UTM
UNIVERSITI TEKNOLOGI MALAYSIA

DATA STRUCTURE & ALGORITHM

SEARCHING TECHNIQUES 1

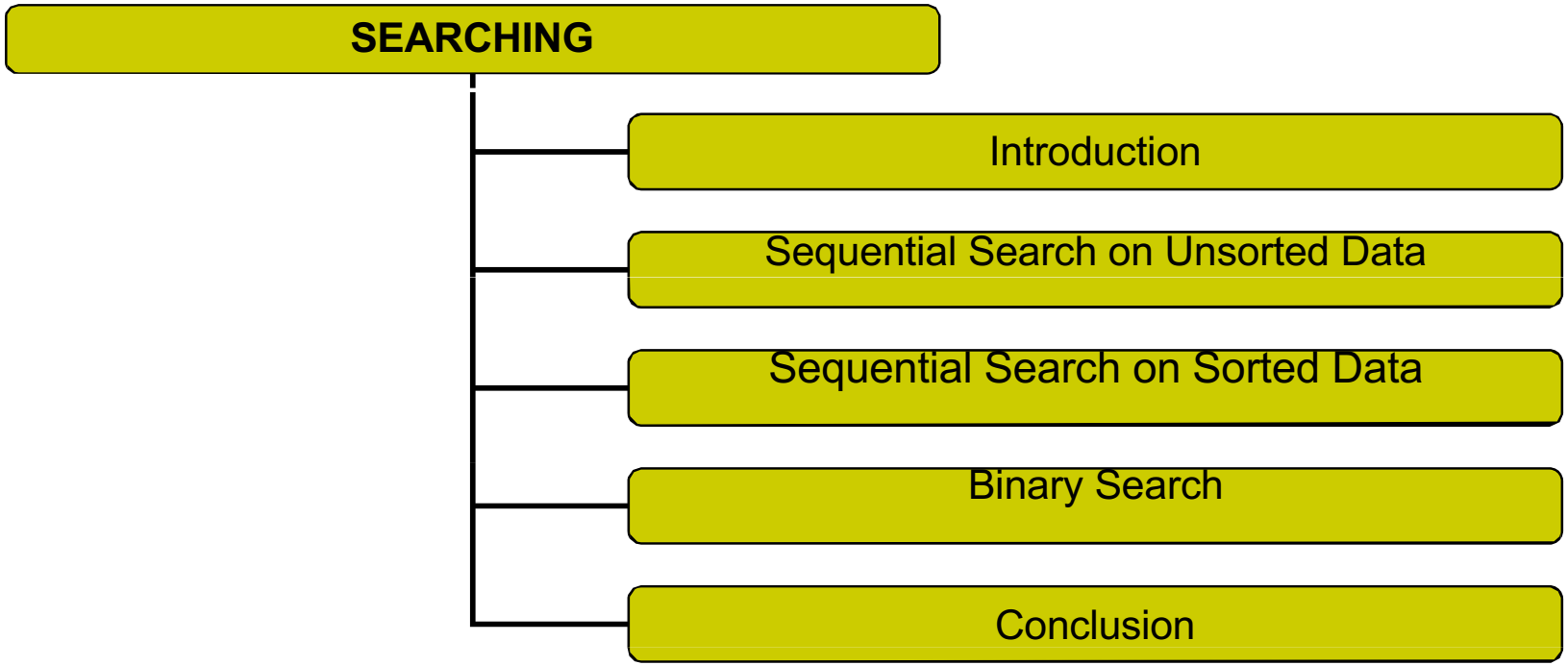
Nor Bahiah Hj Ahmad & Dayang Norhayati A.Jawawi
School of Computing

Objective

At the end of the lecture, students are expected to be able to :

- Understand the searching technique concept and the purpose of searching operation.
- Understand the implementation of basic searching algorithm;
  - Sequential search.
    - Sequential search on unsorted data.
    - Sequential search on sorted data.
  - Binary Search.
- Able to analyze the efficiency of the searching technique.
- Able to implement searching technique in problem solving

Outline


SEARCHING
Introduction
Sequential Search on Unsorted Data
Sequential Search on Sorted Data
Binary Search
Conclusion

Introduction

Searching Definition

- Clifford A. Shaffer[1997] define searching as a process to determine whether an element is a member of a certain data set.
- The process of finding the location of an element with a specific value (key) within a collection of elements
- The process can also be seen as an attempt to search for a certain record in a file
  - Each record contains data field and key field
  - Key field is a group of characters or numbers used as an identifier for each record
  - Searching can done based on the key field.

