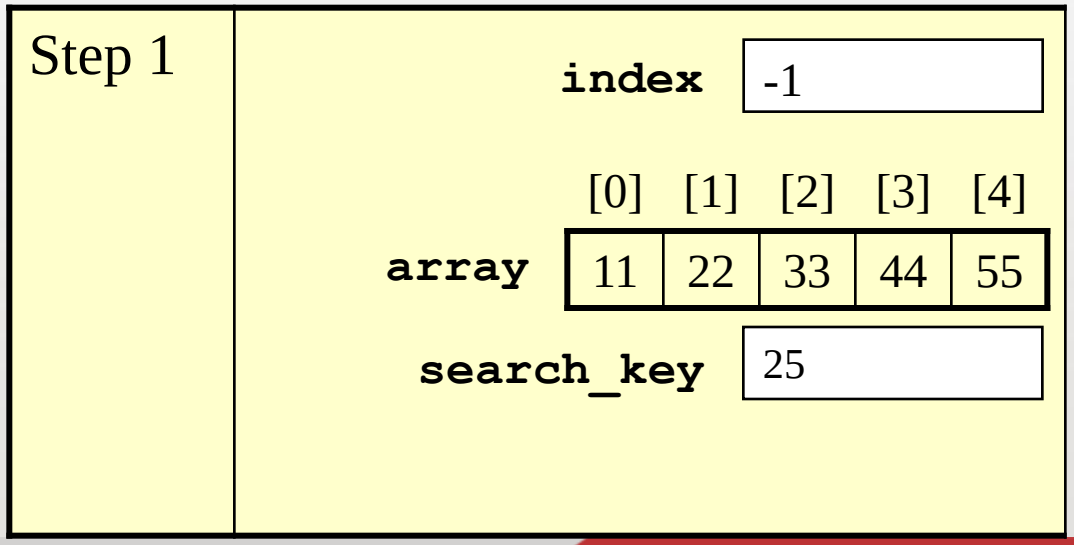
Sequential Searching on Sorted Data

```
int SortedSeqSearch ( int search_key, const int
array[ ],
                    int array_size)
{   int p;
    int index = -1; //-1 means record not found
    for ( p = 0; p < array_size; p++ )
    {   if (search_key < array [p] )
              break;
             // loop repetition terminated
             // when the value of search key is
             // smaller than the current array element
         else if (search_key == array[p])
         {
              index = p;  // assign current array index
              break;
         } // end else-if
    }//end for
    return index; // return the value of index
} //end function
```

Steps to Execute Sequential Search Function on a Sorted List

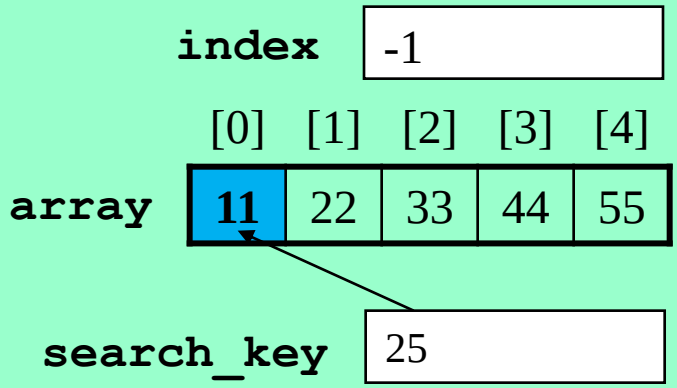
Assume:

- `search_key = 25`
- `array_size = 5`

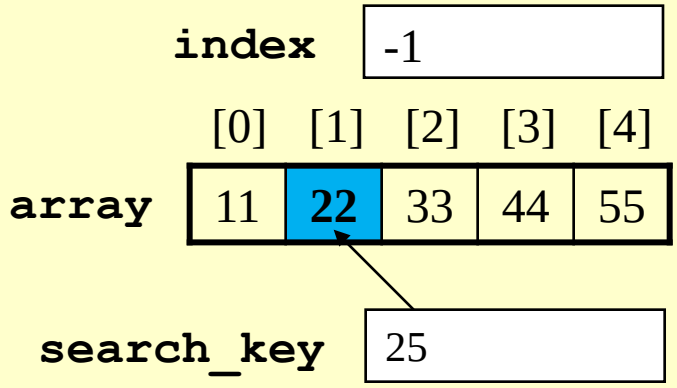


Steps to Execute Sequential Search Function on a Sorted List

Step 2

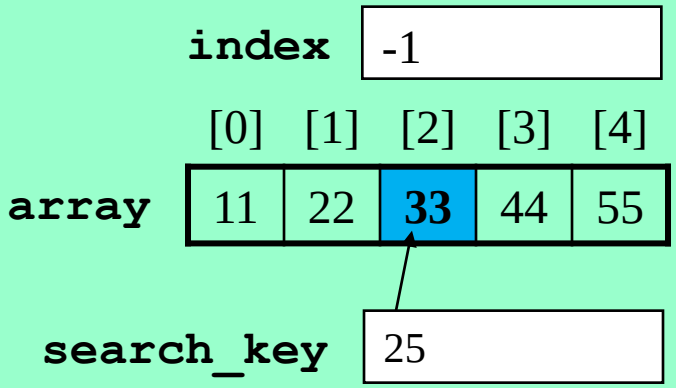


Step 3



Steps to Execute Sequential Search Function on a Sorted List

Step 4



Summary

- If the elements in the list is not in a sorted (asc/desc) order, loop will be repeated based on the number of elements in the list
- Sequential search on sorted data is more efficient than sequential search on unsorted data.
- If the list is sorted in descending order, change operator "<" to operator ">" in the loop `for`


Thank
You