Data Structure
BSSE

By
Islam Zada
Lecturer in Computer Science

Data structure

- A **Data Structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently
- The logical or mathematical model of a particular organization of data is called data structure.
- The term ***Data Structure*** refers to a scheme for organizing related pieces of information

Data structure

- Data structures are classified as either
  - Linear
    - If its elements forms a sequence / linear list
      - Array
      - Linked list
  - Non-Linear
    - Its elements do not forms a sequence
      - Graphs
      - Trees

Data structures

- Some of the most commonly used Data Structures are
  - Array
  - Linked list
  - Tree
  - Stack
  - Queue
  - Graph

Data Structure Operations

- Traversing
- Searching
- Sorting
- Insertion
- Deletion
- Updation
- Merging
- Appending

Algorithm

- A step by step process for the solution of a problem is called algorithm
- A well defined list of steps for the solution of a problem is called algorithm
- Algorithm has two parts
  - Descriptive part
  - List of Steps

Complexity of algorithm

- The function which gives the running time and/or space in terms of the input size.
- The time and space an algorithm uses are the two major measures of the efficiency of an algorithm

Arrays

- A list of a finite number n of homogeneous data elements, such that
  - The elements of the array are referenced respectively by an index set consisting of n consecutive numbers
  - The elements of the array are stored respectively in successive memory locations.
- A collection of adjacent memory locations where homogeneous data can be stored.

Arrays

- The number n of the elements is called the length or size of the array and is obtained by using the formula

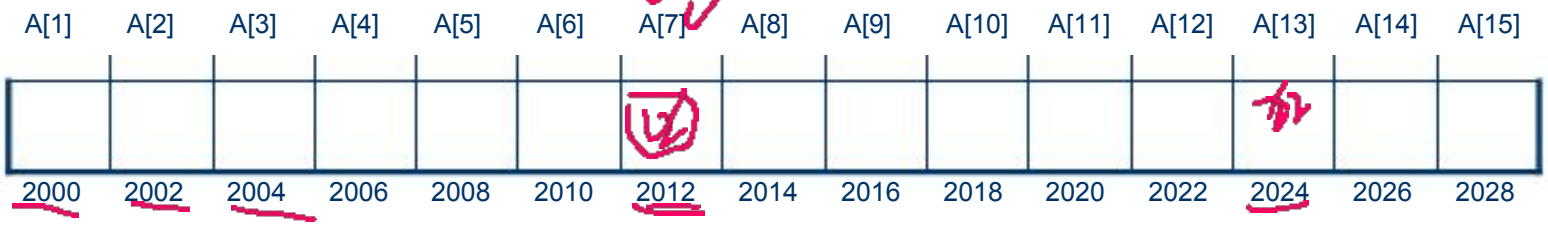
  length = UB – LB + 1

  where

  – UB ---□ Upper Bound
  – LB ----□ Lower Bound

Arrays

- Representation of linear array in the memory



- Loc(A[k]) = Address of the element A[k] of the array A.
- Computer needs only to keep track of the address of the first element i.e.

  Base (A)

Arrays

- Using this address , the computer calculates the address of any element of the array by the following formula