Example: Employee Record

index	empID	Empl_IC	EmpName	Post
[0]	1111	701111-11-1234	Ahmad Faiz Azhar	Programmer
[1]	**122**	**800202-02-2323**	**Mohd. Azim Mohd. Razi**	**Clerk**
[2]	211	811003-03-3134	Nurina Raidah Abdul Aziz	System Analyst

Searching can be done based on certain field:

empID, or Empl_IC, or EmpName

To search empID = 122, give us the record value at index 1.

To search empID = 211, give us the record value at index 2

5

Introduction

- Among Popular Searching Technique
  - Sequential search *
  - Binary search *
  - **Binary search Tree** **
  - Indexing.

- Similar with sorting, Searching can also be implemented in two cases, internal and external search.

* topic discuss in this chapter

** topic discussed in later chapter

Introduction

- External search – only implemented if searching is done on a very large size of data. Half of the data need to be processed in RAM while half of the data is in the secondary storage.
- Internal search – searching technique that is implemented on a small size of data. All data can be load into RAM while the searching process is conducted.
- This session will only discuss several searching techniques on the data stored in an array.

Basic Sequential Search

- Basic sequential search usually is implemented to search item from unsorted list/ array.
- The technique can be implemented on a small size of list.
- This is because the efficiency of sequential search is low compared to other searching techniques.
- In a sequential search, every element in the array will be examine sequentially, starting from the first element.
- The process will be repeated until the last element of the array or until the searched data is found.

Basic Sequential 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 (sequentially) and compares its value with the one being looked for – the search key.
  - Search is successful if the search key matches with the value being compared in the array. Searching process is terminated.
  - Else,if no matches is found, the search process is continued to the last element of the array.
  - Search is failed if there is no matches found from the array.

Basic Sequential Search Function

```
int SequenceSearch(int search_key,
  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; //terminate searching
        }
     } //end for
  return index;
  // return location of value
} //end function
```

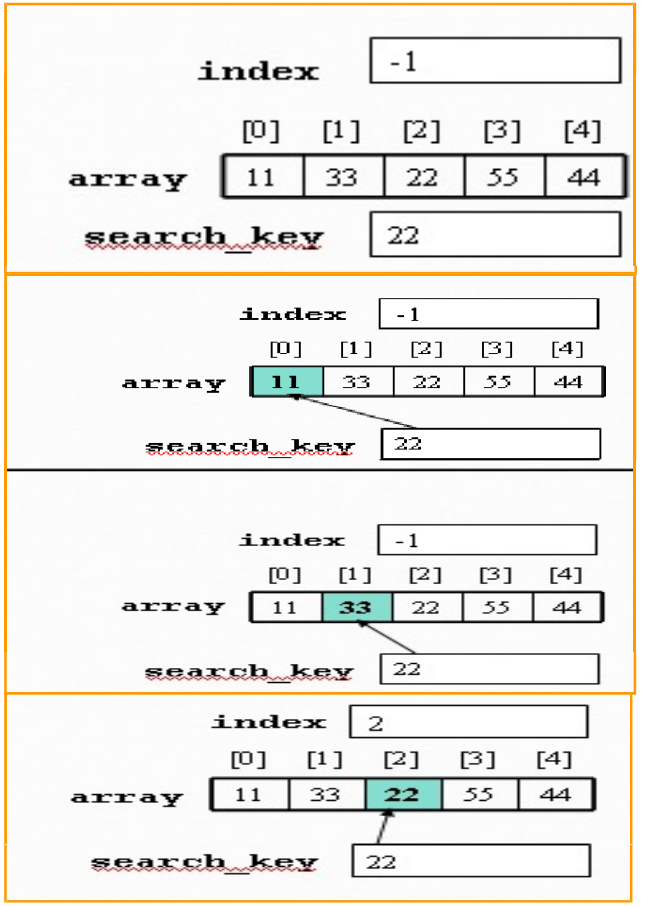
Every element in the array will be examined until the search key is found

or

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

Basic Sequential Search implementation – Search key = 22

```
int SequenceSearch(int search_key,
  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; //terminate searching
     }// end if
   } //end for
  return index;
  // return location of value
} //end function
```

