

#  
1 to 9

5	3	1	2	7	6	4	9	8
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

A partially filled sudoku which is valid.

1 to 9

```
public static boolean Print(int[][] grid, int row, int col) {
    if(grid[row][col]!=0) {
        return Print(grid, row, col+1);
    }
    else {
        for (int val = 1; val <=9; val++) {
            if(issafe(grid,row,col,val)) {
                grid[row][col]=val;
                boolean ans=Print(grid, row, col+1);
                if(ans==true) {
                    return true;
                }
                grid[row][col]=0;
            }
        }
        return false;
    }
}
```

	0	1	2	3	4	5	6	7	8
0	5	3	1	2	7	6	4	9	8
1	6			1	9	5			
2		9	8					6	
3	8				6				3
4	4			8		3			1
5	7				2				6
6		6					2	8	
7				4	1	9			5
8					8			7	9

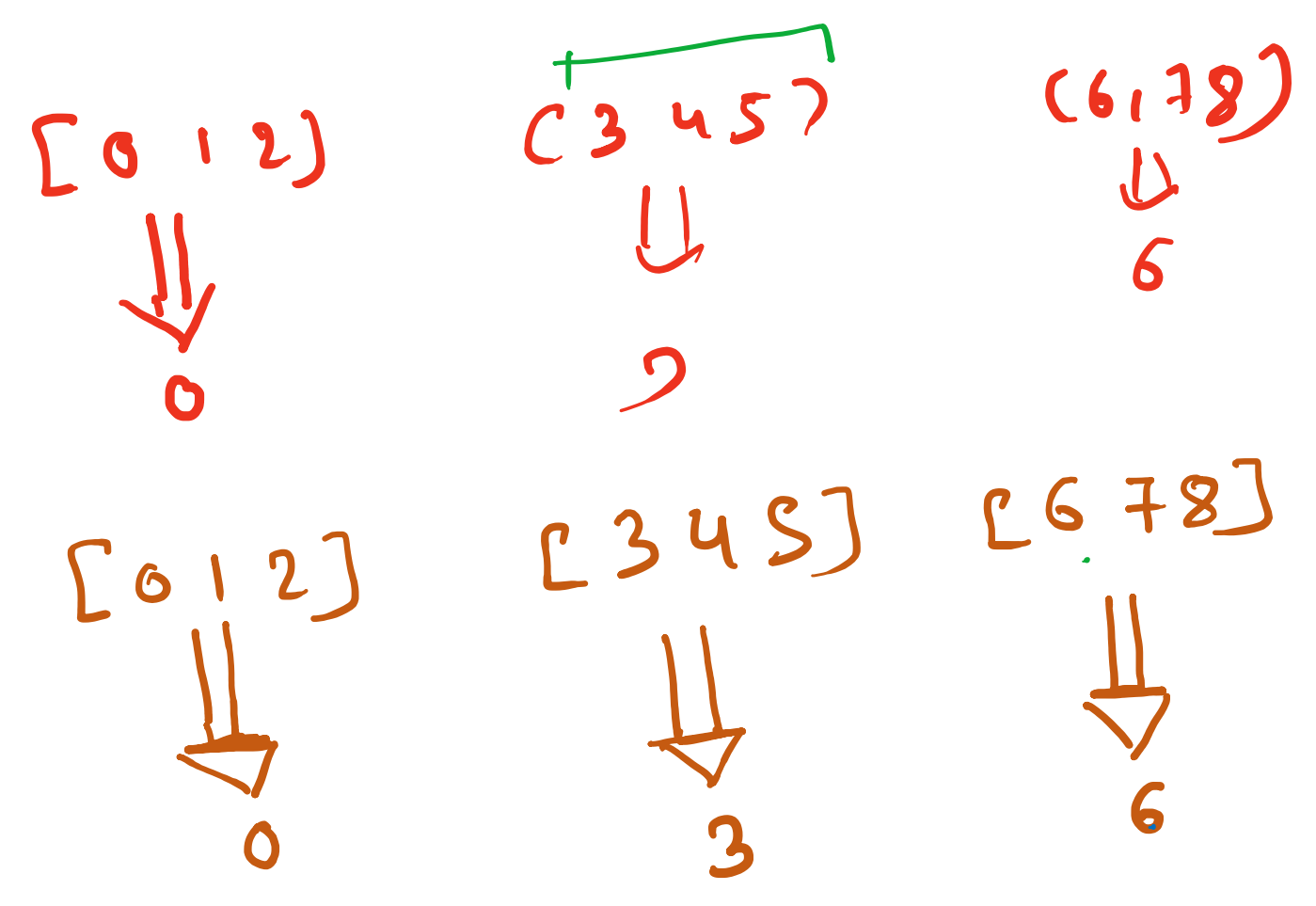
A partially filled sudoku which is valid.