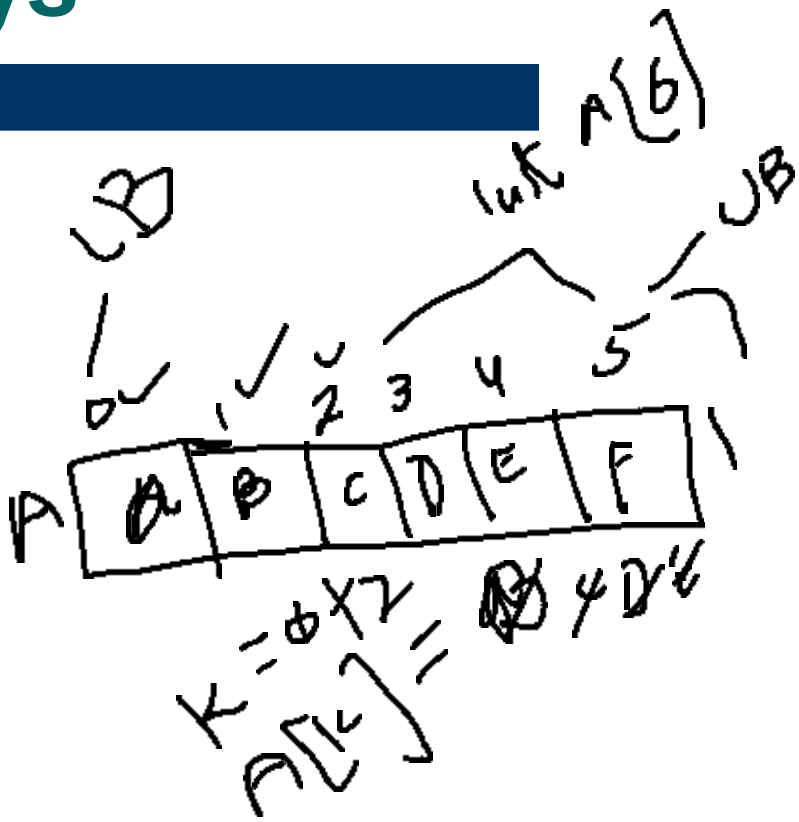
Loc(A[k]) = Base(a) + w (K – LB)