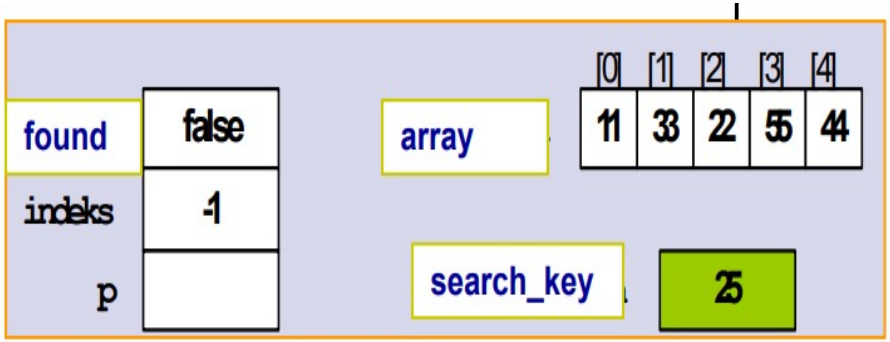


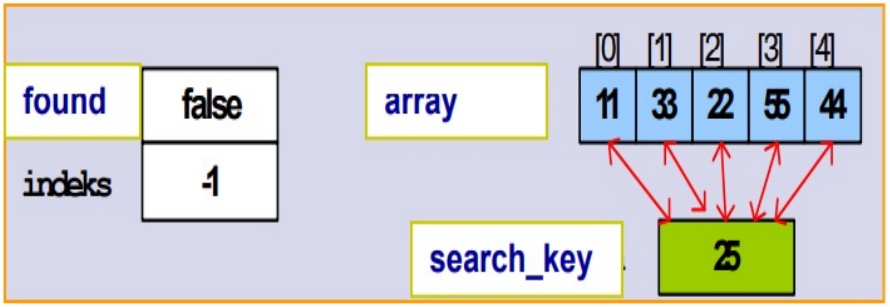
Search for key 22 is successful

p=0,1,2

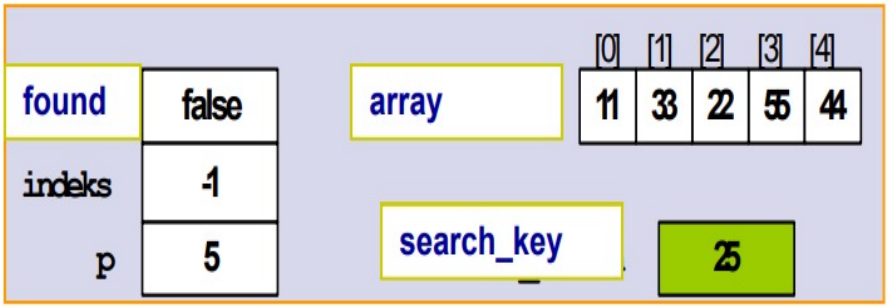
Basic Sequential Search

```
int SequenceSearch(int search_key,
  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; //terminate searching
     }// end if
   } //end for
  return index;
  // return location of value
} //end function
```





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



Searching is unsuccessful

Sequential Search Analysis

- Searching time for sequential search is O(n).
- If the searched key is located at the end of the list or the key is not found, then the loop will be repeated based on the number of element in the list, O(n).
- If the list can be found at index 0, then searching time is, O(1).

Improvement of Basic Search Technique

- The efficiency of basic search technique can be improved by searching on a sorted list.
- For searching on ascending list, the search key will be compared one by one until the searched key is found, or
  - until the searched key value is smaller than the item compared in the list. If this situation happen the searching process stopped.

  => This will minimize the searching process.

