Searching In Array

- Searching is a process of finding the required data in the array.
- Searching becomes more important when the length of the array is very large.
- There are two techniques to searching elements in array as follows:

➢Linear or Sequential search

➢Binary search

Sequential Search

- Sequential search is also known as linear or serial search. It follows the following step to search a value in array.

➢ Visit the first element of array and compare its value with required value.

➢ If the value of array matches with the desired value, the search is complete.

➢ If the value of array does not match, move to next element an repeat same process.

Binary Search

- Binary search is a quicker method of searching for value in the array. Binary search is very quick but it can only search an sorted array.
- It cannot be applied on an unsorted array.

➢ It locates the middle element of array and compare with desired number.

➢ If they are equal, search is successful and the index of middle element is returned.

➢ If they are not equal, it reduces the search to half of the array.

➢ If the search number is less than the middle element, it searches the first half of array.

- Otherwise it searches the second half of the array. The process continues until the required number is found or loop completes without successful search.

Algorithm: (Binary Search)

BINARY (DATA, LB, UB, ITEM, LOC)
//Here DATA is a sorted array with LB and UB. ITEM is a given information. BEG, MID and END represent beginning, middle and end location of the DATA. This algorithm finds the location LOC of ITEM in DATA or sets LOC=NULL.

1. Set BEG=LB, END=UB and MID =INT((BEG+END)/2).
2. Repeat steps 3 and 4 while BEG≤ END and DATA[MID] ≠ ITEM.
3. If ITEM<DATA[MID], then set END=MID-1.
   Else set BEG=MID+1
4. Set MID =INT((BEG+END)/2)
5. If DATA[MID]=ITEM, then set LOC=MID
6. Else LOC=NULL.
7. Exit.

Sorting Arrays

- Sorting is a process of arranging the value of array in a particular order.
- An array can be sorted in two order.

➢Ascending Order

➢Descending Order

12	25	33	37	48
48	37	33	25	12

Techniques of Sorting Array

- There are two techniques of sorting array:

1. Selection Sort
2. Bubble Sort

Selection Sort

- Selection sort is a technique that sort an array.
- It selects an element in the array and moves it to its proper position.
- Selection sort works as follows:

1. Find the minimum value in the list.
2. Swap it with value in the first position.
3. Sort the remainder of the list excluding the first value.

Bubble Sort

- Bubble Sort is also known as exchange sort.
- It repeatedly visits the array and compares two items at a time. It works as follows:

1. Compare adjacent element. If the first is greater than the second, swap them.
2. Repeat this for each pair of adjacent element, starting with the first two and ending with the last two. (at this point last element should be greatest).
3. Repeat the step for all elements except the last one.
4. Keep repeating for one fewer element each time until there are no pairs to compare.

Bubble sort algorithm

INPUT: Before sorting: 12, 295, 2, 13, 14

OUTPUT: After ascending sorting: 2, 12, 13, 14, 295

//BUBBLE (DATA, N)

// Here data is an array with N elements. This algorithm sorts the elements in DATA

1. Repeat steps 2 and 3 for K=1 to N-1.
2. Set I=1.
3. Repeat while I≤N-K

a. If DATA[I] > DATA[I+1], then interchange them.

b. Set I=I+1.

4. Exit.

Two-D Arrays

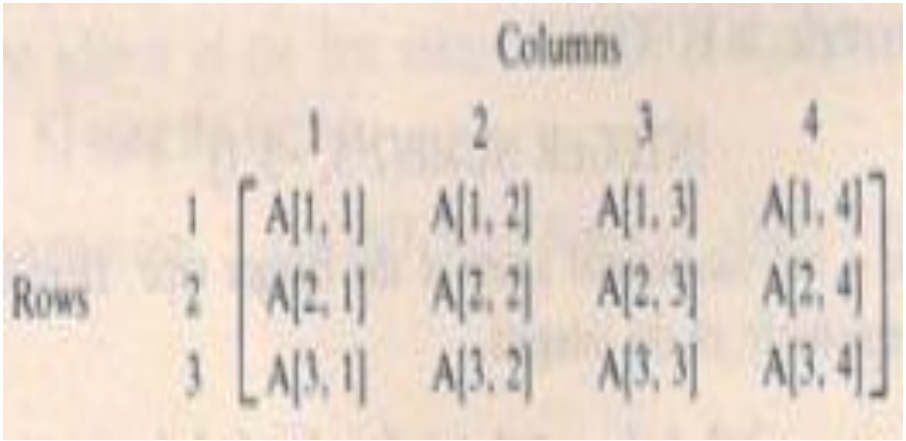
- Two-D array can be considered as table that consists of rows and columns.
- Each element in 2-D array is referred with the help of two indexes. One index indicates row and second indicates the column.
- Declaring 2-D Array: Data_type Identifier[row][column];

