(7) Rotate matrix by 90

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Rotate clockwise

top, left, right

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 & 1 & 2 \\ 7 & 5 & 3 \\ 8 & 9 & 6 \end{bmatrix}$$

bottom

Brute force:

① Sort

② for(i=0 — n)
{ int start = arr[i][0]
int end = arr[i][1]
// ~~add merged ans into the list~~ Skip all the merged interval.
for(int j=i+1; j<n; j++)
{ if(arr[j][0] <= end)
{ end = Math.max(end, arr[j][1])
} else
{ break;
}
}
// add merged interval into the list
}

Optimal approach:

① sort

② (1,3) (2,4) (2,6) (8,9) (8,10) (9,11) (15,18) (16,17)

(2<=3) (2<=4) 8!<=6 8<=9 9<=10 15!<=11 16<=18

(1,3) ⇒ (1,4) ⇒ [(1,6)]
(8,9) ⇒ (8,10) ⇒ [(8,10)]
(15,18) ⇒ [(15,18)]

③ for(int i=0; i<n; i++)
{ if(ans.empty() || arr[i][0] > ans.back()[1])
{ ans.push_back(arr[i]);
} else
ans.back()[1] = max(ans.back()[1], arr[i][1])
}
return ans;

(9) Merge two ~~Overlapping~~ Sorted Arrays:
(without using extra space)

→ arr1[left] >= arr2[right]: Swap & move to next
→ arr1[left] <= arr2[right]: Stop moving pointer
→ now arr1 contains all smaller elements & arr2 contains all bigger elements.

arr1 | 1 | 4 | 8 | 10 |

arr2 | 2 | 3 | 9 |

(10 > 2) (Swap)

arr1 | 1 | 4 | 8 | 2 |

arr2 | 10 | 3 | 9 |

(8 > 3) (Swap)

arr1 | 1 | 4 | 3 | 2 |

arr2 | 10 | 8 | 9 |

(4 < 9) (Stop moving)

Sort(arr1)
Sort(arr2)

| 1 | 2 | 3 | 4 |

| 8 | 9 | 6 |

(10) Find duplicate in an array of N+1 integers.

(there is only one duplicate no.)

arr = [1, 3, 4, 2, 2] $\overset{N+1}{(4+1)}$ elements

(N=4) : must contain

(1 2 3 4) ^ (1 3 4 2 x)

(3 ^ 2) xor (1 3 4 2) ^ (1 2 3 4 x)

duplicate no.

(11) Find repeating & Missing No:

given array = {3, 1, 2, 5, 3} with given values in range of [1, N]

Result: {3, 4} '4' is missing & '3' is repeated

Brute force approach

Store freq of each element & return ans

Best approach:

math (Sum of 'N' no. is)

$S_n = \left(\frac{N*(N+1)}{2}\right)$

S = Summation of all elements of the array

$S - S_n = X - Y$ ——— (1)

S_{n_2} = Sum of Squares of first N no.

