* Merge 2 sorted arrays:

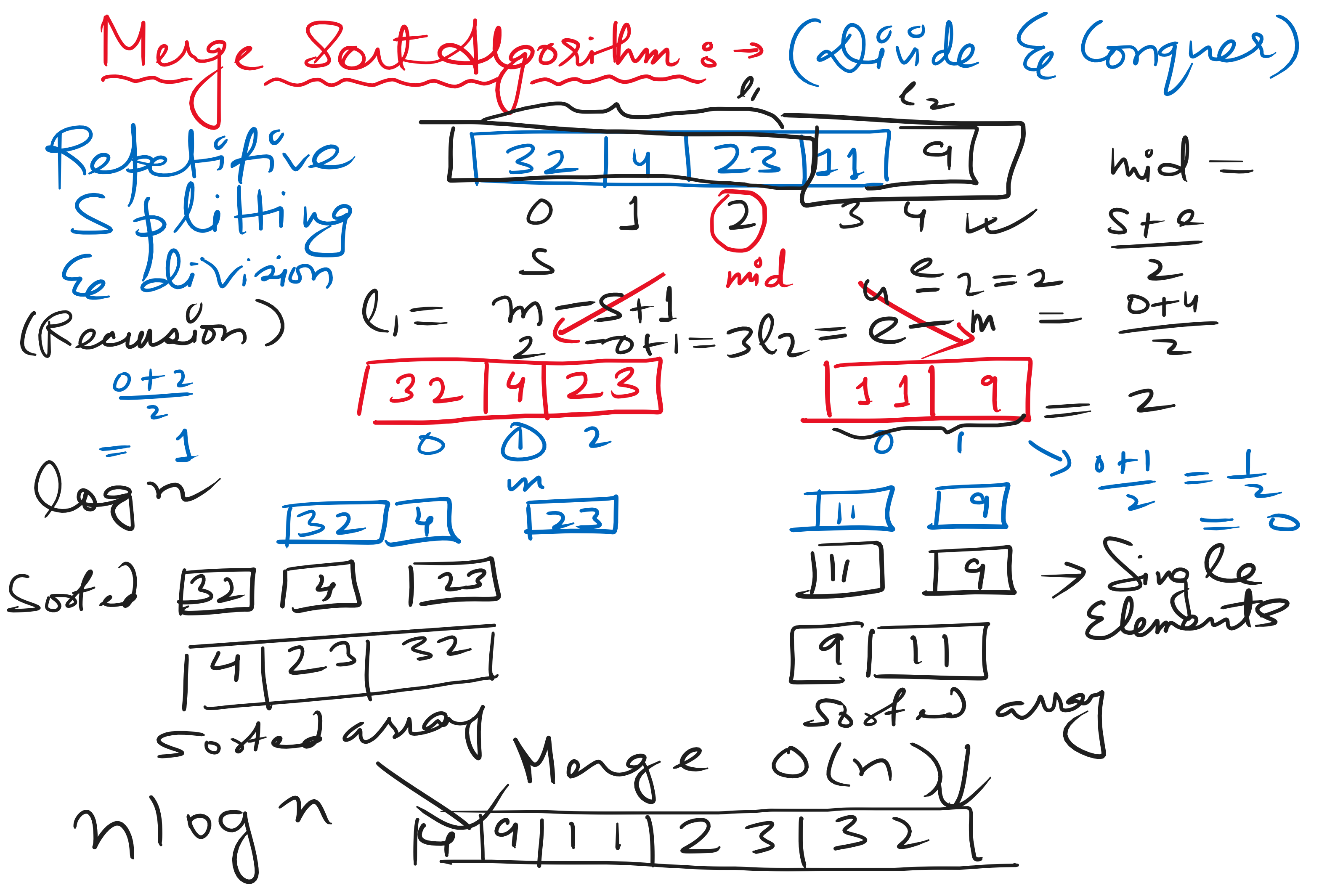
```
a = { 1, 3, 5, 7, 9 }   n1
b = { 2, 4, 6 }         n2

if (a[i] < b[j]) {
    c[k] = a[i];
    i++;
    k++; }
else {
    c[k] = b[j];
    j++;
    k++; }
k=0
c = { 1, 2, 3, 4, 5, 6, 7, 9 }
```

$i < n1$
$j < n2$

Any extra elements from any of the arrays is simply copied

Merge Sort Algorithm :→ (Divide & Conquer)



Introduction to Data Structures :→

Data Structures are the Background & base of any programming language. They allow us to perform the following common operations in general :→

- (i) Store
- (ii) Manage
- (iii) Insert
- (iv) Delete
- (v) Search
- (vi) Display
- (vii) Access

It maybe larger time or lesser time depending on the DS.

DSA450.com

Data Structures

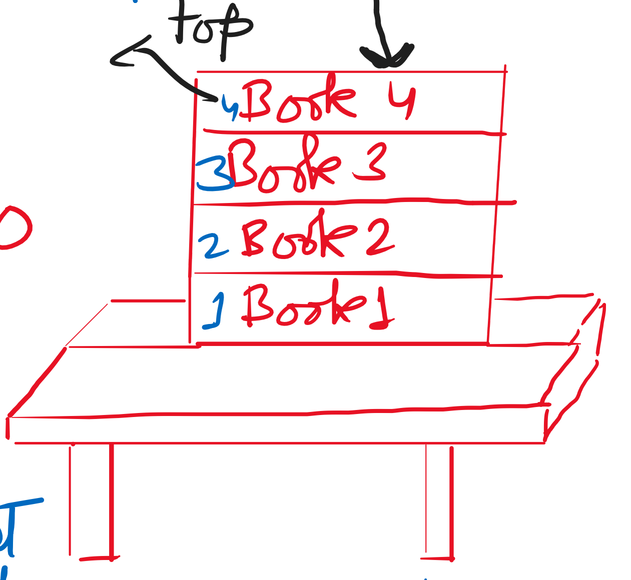
- STL → Standard Template Library
- Linear DS
  - → Arrays → 1D, 2D, MD
  - → Stacks (LIFO)
  - → Queues (FIFO)
  - → Linked Lists
    - → Singly
    - → Doubly
    - → Circular
- Non-linear DS
  - Trees **
    - → Normal Trees
    - → Binary Trees
    - → Search Trees
      - → BST
      - → AVL Tree
      - → Red Black Tree
    - → Complete Binary Tree (Heap)
    - → Segment Tree
    - → Fenwick Tree
  - Graphs**
- → Tries ****
- → K-D Tree
- → Orthogonal Range Tree
- → Binary Index Tree

peek push, pop, size, empty, ~~top~~

Introduction to Stack :→

A stack of plates
A stack of cards
A stack of books

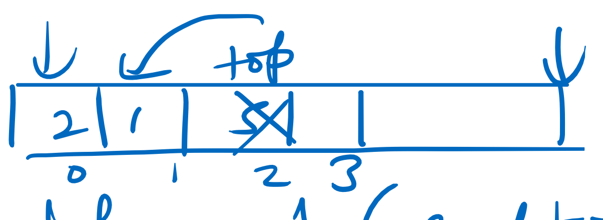
LIFO



* Book 4 is at the top so, can be accessed first
* Book 1 is at the bottom, so can be accessed at last.
* Therefore: Last In First Out ✓
  First In Last Out ✓

"Reverse"

arr



Max-size
100
99

top = -1 (empty)

top++

arr[top] = element;

~~top~~+1 = 3+1 = 4

top = -1+1 = 0