A partially filled sudoku which is valid.

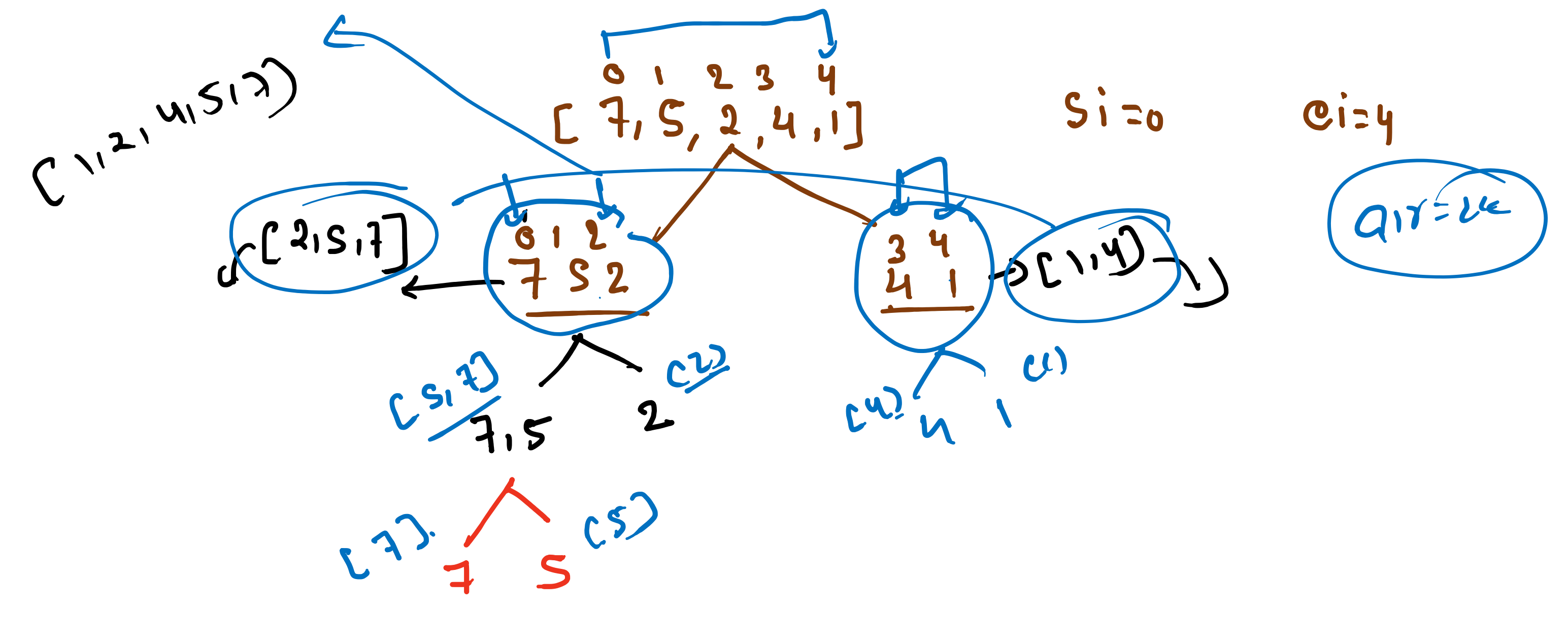
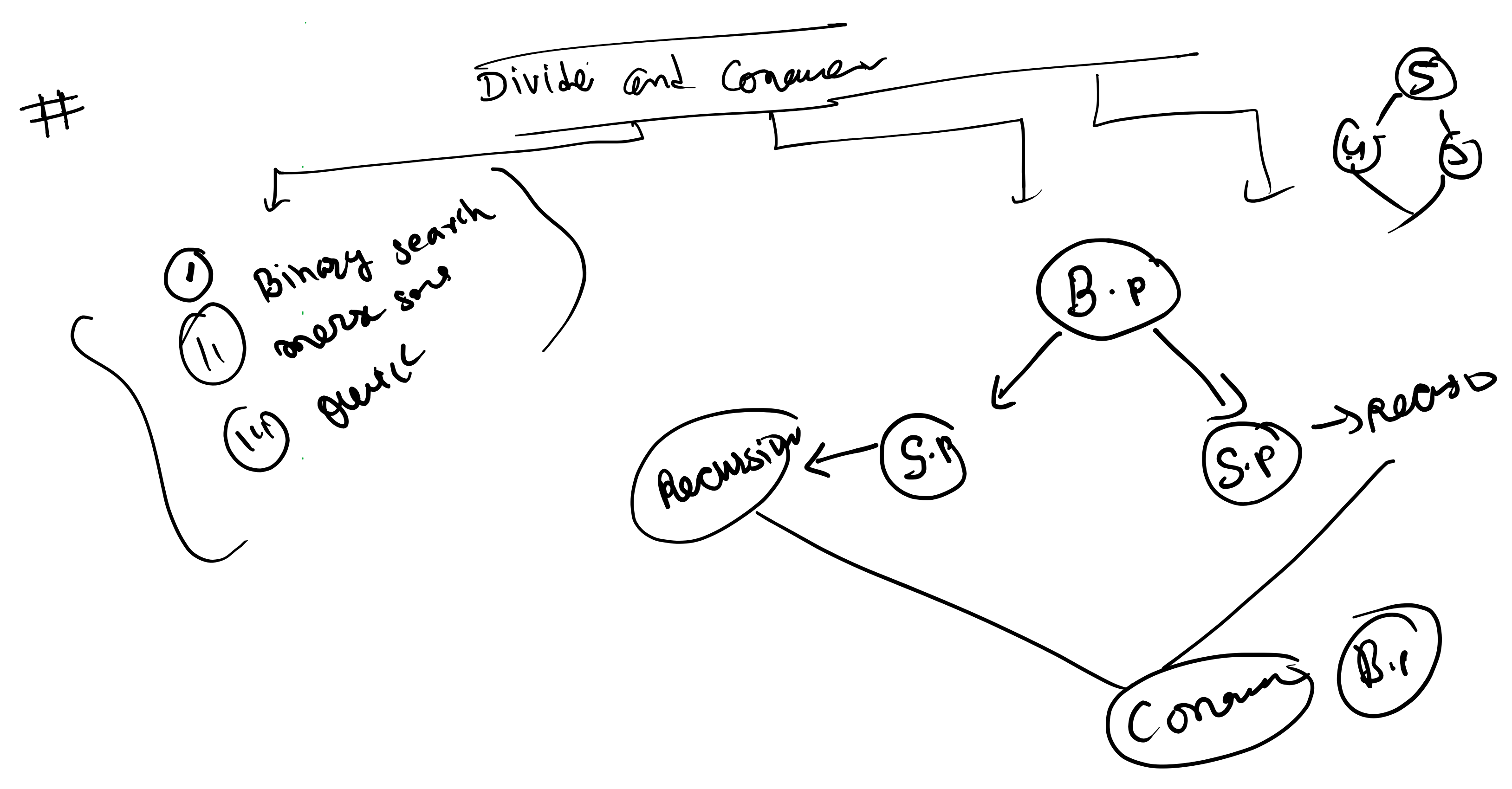
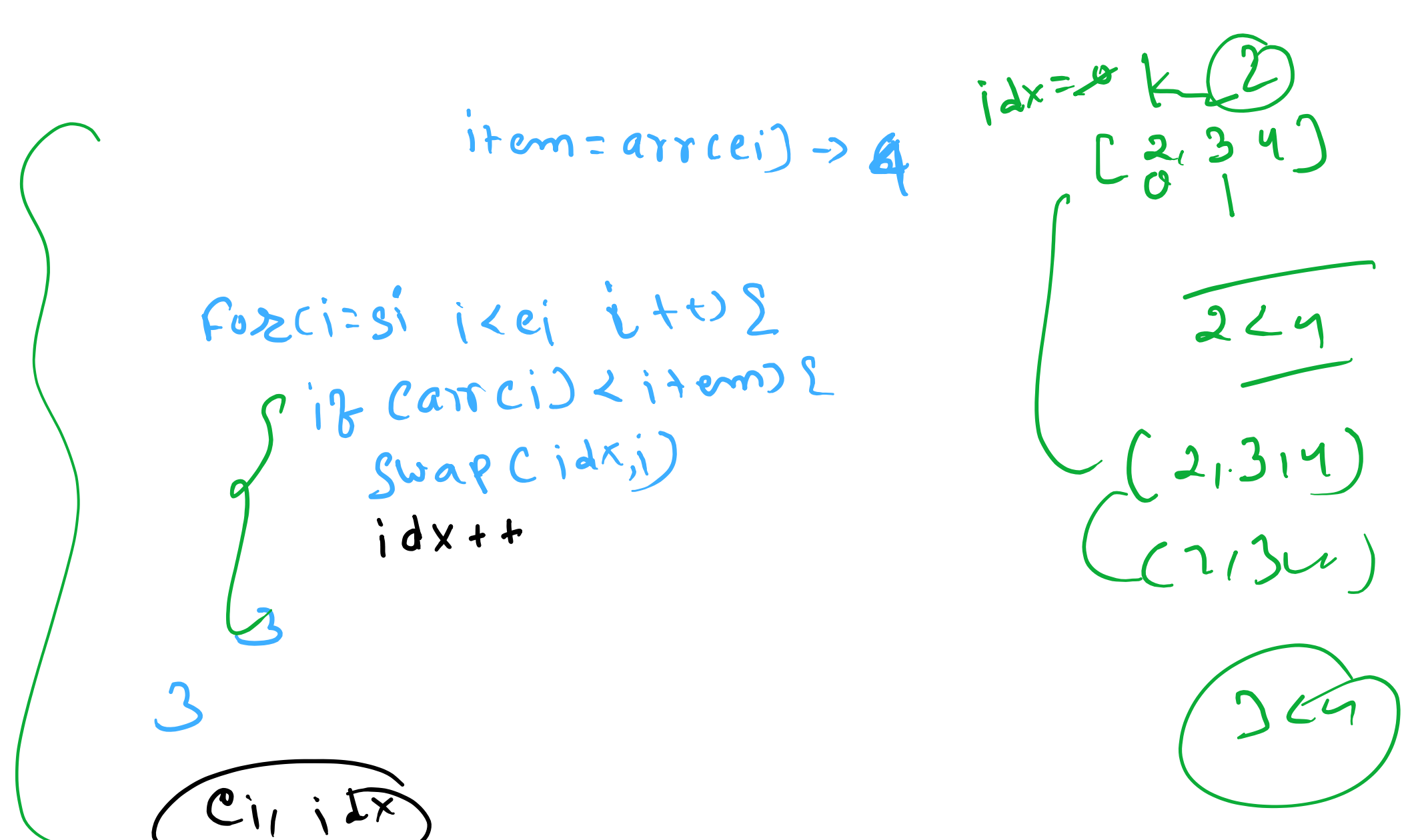
