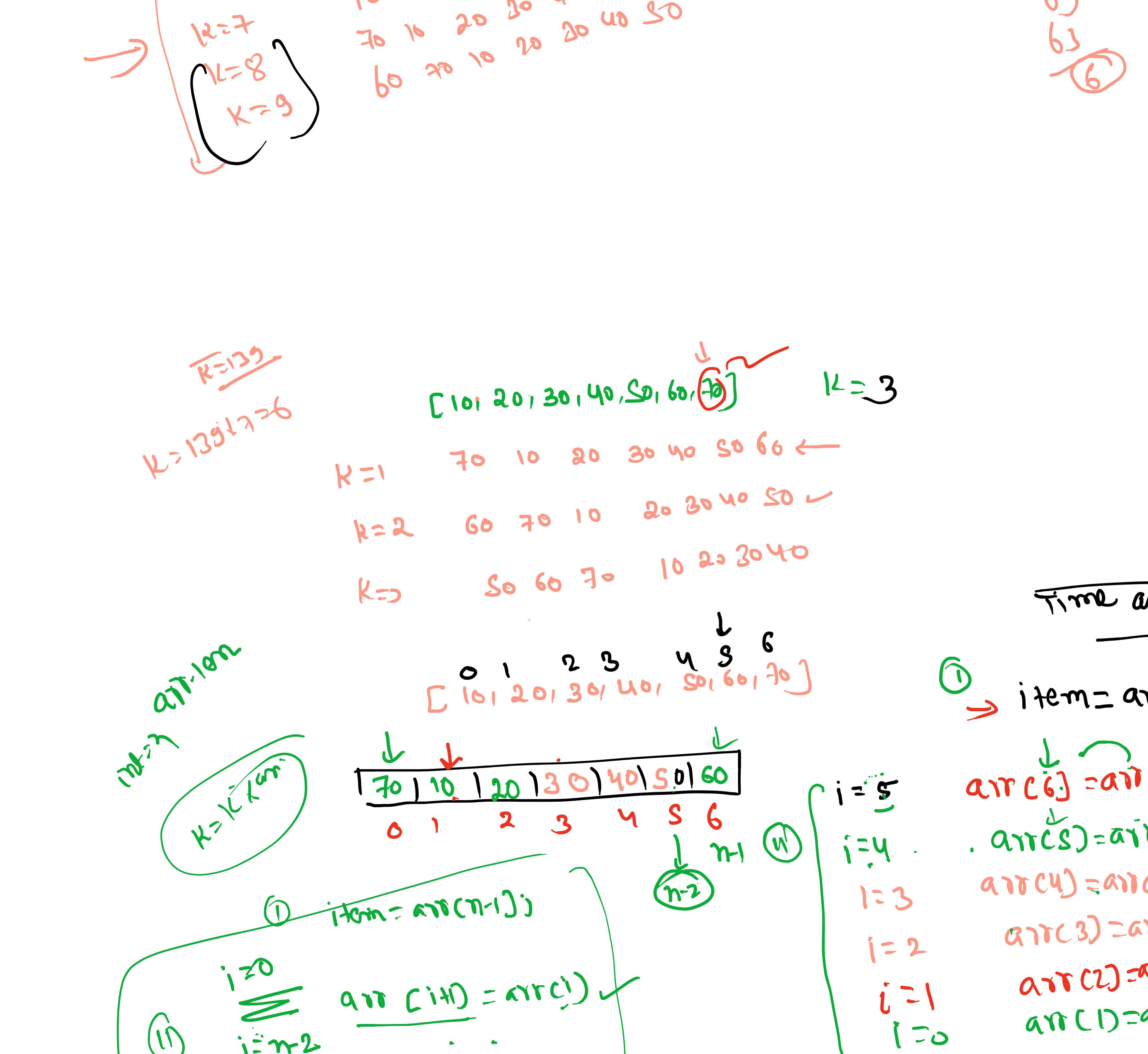


[10, 20, 30, 40, 50, 60, 70] K=1
50, 60, 70, 10, 20, 30, 40
40 50 60 70 10 20 30 K=1



```
public static void Rotate(int[] arr, int k) {  
    int n = arr.length;  
    k = k % n;  
    for (int j = 1; j <= k; j++) {  
        int item = arr[n - 1];  
        for (int i = n - 2; i >= 0; i--) {  
            arr[i + 1] = arr[i];  
        }  
        arr[0] = item;  
    }  
}
```