Traversing an Arrays

- **Algorithm**
  - Traversing Linear Array

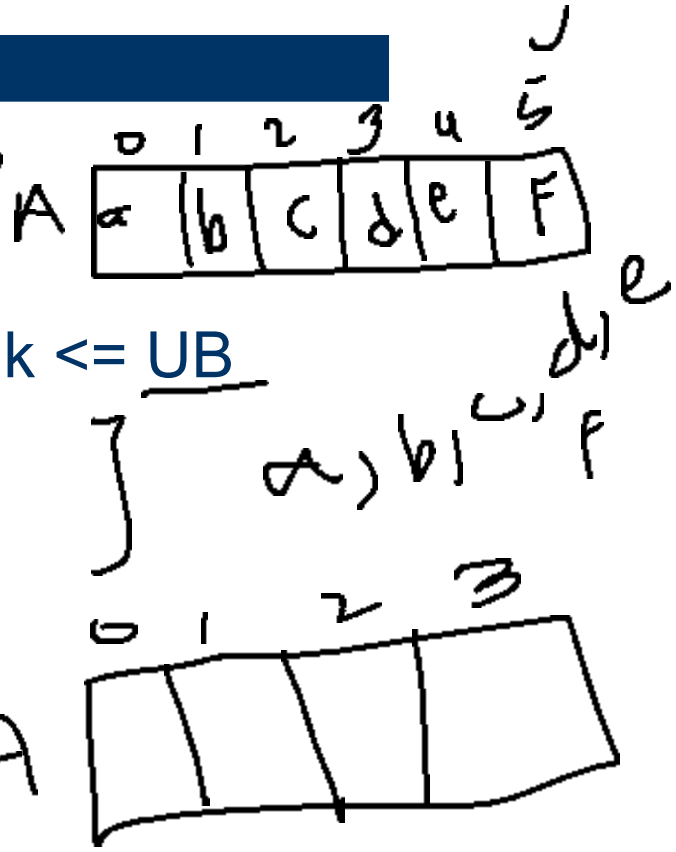
Steps

1. Repeat for K = LB to UB
   apply process to A[k]
2. Exit

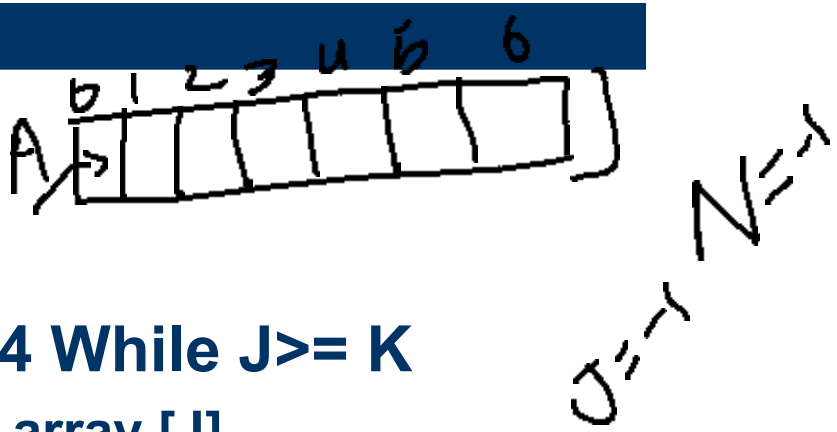


Traversing an Arrays

- **Algorithm**
  - 1. Set k = LB
  - 2. repeat step 3 and 4 while k <= UB
  - 3. Apply process A[k]
  - 4. set k = k + 1
  - 5. Exit



Insertion into an Array



- **Algorithm**
  1. Set j := N
  2. Repeat step 3 and 4 While J>= K
     3. set array[J+1] := array [J]
     4. set J:= J -1
  5. Set array [K] := ITEM
  6. Set N:= N+1
  7. Exit

Deletion from the Array

- **Algorithm**
  1. Set ITEM = array [k]
  2. Repeat for J = K to N - 1
     3. set array [J] := array[J+1]
  4. Set N:= N - 1
  5. Exit

Sorting: Bubble Sort Algorithm

1. Repeat step 2 and 3 for k = 1 to N – 1
2. Set ptr = 1
3. Repeat while ptr <= N – k
   a) If data [ptr] > data [ptr + 1]
      Swap data [ptr] and data [ptr + 1]
   b) Ptr = ptr + 1

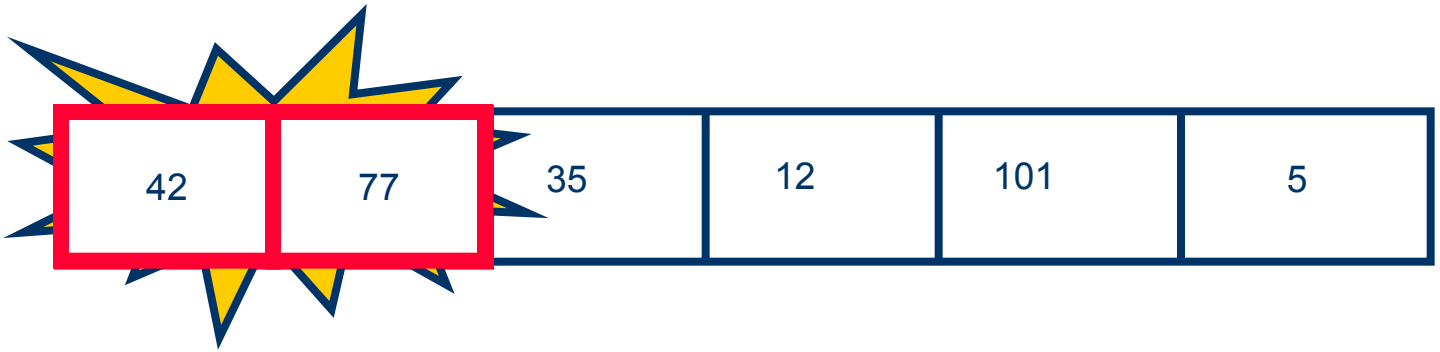
   [End of inner loop]