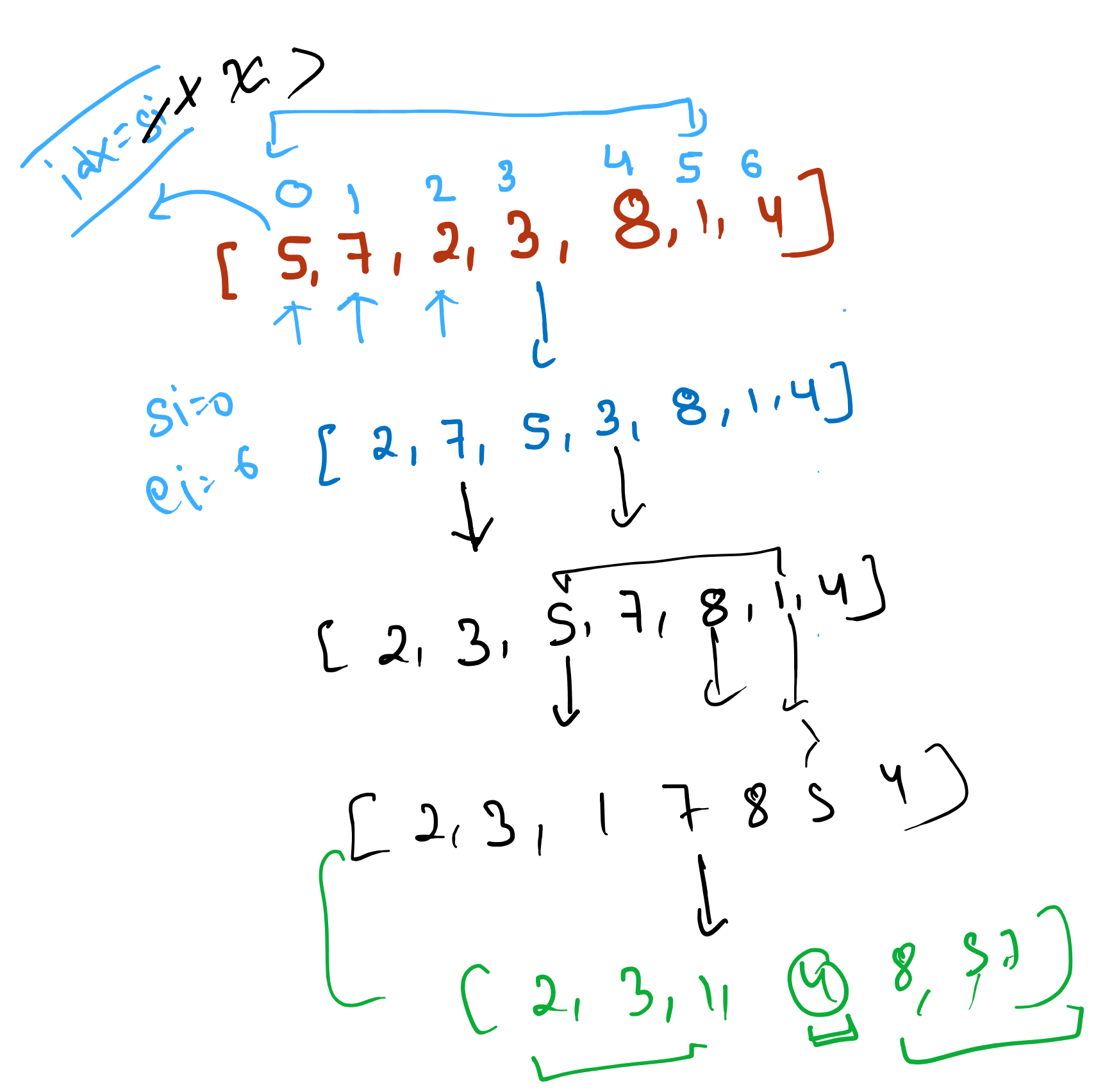
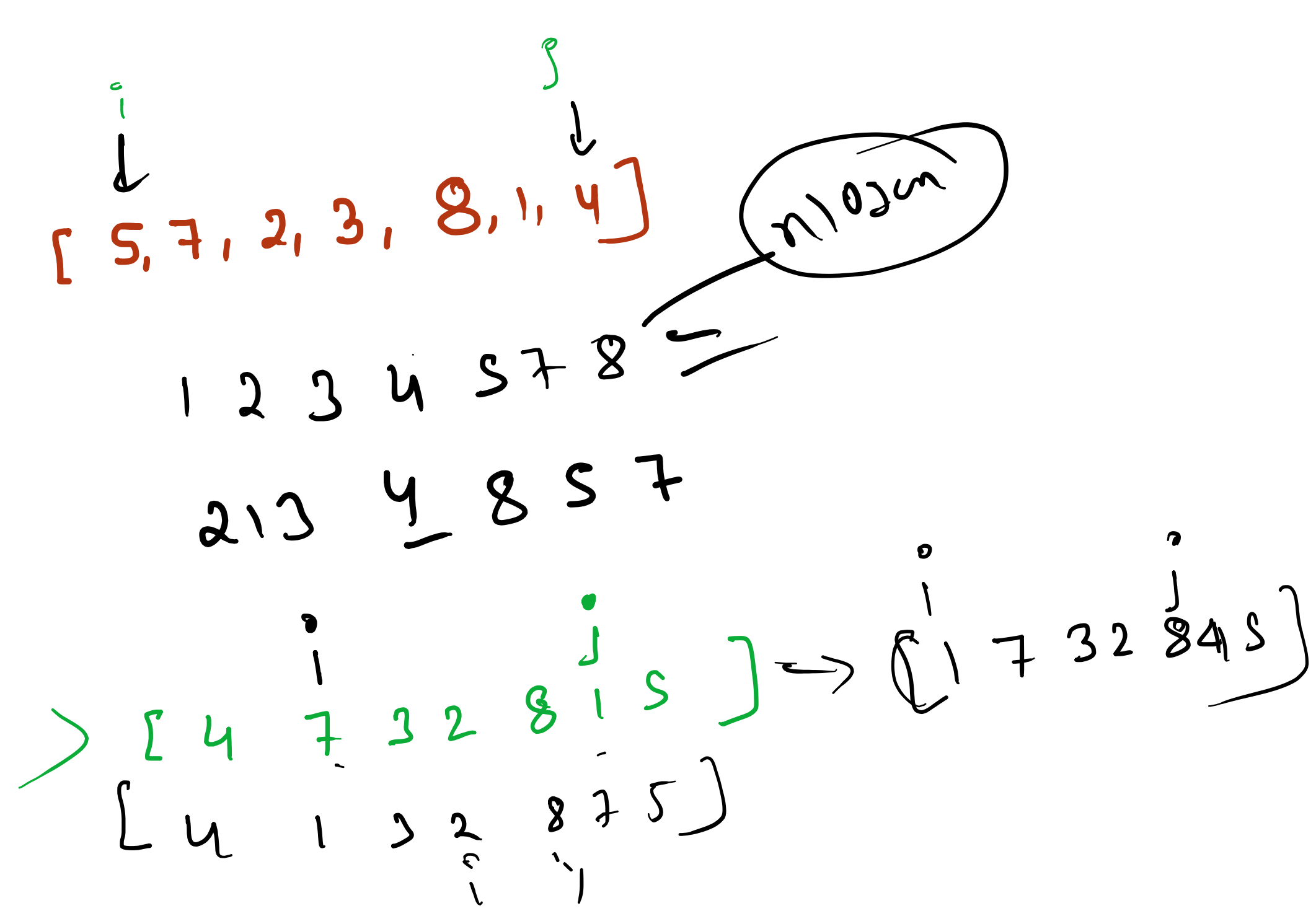
	0	1	2	3	4	5	6	7	8
0	5	3			7				
1	6			1	9	5			
2		9	8					6	
3	8				6				3
4	4			8		3			1
5	7				2				6
6		6					2	8	
7				4	1	9			5
8					8			7	9

A partially filled sudoku which is valid

A partially filled sudoku which is valid.



$$r = \text{row} - \text{row} \% 3$$
$$c = \text{col} - \text{col} \% 3$$



```
public static int[] Sort(int[] arr, int si, int ei) {
    if (si == ei) {
        int[] bs = new int[1];
        bs[0] = arr[si];
        return bs;
    }
    int mid = (si + ei) / 2;
    int[] f = Sort(arr, si, mid);
    int[] s = Sort(arr, mid + 1, ei);
    return Merge(f, s);
}
```

