15CSE374
INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS

Sarath tv

Last Lecture

Asymptotic Analysis.

Space Bounds.

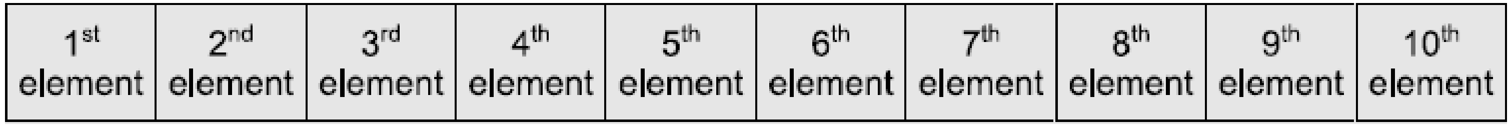
LIST

- Represents a countable number of ordered values
- An abstract data type.
- Many programming languages provide support for list data types
- Can be used to store a list of elements
- Basis for other abstract data types including the queue, the stack.

Arrays

- An array is a collection of similar data elements.
- The elements of the array are stored in consecutive memory locations and are referenced by an index.
- Static & dynamic Array.

```
marks[10]
```



1st element	2nd element	3rd element	4th element	5th element	6th element	7th element	8th element	9th element	10th element

Operations

- Accessing the elements- index
- Initializing elements
- Traversing an array.
- Searching an element in an array

Stack

Last In First Out (LIFO), the last data stored in the stack is the first than can be retrieved and the first data stored in the stack is the last to be retrieved.

Analogy - **pile** of **plates.** The **bottom** plate is the **first data** pushed onto the stack and the **top plate** is the **last data** pushed.

When the **top plate** is removed (**pulled**), the **one below pops** up to become the **new top.**

only the **top element** in the **stack can be accessed.**

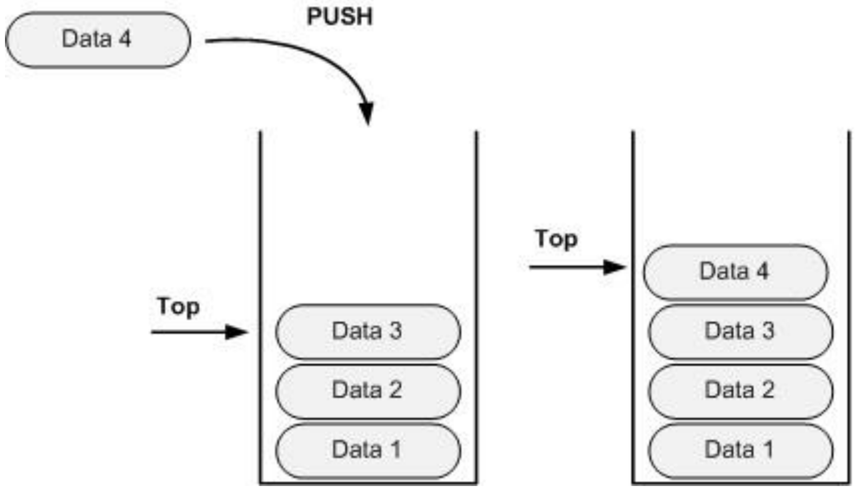
Stack

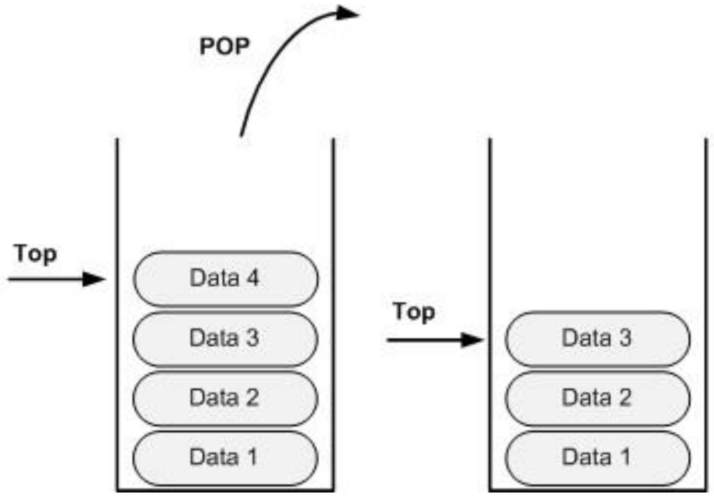
A ***stack*** is a container of objects that are inserted and removed according to the ***last-in first-out*** (***LIFO***) principle

Fundamental operations involve the “pushing” and “popping”

Example 1: Internet Web “back” button

Example 2: Text editors “undo” operation


Data 4
PUSH
Top
Data 3
Data 2
Data 1
Top
Data 4
Data 3
Data 2
Data 1


POP
Top
Data 4
Data 3
Data 2
Data 1
Top
Data 3
Data 2
Data 1

Interfaces to Stack ADT

- PUSH
- POP
- Size
- top
- isEmpty

Interfaces

- push(e): Insert element e at the top of the stack.
- pop(): Remove the top element from the stack; an error occurs if the stack is empty.
- top(): Return a reference to the top element on the stack, without removing it; an error occurs if the stack is empty.

Additionally, let us also define the following supporting functions:

- size(): Return the number of elements in the stack.
- isEmpty(): Return true if the stack is empty and false otherwise.

If TOP = NULL, then it indicates that the stack is empty and

if TOP = MAX–1, then the stack is full

Stack

Operation	Output	Stack Contents
Push(5)	-	{5}
Push(3)	-	{5,3}
Pop()	-	{5}
pop()	-	{}
Pop()	“error”	{}
Push(9)	-	{9}
Push(5)	-	{9,5}
Push(8)	-	{9,5,8}
Top()	8	{9,5,8}
Size()	3	{9,5,8}

Applications of stack

- The simplest application of a stack is **to reverse a word**. You push a given word to stack - letter by letter - and then pop letters from the stack.
- Another application is an **"undo" mechanism** in text editors; this operation is accomplished by keeping all text changes in a stack.
- **Backtracking**. This is a process when you need to access the most recent data element in a series of elements. Think of a labyrinth or **maze** - how do you find a way from an entrance to an exit?
- Once you reach a **dead end**, you must backtrack. But backtrack to where? to the previous choice point. Therefore, at each choice point you store on a stack all possible choices. Then backtracking simply means popping a next choice from the stack.

THANK YOU!!!!!