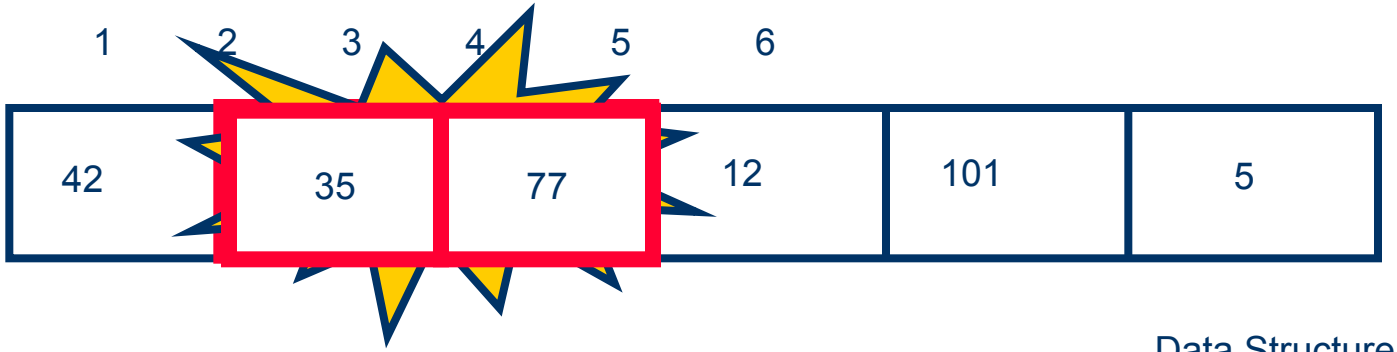
   [End of outer loop]
4. Exit

Bubble Sort

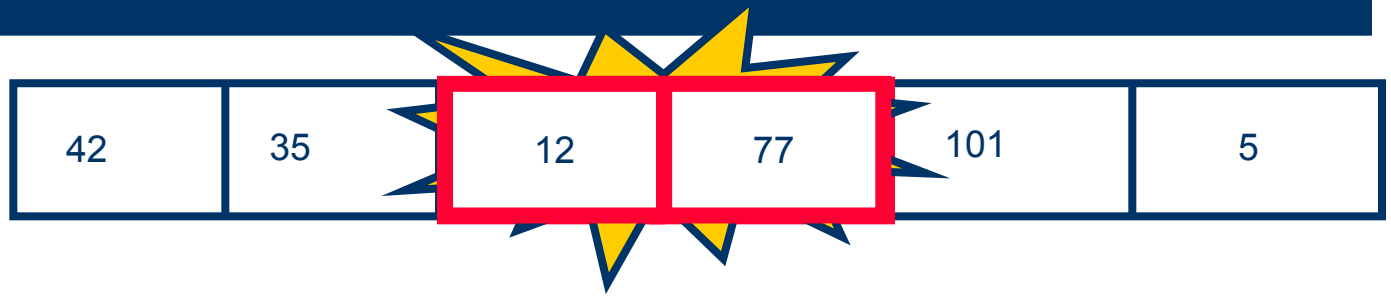
- Bubbling Up" the Largest Element
- Traverse a collection of elements
- Move from the front to the end
- “Bubble” the largest value to the end using pair-wise comparisons and swapping

