

## **Binary Search using Recursion!**

Given a sorted array, return the index of an element x in the array. Use binary search to find the element in the array !

Return -1 if the element is not present.

### **Input Format**

In the function an integer vector is passed.

### **Output Format**

Return an integer denoting index of the element to be find.

### **Sample Input**

{1, 3, 5, 7, 9}, x = 3

### **Sample Output**

1

**Solution:** recBinarySearch.cpp

## **2D Array Merge**

Implement merge sort for a two-dimensional array. In case of odd dimension, the first division contains more number of elements than the second one. The complete execution of merge sort arranges the elements in increasing order either moving row-wise or column-wise.

For example, let there be a

**4×4**

4×4 two dimensional array. The complete process to be implemented is illustrated in Fig. 1.

Similarly, Fig. 2 demonstrates the scenario for a

**3×3**

3×3 two dimensional array. One has to keep on dividing till a single element is remaining. During merging, first the row elements get sorted in increasing order followed by sorting of elements lying in the same column.

|    |    |    |    |
|----|----|----|----|
| 18 | 4  | 16 | 8  |
| 23 | 13 | 20 | 11 |
| 28 | 24 | 26 | 25 |
| 1  | 30 | 15 | 19 |

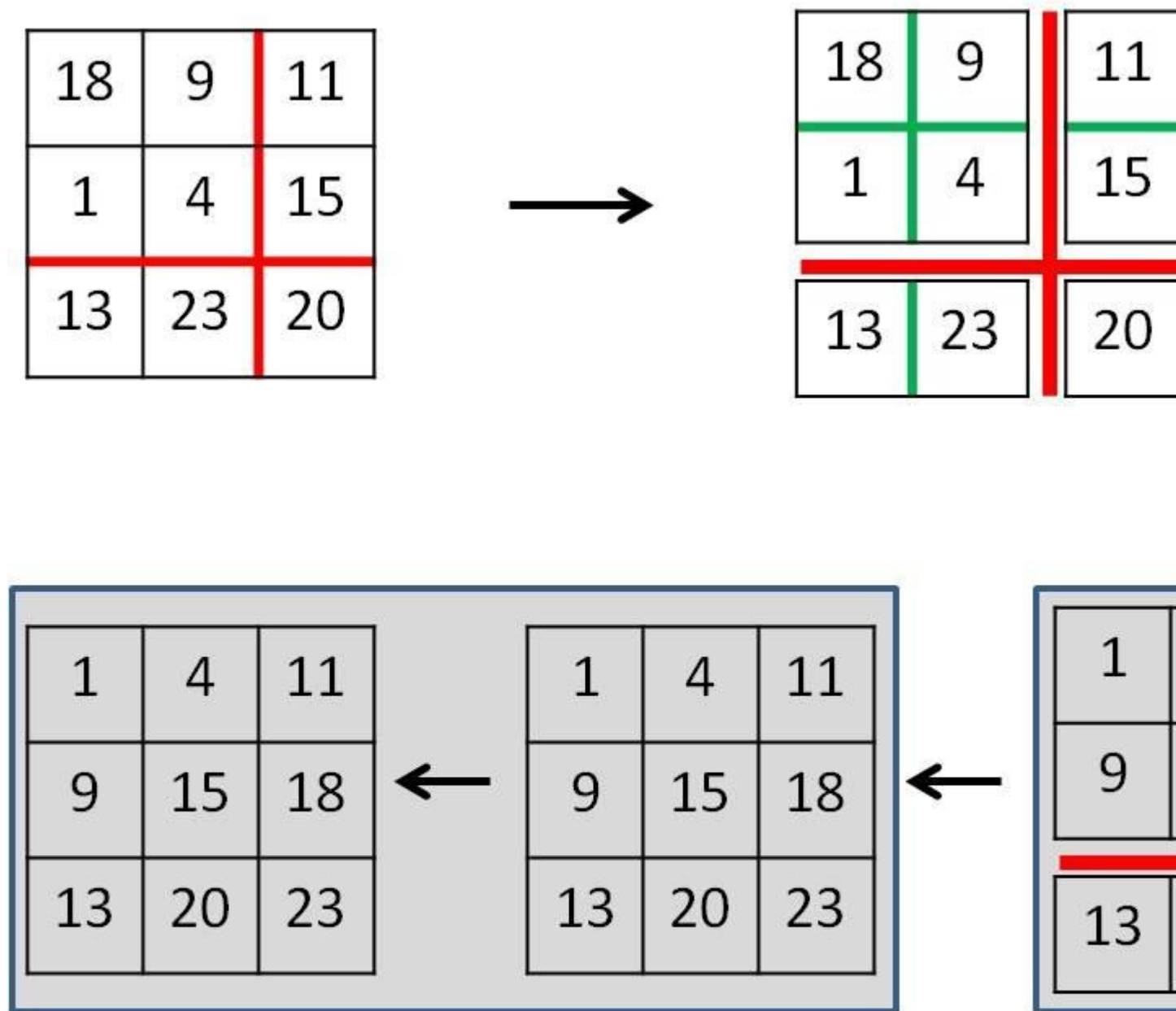


|    |    |
|----|----|
| 18 | 4  |
| 23 | 13 |
| 28 | 24 |
| 1  | 30 |

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 1  | 8  | 16 | 18 | 4  | 8  | 16 | 18 |
| 4  | 13 | 19 | 23 | 11 | 13 | 20 | 23 |
| 11 | 15 | 20 | 28 | 1  | 15 | 19 | 28 |
| 24 | 25 | 26 | 30 | 24 | 25 | 26 | 30 |

Merging first row-wise  
followed by column-wise

Fig.1 - 2D Mergesort on a  
4×4  
4×4 two dimensional array.



Merging first row-wise  
followed by column-wise

**Solution:** mergeSort.cpp

## Game of Greed

You are playing a game with your 'k' friends. You have an array of N coins, at each index i you have a coin of value  $a[i]$ .

Your task is to divide the coins, among a group of K friends by doing consecutive segments (k-subarrays) of the array.

Each friend will get a total sum of the coins in that subarray. Since all your friends are greedy, and they will pick the largest k - 1 segments and you will get the smallest segment. Find out the maximum value you can make by making an optimal partition.

Note : The coins array may or may not be sorted!

(Refer Hints at the end if needed)

### **Input**

K = 3

coins = {1, 2, 3, 4};

### **Output**

3

### **Explanation**

The ideal partition looks like this -

$\{1 + 2\} = 3$

$\{3\} = 3$

$\{4\} = 4$

You will get a maximum of 3 coins in the best case.

**Solution:** gameOfGreed.cpp

## Reading Books

Given number of pages in n different books and m students. The books are arranged in **any order (not necessarily ascending)** of number of pages. Every student is assigned to read some **consecutive segment of books**. The task is to assign books in such a way that the *maximum number of pages assigned to a student is minimum*.

Find the maximum number of pages that a student will have to read.

### **Sample Input**

books = [10, 20, 30, 15] students = 2

### Sample Output

45

### Explanation

There are 2 number of students. Books can be distributed in following fashion :

1) [10] and [20, 30, 15]

Max number of pages is allocated to student

2 with 65 pages

2) [10, 20] and [30, 15]

Max number of pages is allocated to student

2 with  $30 + 15 = 45$  pages

3) [10, 20, 30] and [15]

Max number of pages is allocated to student

1 with 60 pages

Of the 3 cases, Option 2 has the minimum pages = 45.

**Solution:** readingBooks.cpp