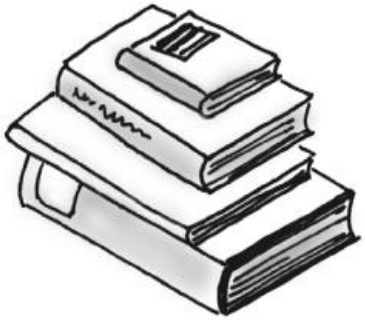
Data Structures and Algorithms

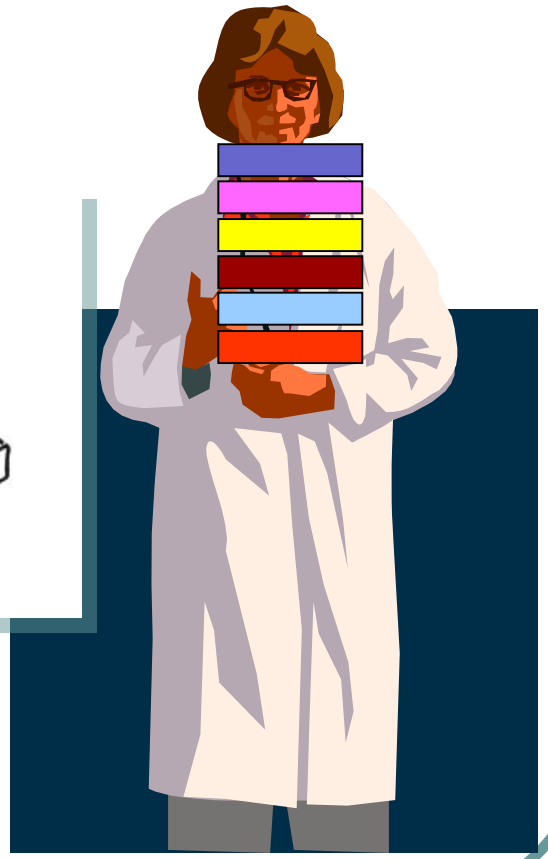
Stacks

Introduction

- Stack data structure also behaves as any other stack we would find in the real world.
- Allows to access only one data item (last item inserted).
- If you remove this item then you can access the next-to-last-item inserted, and so on.

Examples of a stack





Stacks

- A Stack is a data structure, which is a list of data elements, that all insertions and deletions are made at one end. This is the TOP (Begin) of the Stack.

- Elements are removed from a Stack in the reverse order of that in which the elements were inserted into the Stack.

Stacks

- Insertions and Deletions are restricted from the Middle and at the End of a Stack.
- The elements are inserted and removed according to the Last-In-First-Out (LIFO) principle.

Stacks

- When we add an item to a Stack, we say that we PUSH it into the stack and when we remove an item, we say that we POP it from the stack. In a Stack only the most recently inserted (“Last”) element can be removed at any time.

- Name ‘STACK’ is derived from the spring-loaded, cafeteria plate dispenser.
  - Examples:
    - Internet web browsers storing addresses of recently visited sites.
    - Text Editor function ‘undo’

Implementing a stack

- The stack implementation is based on an array. Although it's based on an array the stack restricts access. You cannot access it as you would access a normal array.

- The fields of the stack would comprise of a variable to hold the maximum size of the array (the size of the array), the array itself and a variable top which holds the index of the top of the stack.

The fields of a stack

```
int maxSize;              // size of stack array

int [] stackArray;        // array for save stack data

int top;                  // top of stack
```

Stack methods

```
Stack(int s);        // constructor
void push(int j);    // put item on top of stack
int pop();           // take item from top of stack
int peek();          // peek at top of stack
boolean isEmpty();   // true if stack is empty
boolean isFull();    // true if stack is full
```