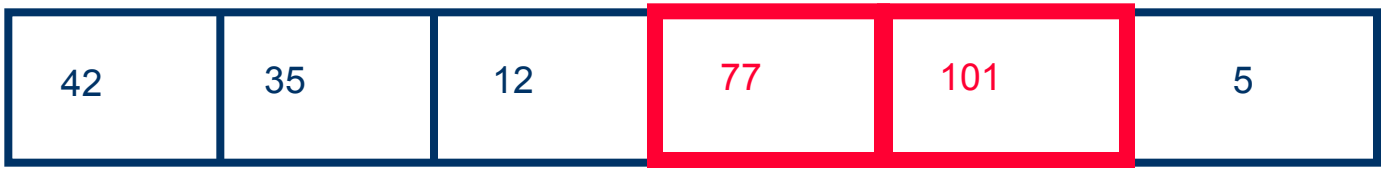
"Bubbling Up" the Largest Element



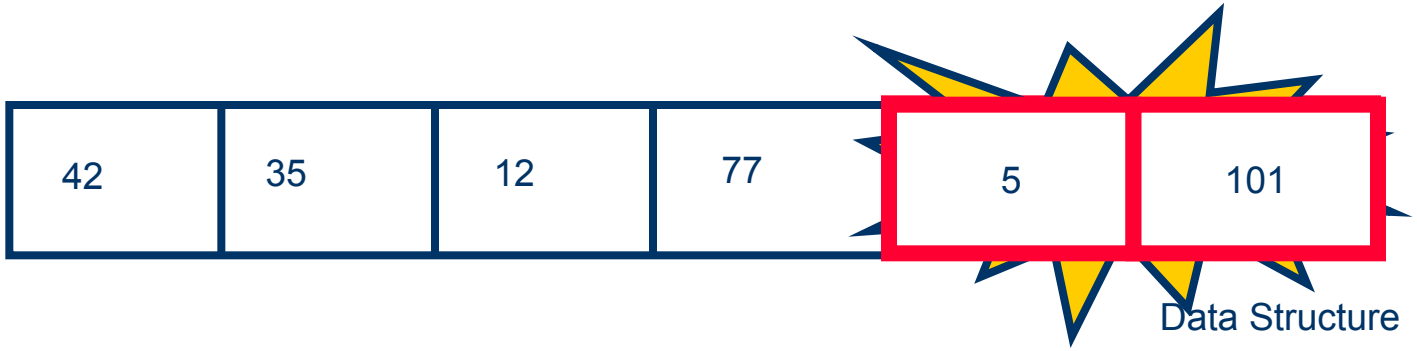


"Bubbling Up" the Largest Element





No need to swap



Important

- Notice that only the largest value is correctly placed
- All other values are still out of order
- So we need to repeat this process

Largest value correctly placed

Repeat “Bubble Up” How Many Times?

- If we have N elements…
- And if each time we bubble an element, we place it in its correct location…
- Then we repeat the “bubble up” process N–1 times.
- This guarantees we’ll correctly place all N elements.

Linear Search Algorithm 1

1. Set k := 1 and loc := 0
2. Repeat step 3 and 4 while loc = 0 and K <= N
3. If item = data [k] then set loc = k
4. Set k := k+1
5. If loc = 0 then
   Write Item does not exist
   Else
   Write loc is the location of the element
6. exit

Linear Search Algorithm 2

1. [insert item at the end of the array] set data [N+1] := item
2. loc := 1
3. Repeat while (data [loc] != item)
   Set loc := loc+1
4. If loc = N + 1 then
   loc := 0
   Write Item does not exist
   Else
   Write loc is the location of the element
5. exit

Binary Search

1. Set beg = LB, end = UB , mid = INT ((beg + end)/2)
2. Repeat 3 and 4 while beg <= end and data [mid] != item
3. If item < data [mid] then
   end = mid – 1
   else
   beg = mid + 1
4. mid = INT ((beg + end)/2)
5. If data [mid] = item
   set loc = mid
   else
   loc = null
6. exit

Two Dimensional array

- Each element in the array is referenced by a single subscript
- A two dimensional (**m** x **n**) array A is a collection of m . n data elements such that each element is specified by a pair of integers (such as j , k) called subscript with the property that
  1. **1 <= j <= m and**
  2. **1 <= k <= n**

Two Dimensional array

- A two dimensional arrays are called
  - MATRICES in mathematics and
  - TABLES in business applications
- Two dimensional arrays are sometime called matrix arrays
- In a two dimensional array A (j , k) the
  - First subscript j stands for ROW and
  - The second subscript k stands for COLUMN

Two Dimensional array

- **Representation of two dimensional array in memory**
- A two dimensional array is either represented in the computer memory in the form of
  1. **Column major order**

     Or
  2. **Row major order**

Representation of two dimensional array in memory

Row Major Order	
	A(1,1)
	A(1,2)
	A(1,3)
	A(2,1)
	A(2,2)
	A(3,2)
	A(3,1)
	A(3,2)
	A(3,3)

Column Major Order	
	A(1,1)
	A(2,1)
	A(3,1)
	A(1,2)
	A(2,2)
	A(3,2)
	A(1,3)
	A(2,3)
	A(3,3)

Representation of two dimensional array in memory

- Computer needs only to keep track of the address of the first element i.e.

  Base (A)

- Using this address , the computer calculates the address of any element of the array by the following formula
  - **For column major order**
    - **Loc (A [j , k]) = Base (a) + w [m(K – LB)+ (j – LB)]**
  - **For row major order**
    - **Loc (A [j , k]) = Base (a) + w [n(j – LB)+ (k – LB)]**

Quiz