AEL-1

Md. Atiqur Rahman

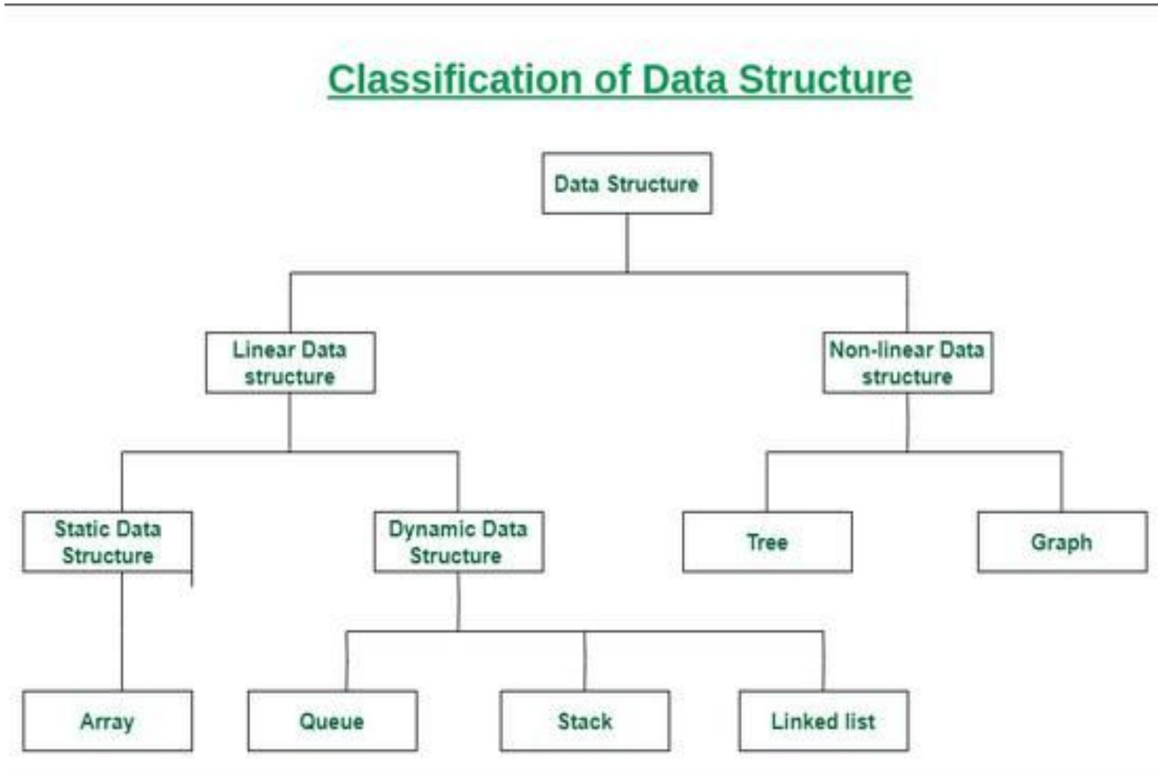
Data Structures - 1

Data Structures

A data structure is a way of organizing and storing data in a computer so that it can be accessed and used efficiently.

Types of data structures,

- Linear
    - Static
        - Array
    - Dynamic
        - Queue
        - Stack
        - Linked List
- Non-linear
    - Tree
    - Graph