e.g: int arr[3][4];

Initialization:

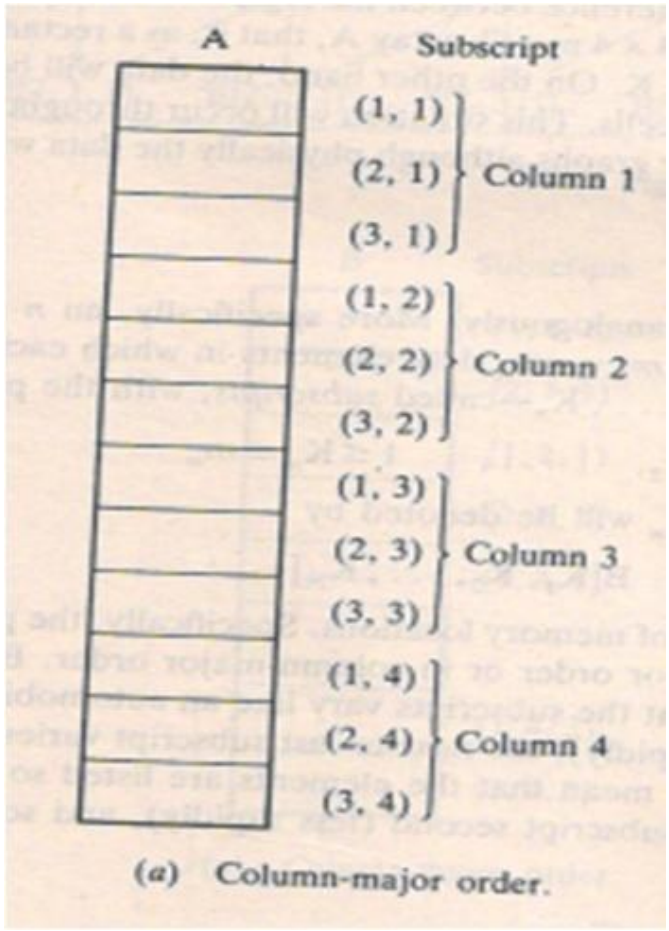
Int arr[3][4]={{12, 5, 22,7}, {4, 23, 78,8}, {15, 5, 6,9}}