$k=2$ $n=2$

$j=1$

$i=5$

$i=4$

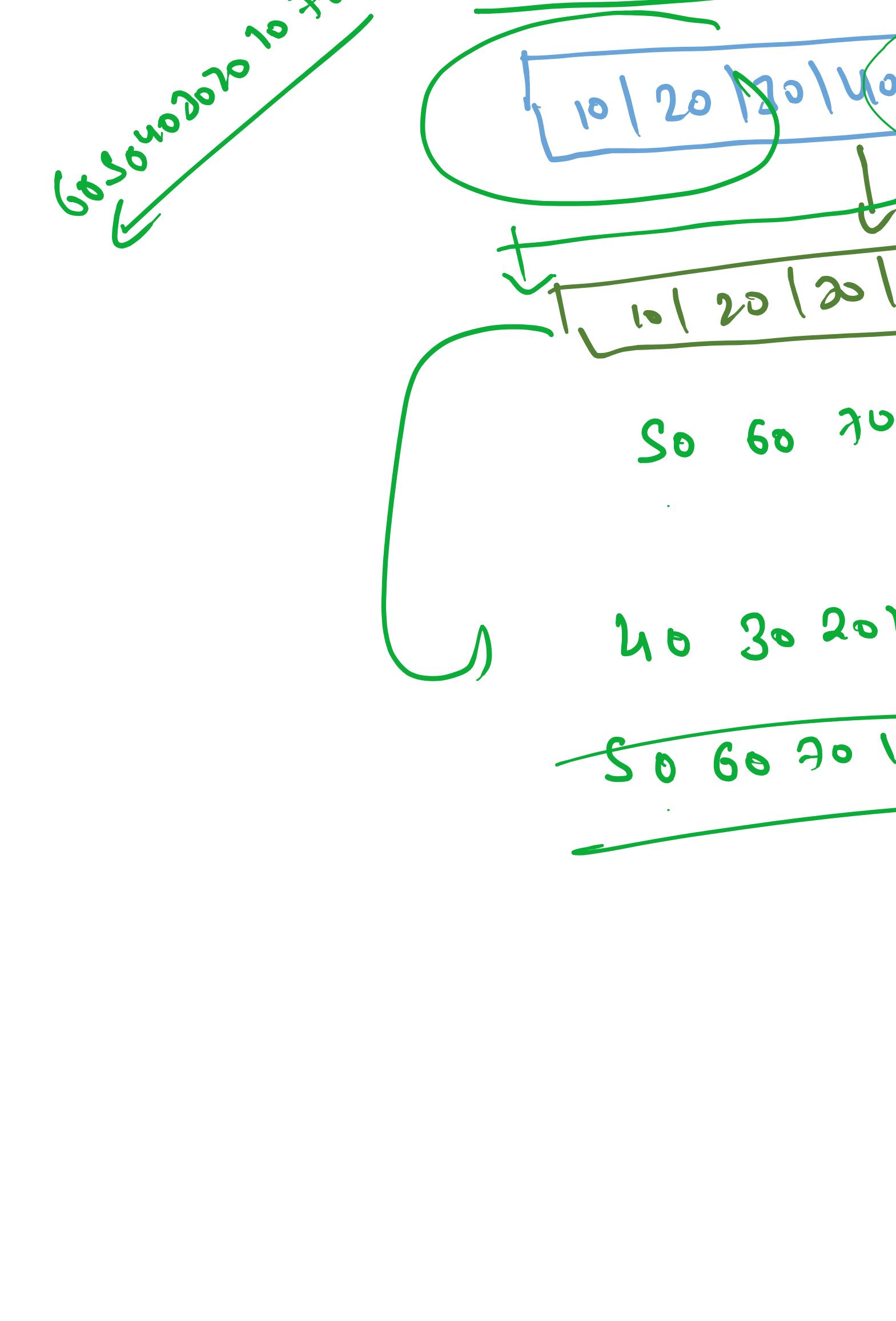
$i=3$

$i=2$

$i=1$

$i=0$

Reversal At 60°



an (1) - .

