Data Structures and Algorithm Analysis

6

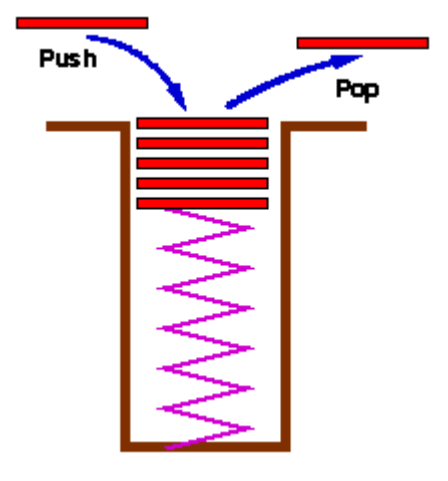
Dr. Syed Asim Jalal
Department of Computer Science
University of Peshawar

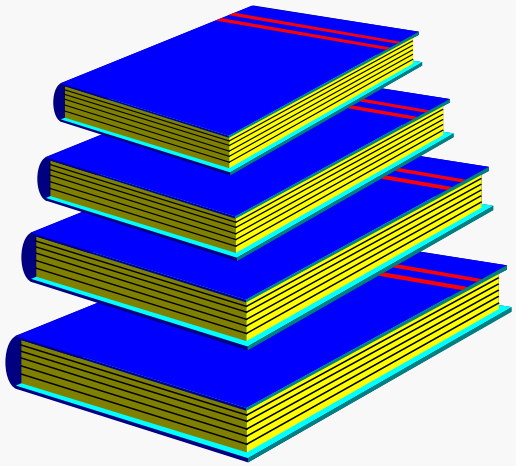
Stack Data Structure

Stack Data Structure

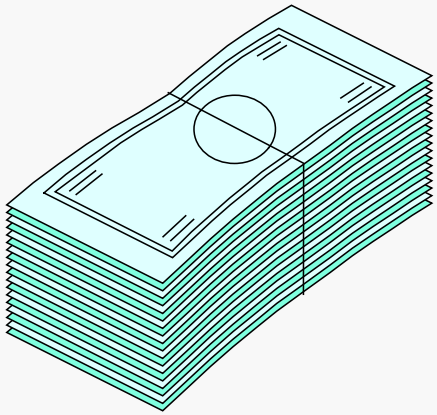
- Linear Data Structures (linked lists or Arrays) allow us to insert and delete elements at any place in the list, i.e. beginning, middle or end.
- In some applications the requirement is to insert and remove data at one end of the list.
- Stack is a Linear Data Structure where both addition and removal of elements takes place at the same end, called the Top of the stack.

- Stack in real life:
  - stack of books
  - stack of plates
- Add new items at the top
- Remove an item from the top
- Can only access element at the top