12	5	22	7
4	23	78	8
15	5	6	9



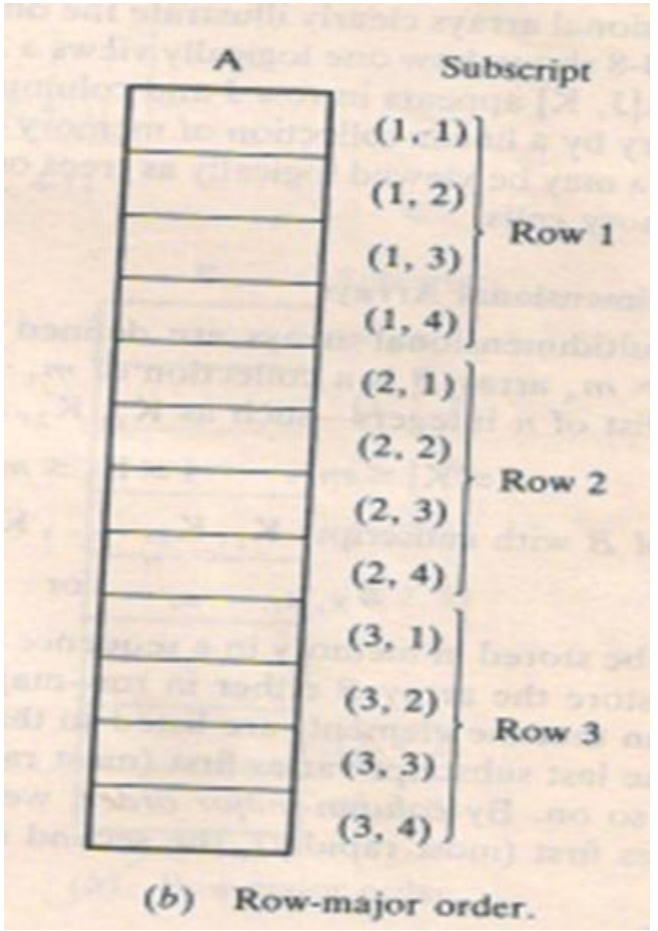
Representation of two-dimensional array in memory

Type 1



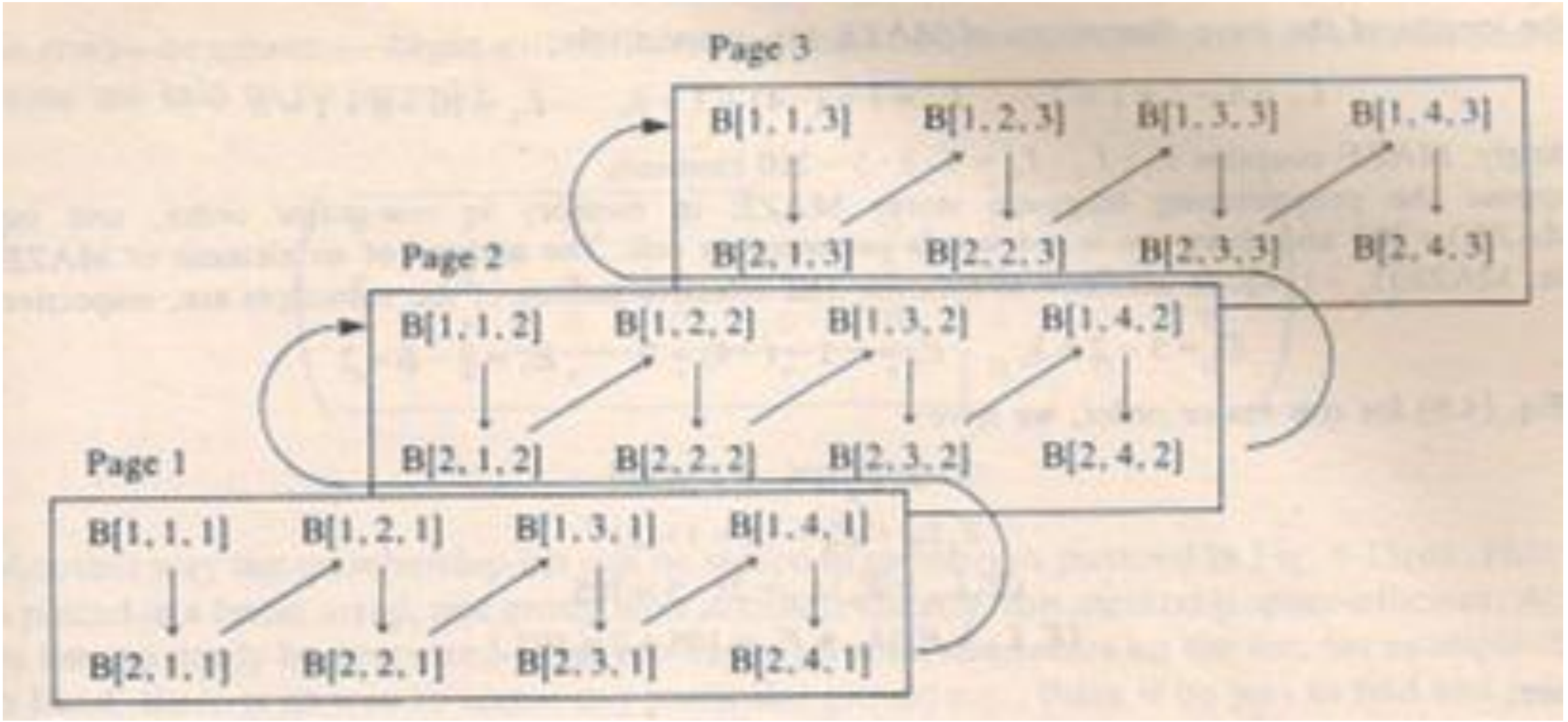
(a) Column-major order.