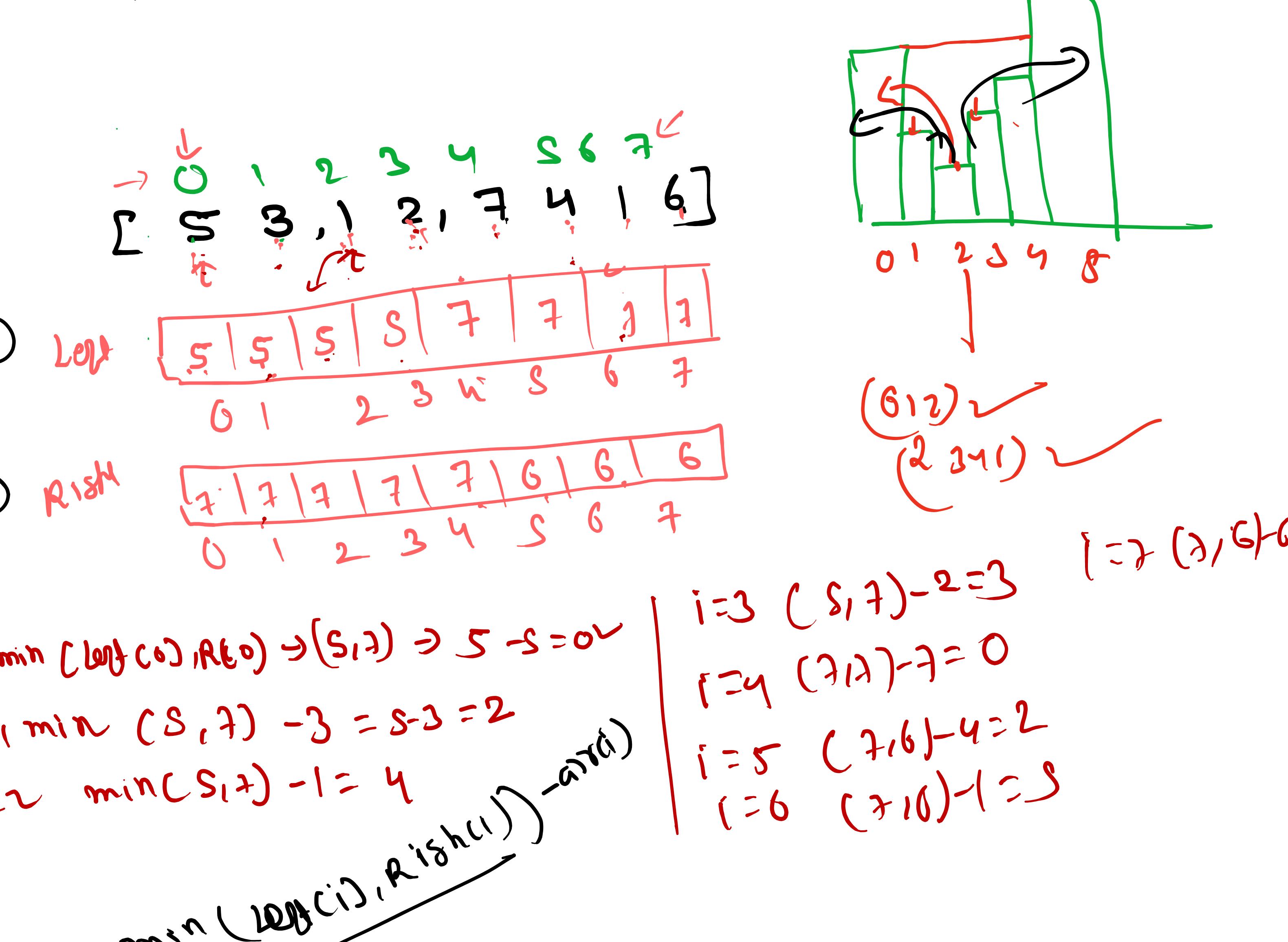
- A hand-drawn diagram illustrating set operations. At the top, there are two separate sets enclosed in curly braces: the first set contains elements $n-k$ and k ; the second set contains elements k and \sim . Below these, a large oval encloses the union of the two sets, labeled "all elements".

[4,2,0,3,2,5]

{ 4 , 2 , 0 , 3 , 2 , 5 }

2 + 4 + 1 + 2
= 5

A hand-drawn diagram showing a stack of six rectangles on a horizontal surface. The rectangles are labeled with green numbers: 2, 4, 1, 2, 2, and 5. A green line starts at the top left and connects the top edges of the first four rectangles. A red 'X' is drawn above the fifth rectangle. A red arrow points from the bottom of the fifth rectangle to the right. The sixth rectangle is partially visible at the bottom.



$\times 10^{-2}$ 0.123125267416

i=1 LCIJ = max [LC0], arr CI
j=2, [2] = max [C1]

Diagram illustrating the construction of a suffix tree for the string "77777666".