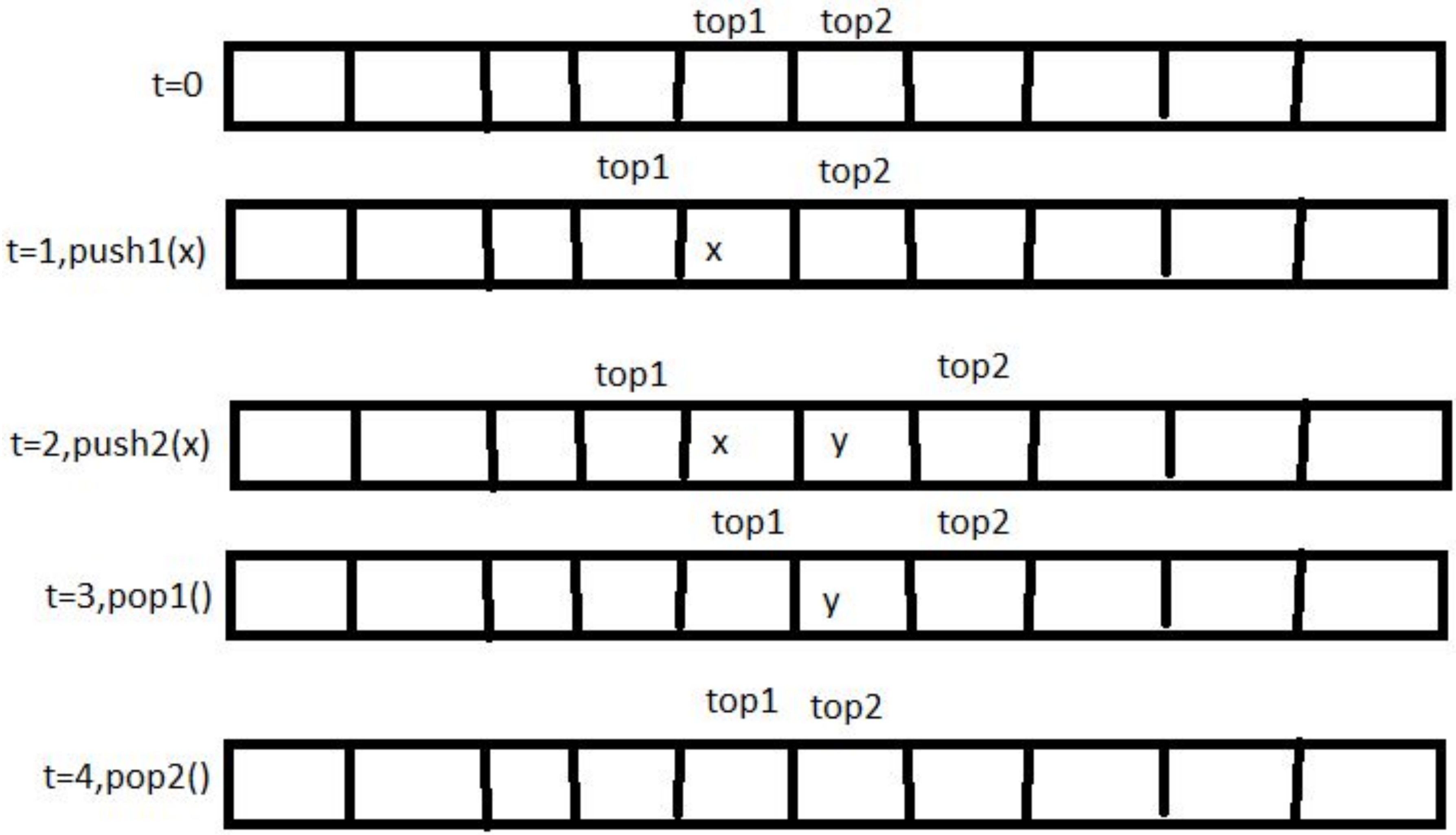
Definitions

- Linear : data elements are arranged sequentially.
- Non-linear : We can not traverse elements in a single run only.
- Static : has fixed memory size and easier to access the elements in a static data.
- Dynamic : Memory size is not fixed. It can be randomly updated during the runtime which may be considered efficient concerning the memory (space) complexity of the code.
- **Array** : An array is a collection of items of same data type stored at contiguous memory locations.
- **Queue** : A Queue is defined as a linear data structure which works almost like list in which all additions to the list are made at one end, and all deletions from the list are made at the other end, like FIFO (First In First Out) order.
- **Stack** : Stack is a linear data structure that follows a particular order LIFO which implies that the element that is inserted last, comes out first.
- **Linked List** : Linked List is a linear data structure, in which elements are not stored at a contiguous location, rather they are linked using pointers. Linked List forms a series of connected nodes, where each node stores the data and the address of the next node.

Implement two Stacks in an Array

- To implement **push1():**
  - First, check whether the **top1** is greater than 0
    - If it is then add an element at the top1 index and decrement top1 by 1
    - Else return Stack Overflow.
- To implement **push2():**
  - First, check whether **top2** is less than n – 1
    - If it is then add an element at the top2 index and increment the top2 by 1
    - Else return Stack Overflow.
- To implement **pop1():**
  - First, check whether the **top1** is less than or equal to n / 2
    - If it is then increment the top1 by 1 and return that element.
    - Else return Stack Underflow.
- To implement **pop2():**
  - First, check whether the **top2** is greater than or equal to (n + 1) / 2
    - If it is then decrement the top2 by 1 and return that element.
    - Else return Stack Underflow.