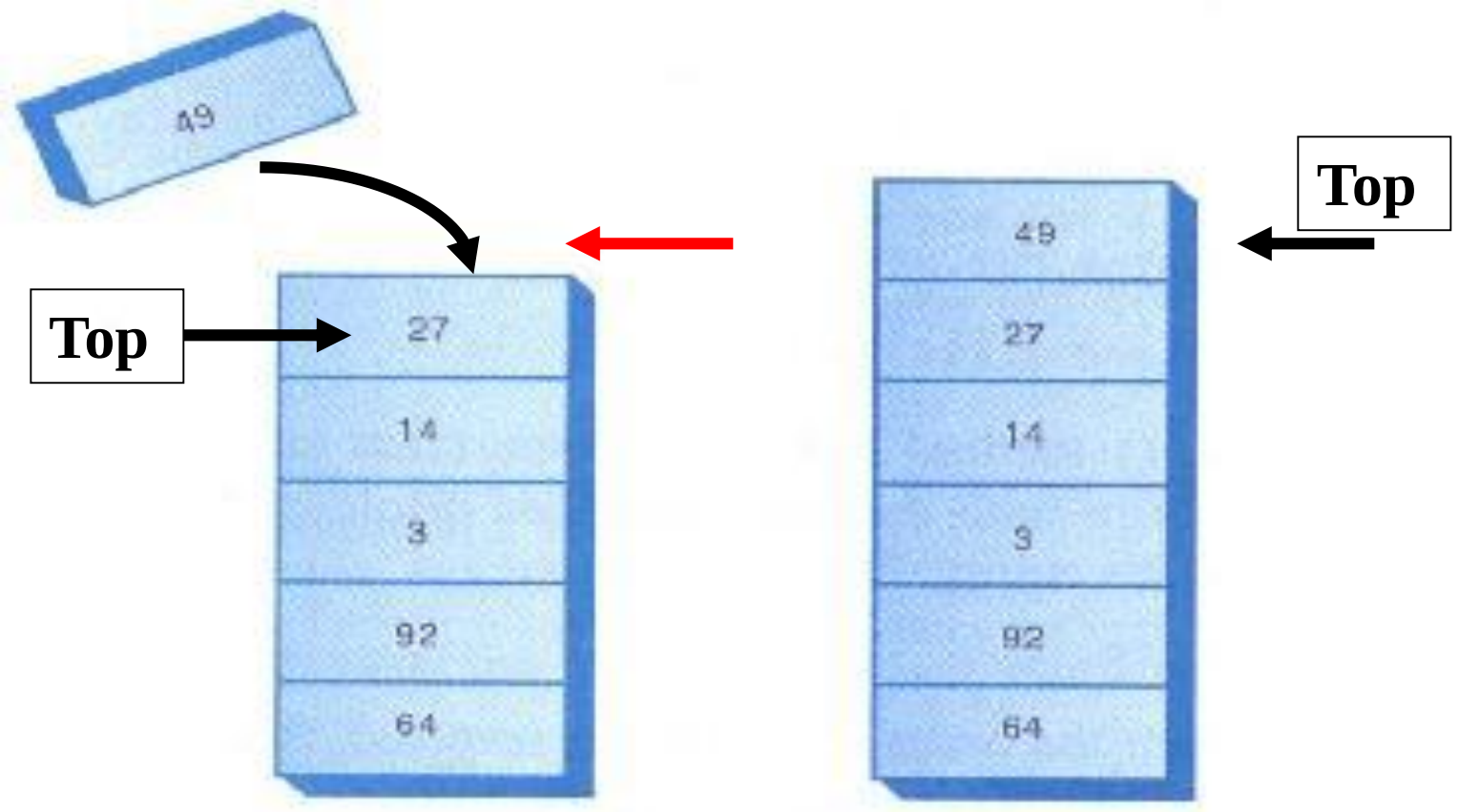
Creating a new Stack – New

- When we create a stack we should specify the length of the stack because we are using an array and the array needs to be initialized to a particular size. The ‘top’ variable is assigned the value -1 because there are no data items in the stack when it is created.

Inserting an element - Push

- Any element would be inserted to the top of the stack. Therefore before inserting an element we would increase the value of 'top' by 1 and put the element at the 'top'.
- When top is increased we should check the value of it and check if that exceeds the length of the array; if it does, then the element should not be inserted.