The tree is shown as a forest of suffixes:

- Root node: "77777666"
- Level 1: "77777666" (left), "666" (right)
- Level 2: "7777766" (left), "66" (right), "6" (right)
- Level 3: "777776" (left), "66" (right), "6" (right)
- Level 4: "77777" (left), "6" (right), "6" (right)
- Level 5: "7777" (left), "6" (right), "6" (right)
- Level 6: "777" (left), "6" (right), "6" (right)
- Level 7: "77" (left), "6" (right), "6" (right)
- Level 8: "7" (left), "6" (right), "6" (right)
- Level 9: "6" (left), "6" (right), "6" (right)

Labels used in the diagram:

- $left(i)$: Identifies the leftmost node at level i .
- $arr(i)$: Identifies the edge from the leftmost node at level $i-1$ to the leftmost node at level i .
- $L(i)$: Identifies the label of the leftmost node at level i .

Calculation of $L(i)$:

$$L(i) = \max(L(i-1), arr(i))$$

Calculation of $arr(i)$:

$$arr(i) = \min(L(i), R(i)) - arr(i-1)$$

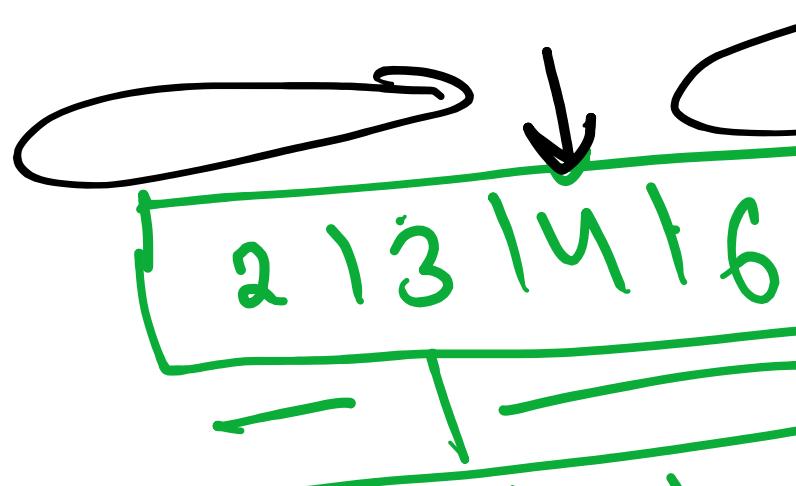
Calculation of $arr(0)$:

$$arr(0) = arr(0) = 0$$

$\text{ans} := \min(L(i), R(i))$

$\sum_{i=0}^{n-1} \text{ans} +$

$\text{ans} = 14$



$i=3 \quad R(3) = \max(R(4), a) \tau(3)$

$i=2 \quad R(2) = \max(R(3), a) \tau(2)$

$i=1 \quad R(1) = \max(R(2), a) \tau(1)$

\vdots $i=0 \quad R(0) = \max(R(1), a) \tau(0)$

A diagram illustrating a binary search tree structure. The root node is 72, with 48 as its left child and 35 as its right child. Node 48 has left child 14 and right child 5. Node 14 has left child 2 and right child 4. Node 5 is a leaf node. The tree is shown with green lines connecting the nodes and green arrows indicating the search path from the root to node 5.

The tree structure is as follows:

- Root: 72
- Left child of 72: 48
- Right child of 72: 35
- Left child of 48: 14
- Right child of 48: 5
- Left child of 14: 2
- Right child of 14: 4
- Node 5 is a leaf node.

Search path from root to node 5:

- From 72 to 48 (down arrow)
- From 48 to 14 (down arrow)
- From 14 to 2 (down arrow)
- From 2 to 5 (down arrow)

A hand-drawn diagram illustrating a merge sort step. At the top, a green box contains four numbers: 72, 29, 6, and 1. Below it, two green boxes show intermediate steps: one with 12 and 48, and another with 36 and 29. A green checkmark is to the left of the second box. To the right, a red bracket groups the numbers 2, 3, 4, and 6. Below this, a black box contains 72, 148, 36, and 29, with indices 0, 1, 2, and 3 underneath. Red arrows point from the green boxes down to the black box.

(1)

$$L_{\text{eff}}(0) = 1$$

i = 1 $L(1) = L(0) \times \text{arr}(0)$

i = 2 $L(2) = L(1) \times \text{arr}(1)$

i = 3 $L(3) = L(2) \times \text{arr}(2)$