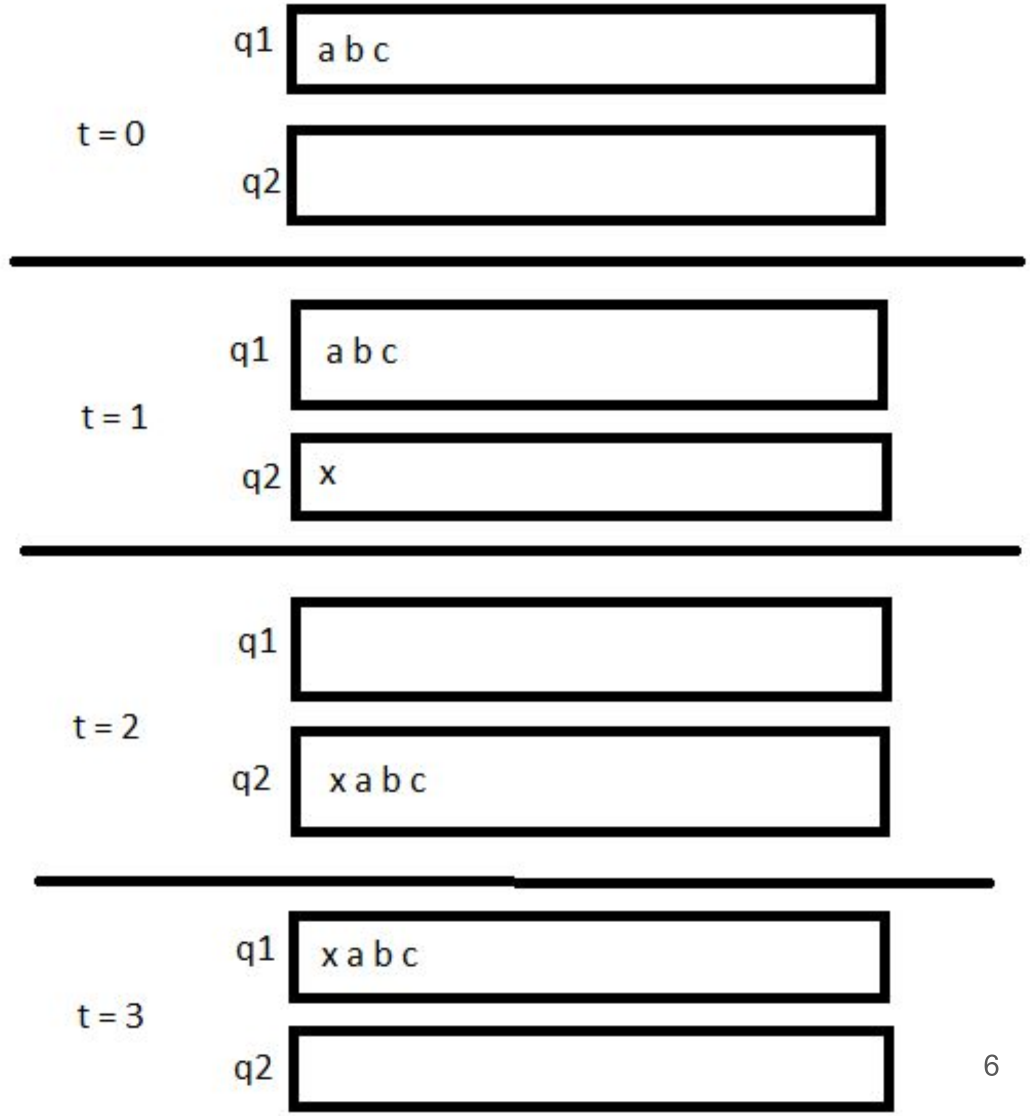
Continued



top1
top2
t=0
top1
top2
t=1,push1(x)
x
top1
top2
t=2,push2(x)
x
y
top1
top2
t=3,pop1()
y
top1
top2
t=4,pop2()

Implement Stack using queues

- Follow the below steps to implement the **push(s, x)** operation:
  - Enqueue x to q2.
  - One by one dequeue everything from q1 and enqueue to q2.
  - Swap the queues of q1 and q2.
- Follow the below steps to implement the **pop(s)** operation:
  - Dequeue an item from q1 and return it.



Reverse a Link List using Stack

- Store the nodes(values and address) in the stack until all the values are entered.
- Once all entries are done, Update the Head pointer to the last location(i.e the last value).
- Start popping the nodes(value and address) and store them in the same order until the stack is empty.
- Update the next pointer of last Node in the stack by NULL.