Type 2

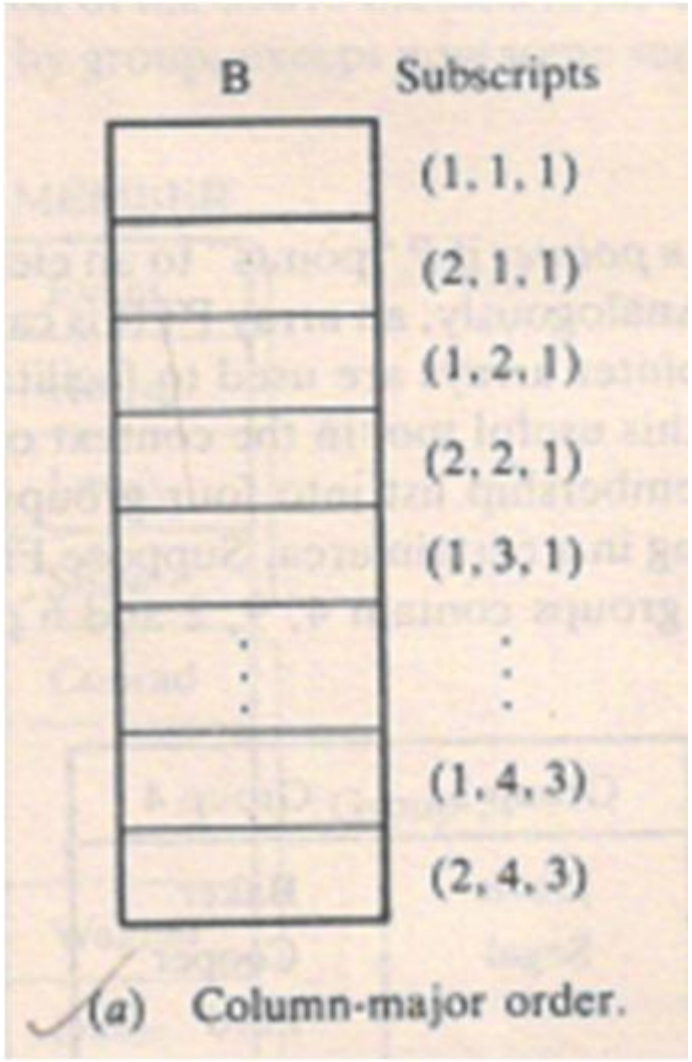


(b) Row-major order.