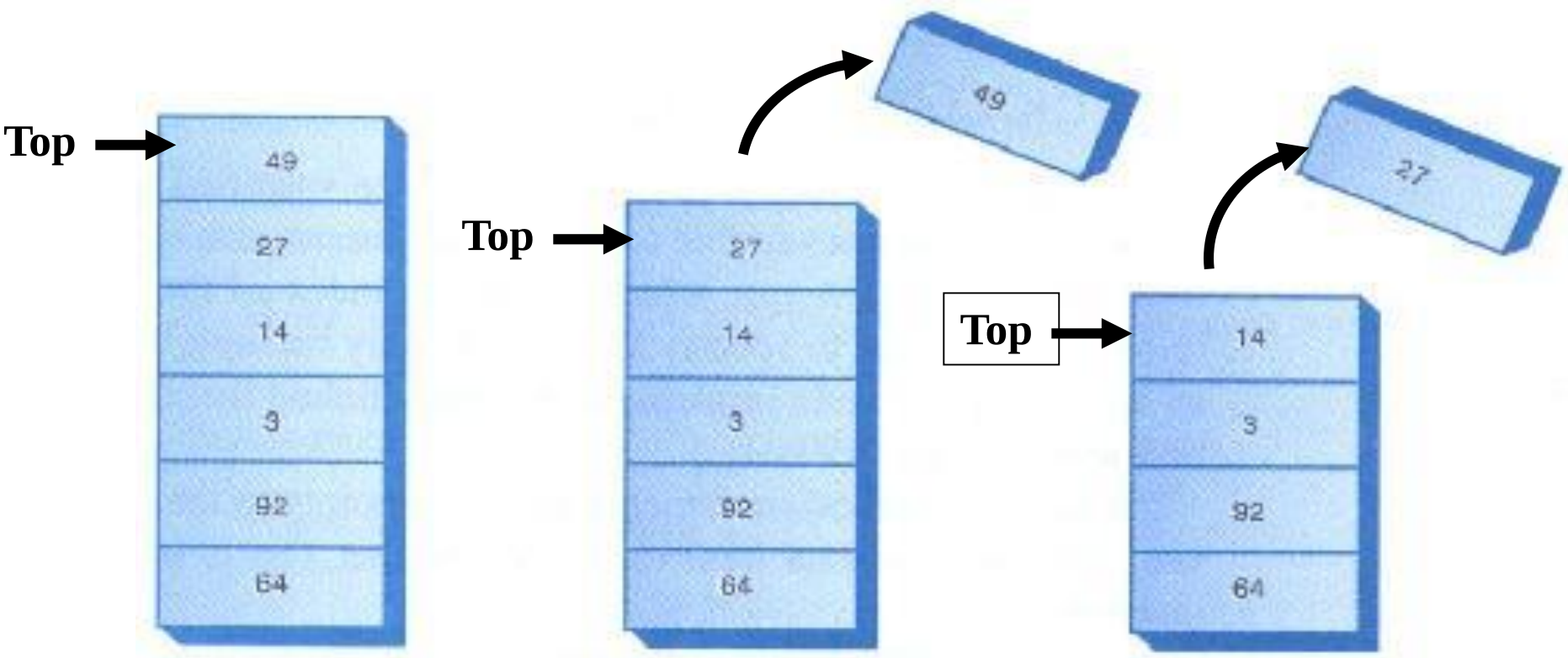
Pushing an item to the stack



Removing an Item from the top of the Stack - Pop

- When removing an item you would only be able to remove the item at the top of the stack at one time. First we should check if the stack is empty. If it isn't then the element at the top should be returned and 'top' decremented by 1.

Popping items from the stack



Efficiency of Stacks

- Items can be both pushed and popped from the stack implemented in our stack class in constant O(1) time. That is, the time is not dependent on how many items are in the stack, and is therefore very quick. No comparisons or moves are necessary.