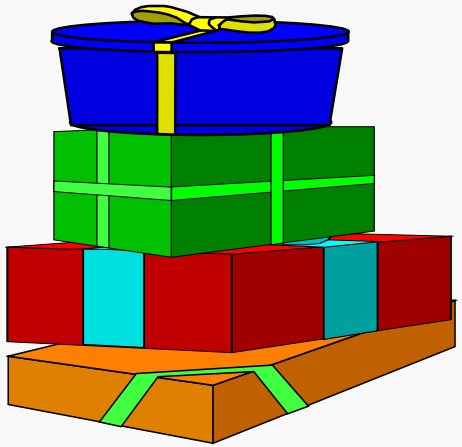




Stack of Books

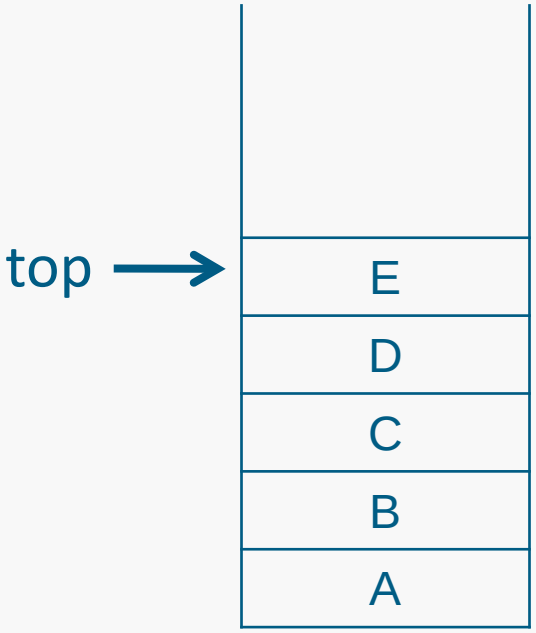


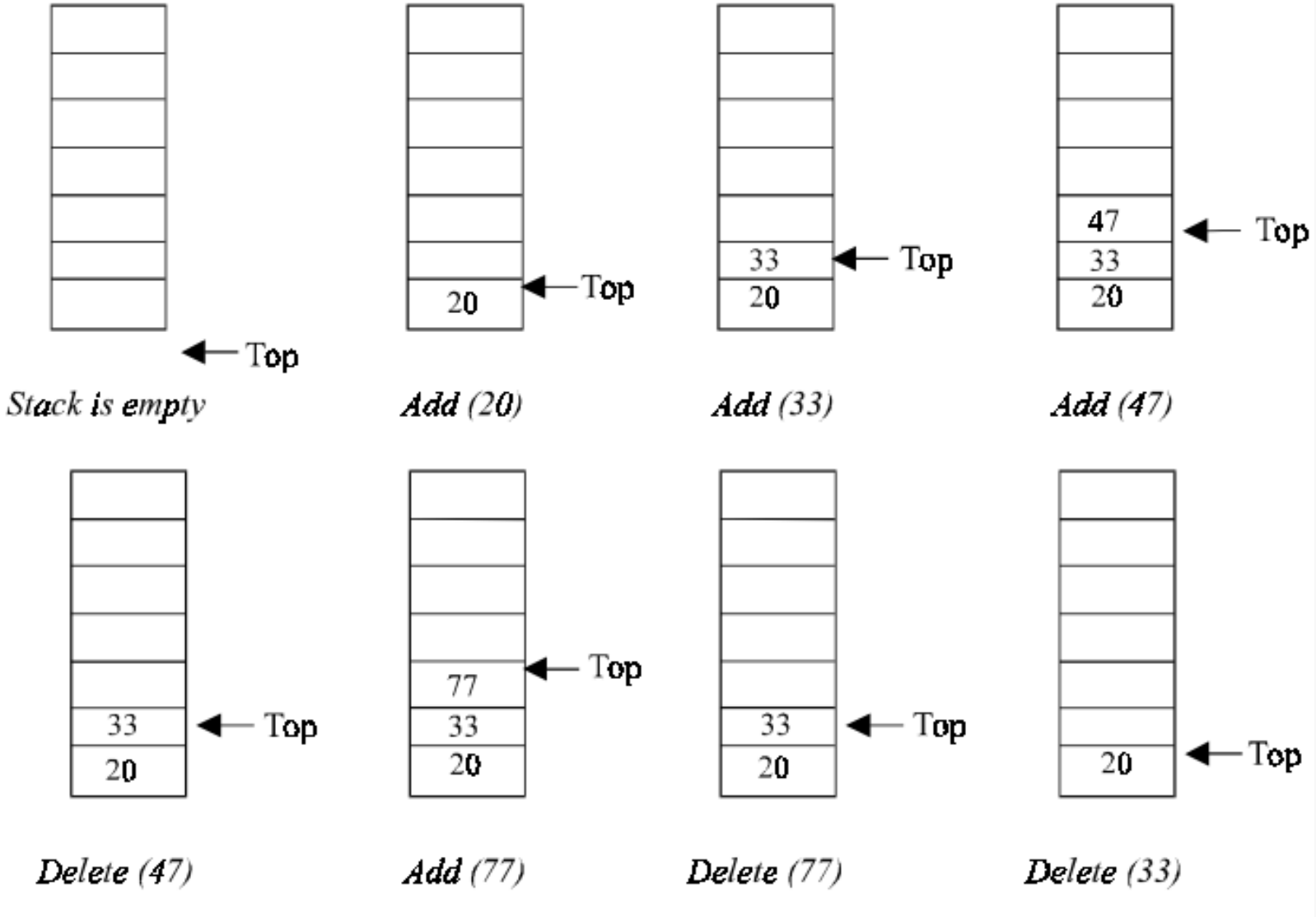
Stack of Notes



Stack of Gifts

- In Stack data structure the last element inserted is the first element to be removed.
- This behavior is known as Last-in-First-out (LIFO)
- A stack is an ordered collection of items in which insertions and deletions are done only at one end, called the **Top** of the stack.




Top
Stack is empty
20
Top
Add (20)
33
20
Top
Add (33)
47
33
20
Top
Add (47)
33
20
Top
Delete (47)
77
33
20
Top
Add (77)
33
20
Top
Delete (77)
20
Top
Delete (33)

- In stack the last-item added is removed first because it needs to be processed first. So the concept of removal of item from stack is not same as deletion, in stack we remove item because we have to process that item.
- So removal is not a deletion in real sense.

Some Applications of Stack

- Mathematical Expression Evaluation
  - Infix, post-fix, expression evaluation
- Memory management
  - Stack is used in nested constructs implementation
    - Brackets or parenthesis matching
  - Stack is used in Recursive function calls
- Backtracking in tree or graphs
  - finding paths
  - exhaustive searching
- Undo operations
- etc.

Operations

- The following are the main operations of Stack
- Push
  - It is insertion of an item at Top of Stack
  - Every Push operation moves the Top of the stack to the inserted item.
- Pop
  - Removal of an item from top of stack if the stack is not empty.
  - We perform POP because we need to process the element on the top of the stack.