Representation of three-dimensional array in memory

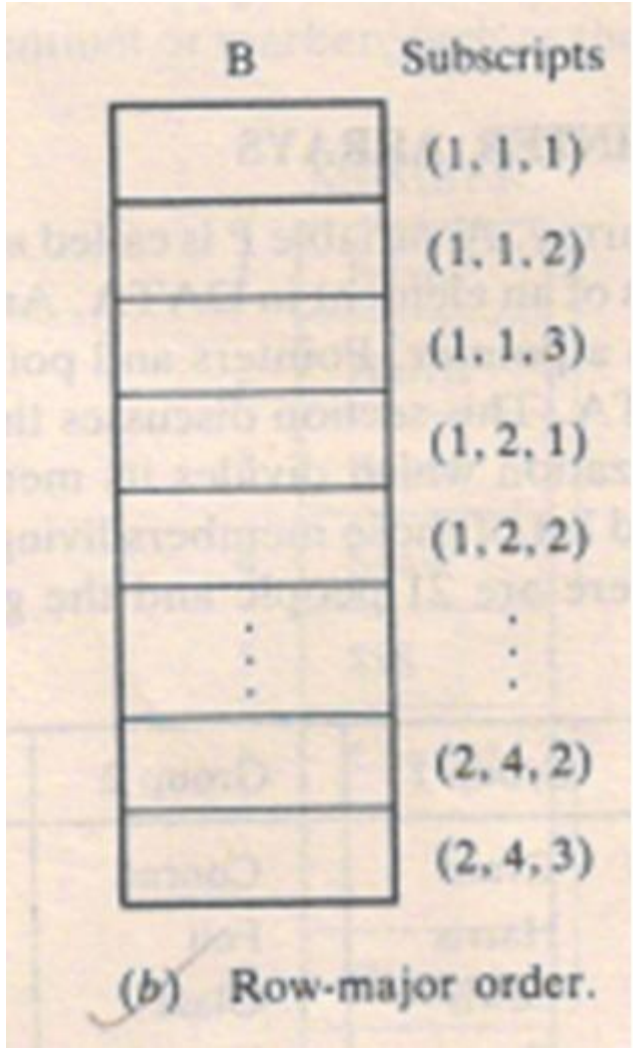


Type 1



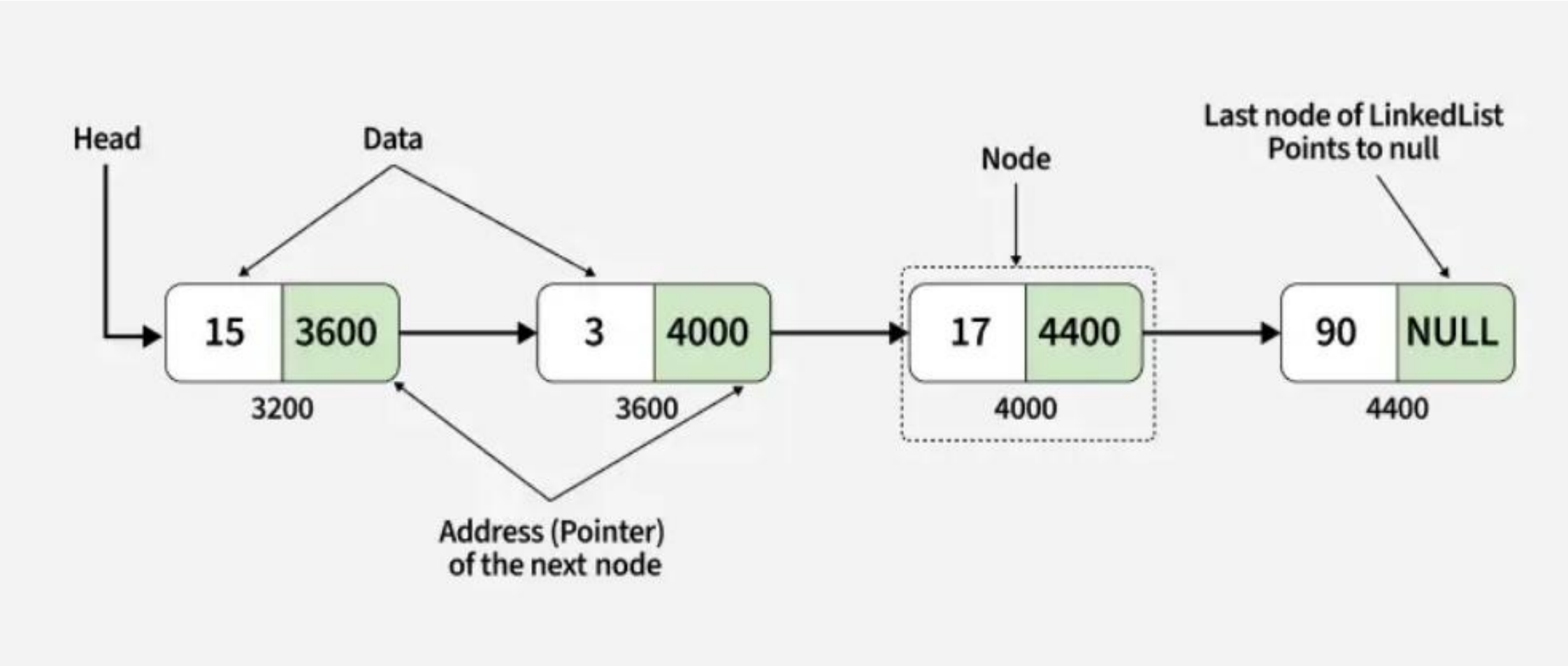
(a) Column-major order.