Sequential Search on Sorted Data

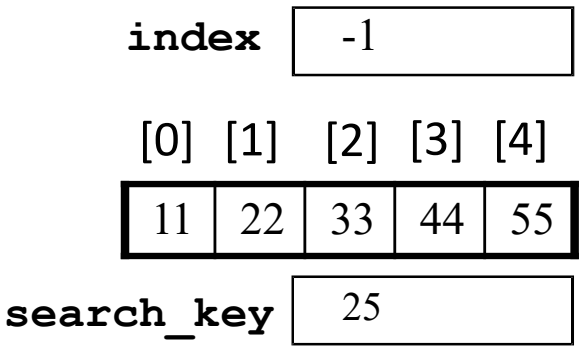
```
int SortedSeqSearch( int search_key,   int array[ ],
                                         int array_size)
{   int p;
    int index = -1;     //-1 means record not found
    for (p = 0; p < array_size; p++)
    {   if (search key < array [p] )
              break;
          // loop terminats when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

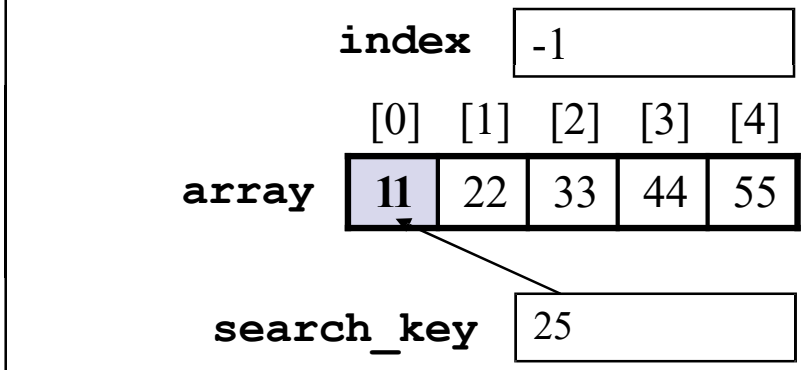
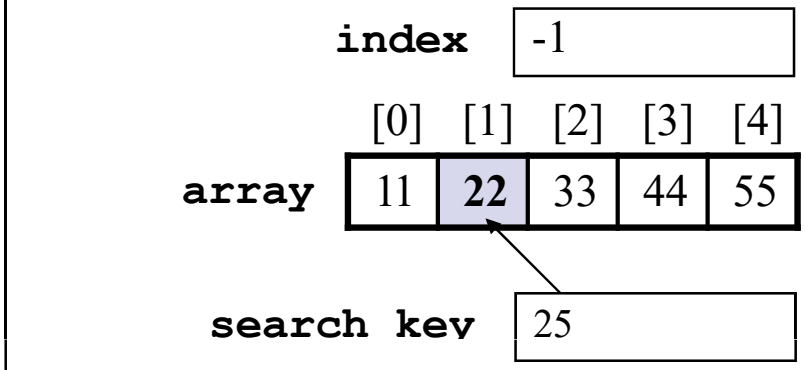
Steps to Execute Sequential Search Function on a Sorted List

Assume:

search_key = 25

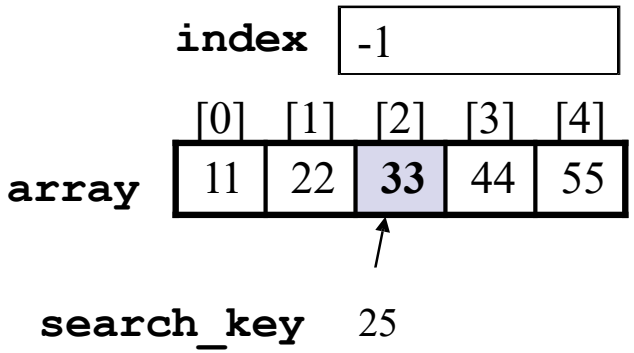
array size = 5

Step 1		Initial value for variable **index** and **array** elements

Step 2		p = 0 **search_key** is compared with the first element in the **array**
Step 3		p = 1 **search_key** is compared with the second element in the **array**

cont...

Step 4



p=2

- **`search_key`** is compared with the third element in the array
- the value of **`search_key`** is smaller than the current element of array
- loop repetition is terminated using "**`break`**" statement