Other Auxiliary Operations on Stack

- IsEmpty():
  - Checks if the stack is empty
    - returns TRUE if the stack is empty
    - returns FALSE if there is at least one item in the stack
- IsFull():
  - Checks if the stack is Full
  - This operation is used ONLY when there is a limit on the size of the stack
    - returns TRUE if the stack is FULL
    - returns FALSE otherwise

Stack Implementations

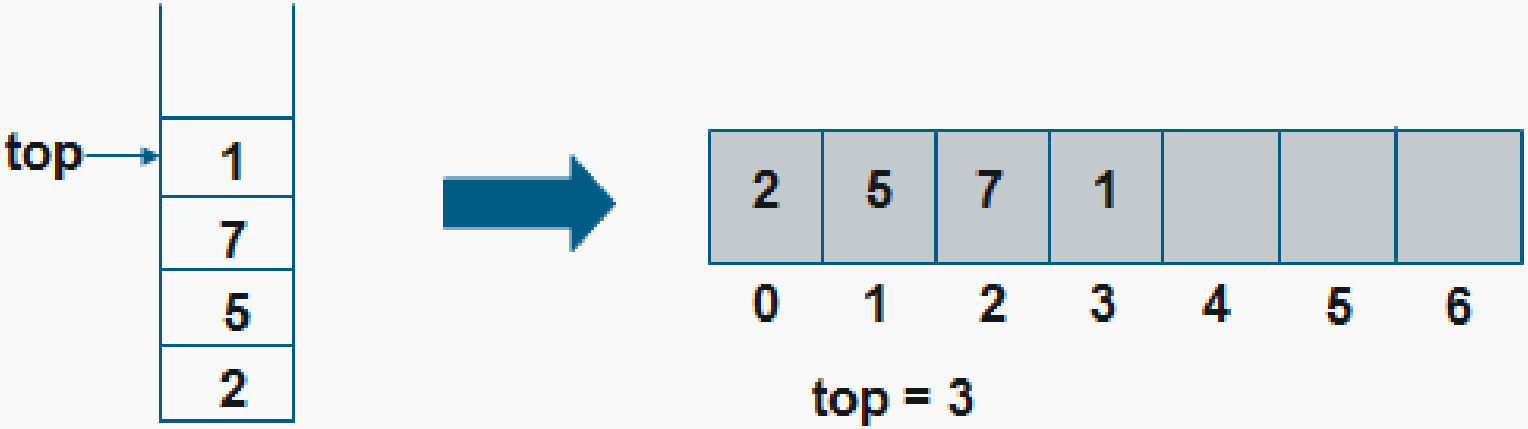
- Static Implementation of Stack
  - Implements stack using an Array
  - Implementation using array is simple but is not flexible as the size of the stack has to be decided before implementation.

- Dynamic Implementation
  - Dynamic implementation is done using Linked Lists
  - The implementation is complex using structures and pointers but offers flexibility in terms of size of the stack.

Stack implementation using Array

- Stack implementation using arrays uses
  - Max_Size variable to store the maximum number of elements to be stored in the stack and
  - Top variable that points to the index of the top element.

- The Top is initialised to -1 in the start when the Stack is empty.
  - Initialisation of Top to -1 or 0 depends on the implementation programming language.



Max_Size = 7

Push Operation - Algorithm

- Suppose ***STACK[Max_Size]*** is a one dimensional array for implementing the stack, which will hold the data items. ***TOP*** is the pointer that points to the top most element of the stack. Let ***Value*** is the data item to be pushed.

1. **If TOP = Max_Size – 1 // Check Over Flow condition**
   a) **Display "The Stack is FULL"**
   b) **Exit**
2. **TOP = TOP + 1**
3. **STACK[TOP] = Value**
4. **Exit**

Pop Operation - Algorithm

- Suppose ***STACK[Max_Size]*** is a one dimensional array for implementing the stack, which will hold the data items. ***TOP*** is the pointer that points to the top most element of the stack. ***Value*** is the popped (or deleted) data item from the top of the stack.

1. **If TOP < 0** **// Check underflow condition**
   a) **Display "The Stack is empty"**
   b) **Exit**
2. **Value = STACK[TOP]**
3. **TOP = TOP - 1**
4. **Exit**

One example implementation of stack in C

```
#include<stdio.h>
#include<conio.h>

#define MAXSIZE 100

struct stack
{
        int stack[MAXSIZE];
        int Top;
};

void main( )
{
        struct stack *sp;
        sp->Top=–1;
              …
      push(sp);
      push(sp);
      pop(sp);
```

```
void push(struct stack *ptr)
{
        int item;
        if (ptr->Top == MAXSIZE–1)
        {
              printf(“\nThe Stack Is Full”);
              getch();
        }
else
      {
              printf(“\nEnter The Element = ”);
              scanf(“%d”,&item);
              ptr->stack[++ptr->Top]=item;
      }
 }
```

Implementation detail

```
void pop(struct stack *ptr)
{
    int item;
    if (ptr ->Top == -1)
        printf("\nThe Stack Is Empty");
    else
    {
        item= ptr ->stack[ptr ->Top--];
        printf("\nThe Poped Element Is = %d",item);
    }
}
```

- Next
  - Stack implementation using Linked List
  - Conversion of Infix, Prefix and Post-fix expressions using Stack