Type 2



(b) Row-major order.

Linked list


Head
Data
Node
Last node of LinkedList
Points to null
15
3600
3
4000
17
4400
90
NULL
3200
3600
4000
4400
Address (Pointer)
of the next node

- Every **node** stores:

➢**Data** (the number).

➢A **link** (the address of the next node).

Definition: The linear Structure that has the relationship between elements, represented by means of pointers or links.

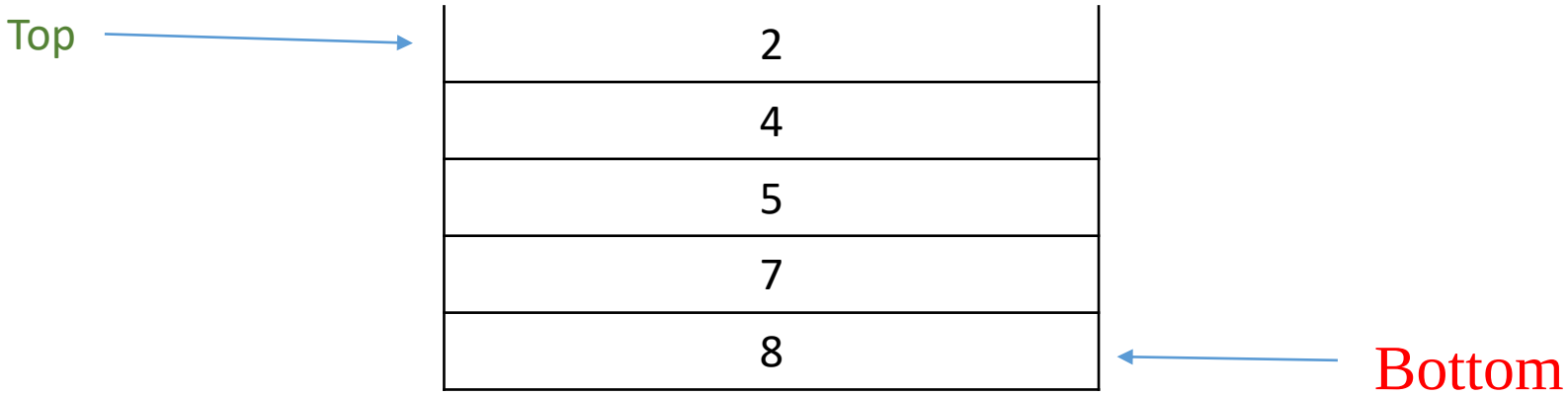
Difference between array and linked list

Array	Linked list
Fixed size (you have to declare the size upfront).	You can **grow or shrink** easily.
Inserting or deleting needs shifting elements.	You **just change the links** to add or remove elements.
	No need to move everything around.

Stack

- New nodes can be added and removed only at the top
- Similar to a pile of dishes
- Last-in, first-out (LIFO)
- push: Adds a new node to the top of the stack
- pop: Removes a node from the top

- A stack is a list in which insertion and deletion take place at the same end
- This end is called Top
- The other end is called Bottom.



Queue

- Similar to a supermarket checkout line
- First-in, first-out (FIFO)
- Nodes are removed only from the head
- Nodes are inserted only at the tail
- Insert and remove operations
- Enqueue (insert) and dequeue (remove)
- A queue is like a line of people waiting for a bank teller.
- The queue has a front and a rear.
- Rear (insertion) and Front(Removal)