$= N*(N+1)*(2N+1)/6$

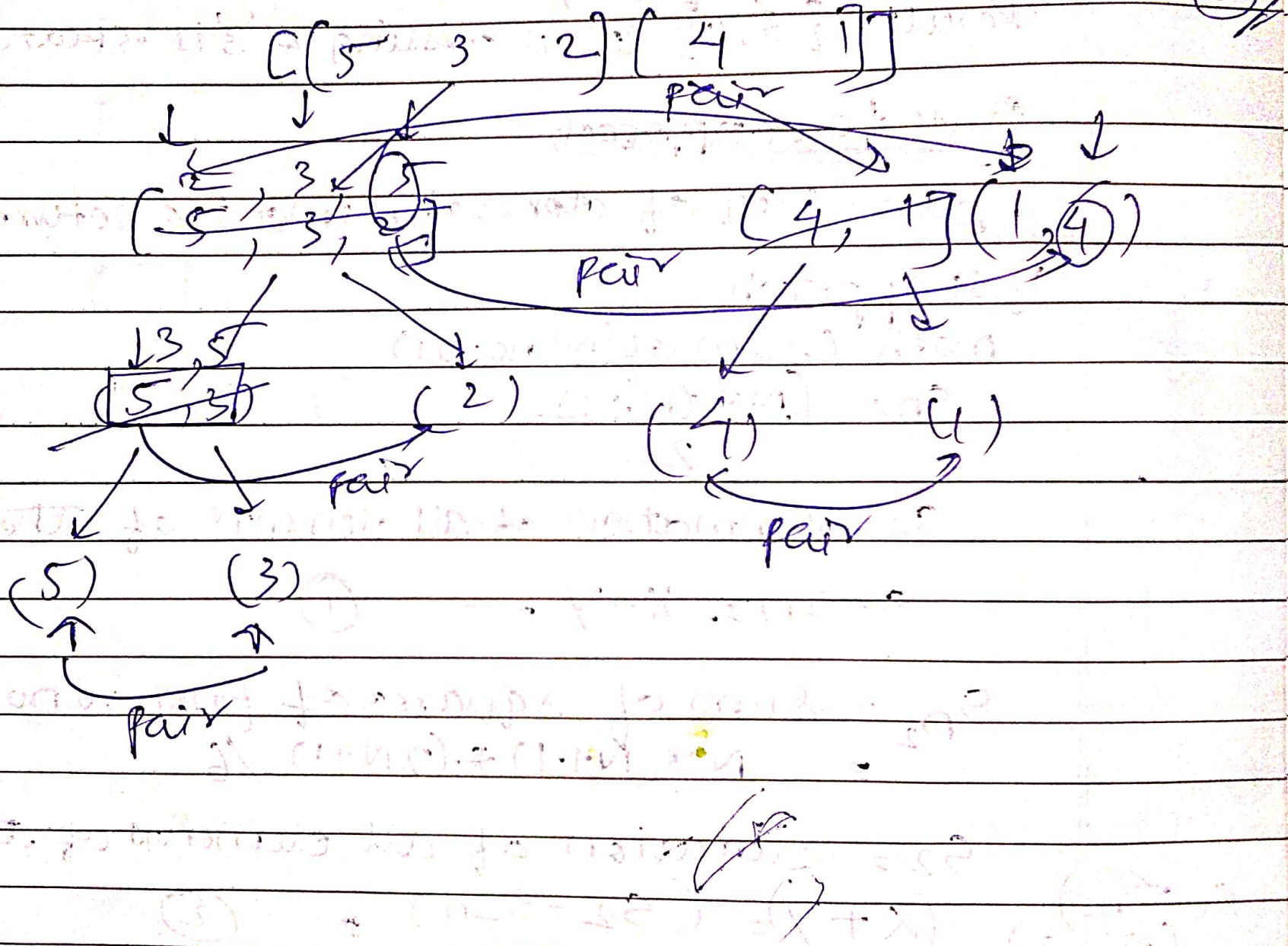
S_2 = Sumation of all elements of the array

$(S_2 - S_{n_2}) = (x^2 - y^2)$

$(X+Y) = \frac{(S_2 - S_{2n})}{(X-Y)}$ ——— (2)

solve linearly.

(12) Count Inversion: and count = 1+2+1+1+2 (5)



Merge two Sorted Arrays (Gap method)

- based on Shell Sort
- Formula:

  initial gap = ceil(Size of arr1[ ] + Size of arr2[ ])
  (btw left & right ptr)

- Steps:

→ assume two arrays as '1' and calculate initial gap between left & right ptr.

→ Run a loop: (left=0, right= left+gap)

if left in arr1 & right in arr2 : if (arr1[left] > arr2[right-n] Swap

if both in arr2 : if (arr1[left-n] > arr2[right-n] Swap

if both in arr1 : if (arr1[left] > arr2[left] Swap

→ After right pointer reaches end, decrease gap by ceil(current gap/2)

→ Repeat until gap > 0

Code: gap = (len/2) + (len%2); ~~left=0, right=left+~~ gap

while(gap>0), int left=0, right = left+gap;

{ if (left<n && right>=n)

while(right <len) SwapIfGreater(arr1, arr2, left, right-n);

else if (left>=n)

SwapIfGreater(arr2, arr2, left-n, right-n);

else

} left++, right++; SwapIfGreater(arr1, arr1, left, right)

if(gap==1) break; gap = (gap/2) + (gap%2);

}

Find the duplicate in array of N+1 integers:

Floyd method: