Merge Sorted Arrays (with Extra Space)

A1: 2 4 5 7 8 — O(N)

A2: 1 3 6 9 10 11 12 — O(M)

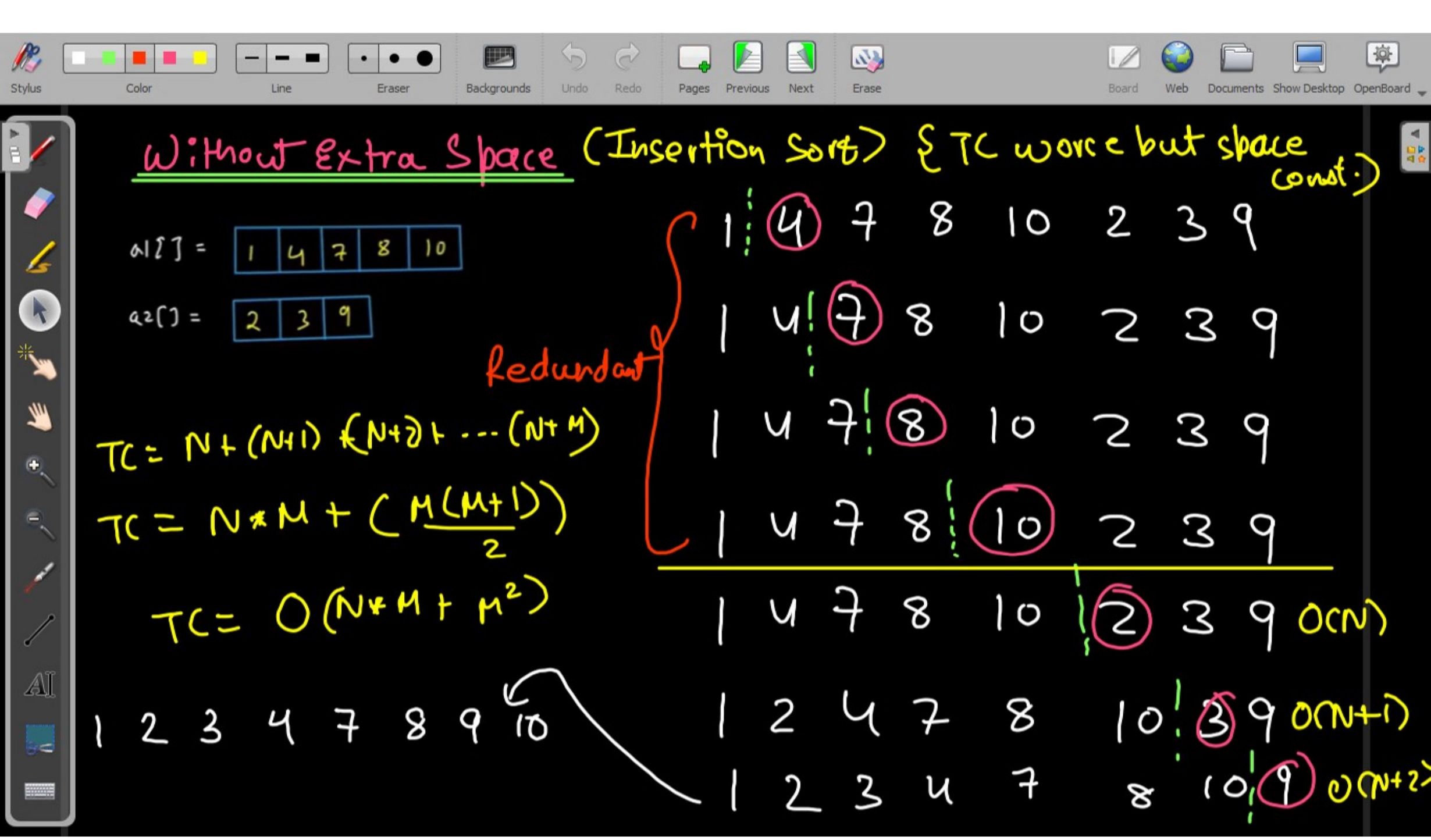
ans = 1 2 3 4 5 6 7 8 9 10 11 12

TC = O(N+M)

SC = O(N+M)

Code With Extra Space

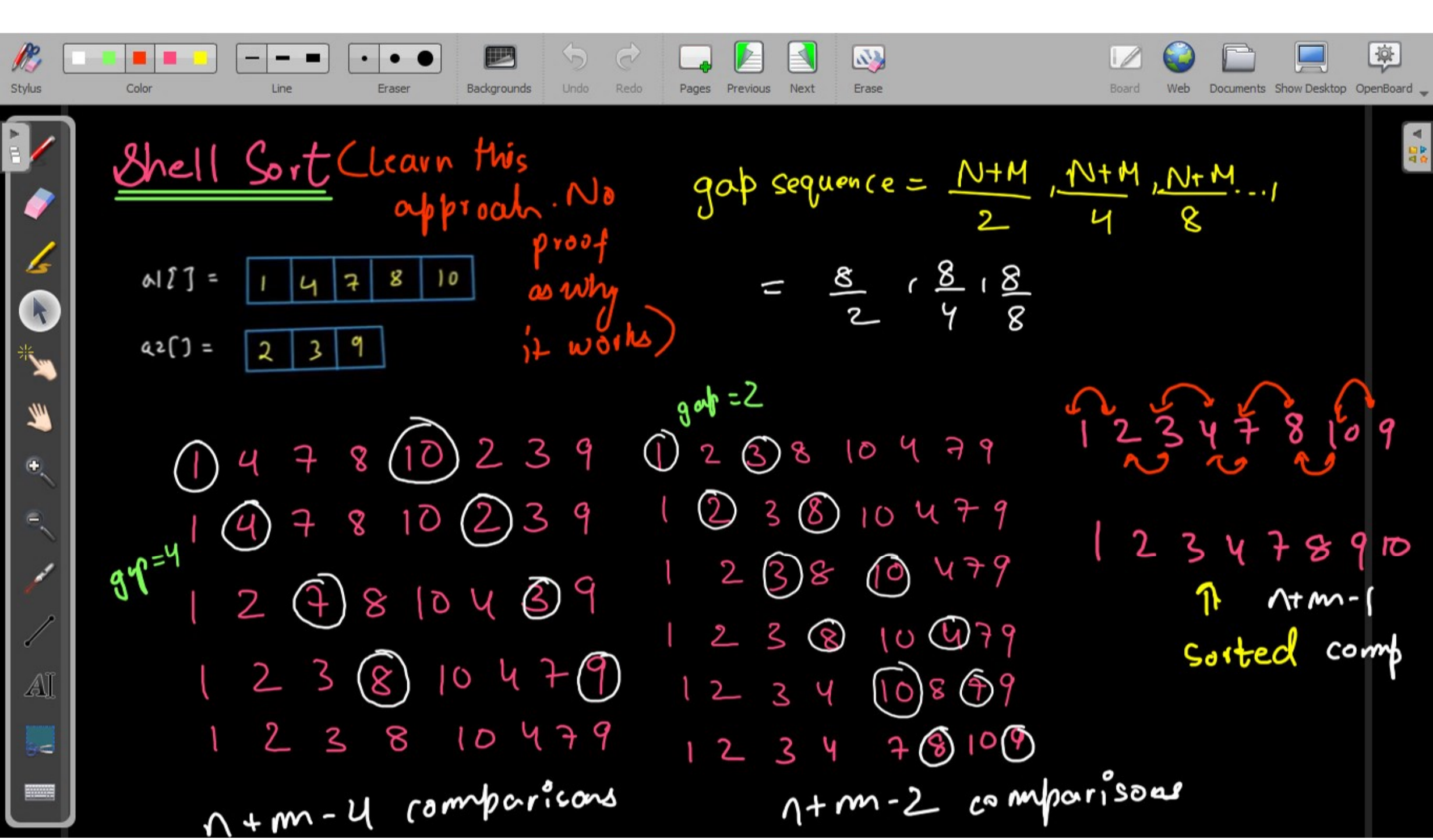
```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int[] c = new int[n + m];

        int i = 0;
        int j = 0;
        int k = 0;

        while(i < m && j < n) {
            if(nums1[i] < nums2[j]) {
                c[k++] = nums1[i++];
            } else {
                c[k++] = nums2[j++];
            }
        }

        while(i < m) {
            c[k++] = nums1[i++];
        }

        while(j < nums2.length) {
            c[k++] = nums2[j++];
        }

        for(int x=0;x<c.length;x++) {
            nums1[x] = c[x];
        }
```


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
Without Extra Space (Insertion Sort) { TC worse but space const.)
a1[] = 1 4 7 8 10
a2[] = 2 3 9
Redundant
TC = N + (N+1) + (N+2) + ... (N+M)
TC = N*M + (M(M+1))/2
TC = O(N*M + M^2)
1 2 3 4 7 8 9 10
1 4 7 8 10 2 3 9
1 4 7 8 10 2 3 9
1 4 7 8 10 2 3 9
1 4 7 8 10 2 3 9
1 4 7 8 10 2 3 9 O(N)
1 2 4 7 8 10 3 9 O(N+1)
1 2 3 4 7 8 10 9 O(N+2)

Stylus Color Line Eraser Backgrounds Undo Redo Pages Previous Next Erase Board Web Documents Show Desktop OpenBoard

Code for Insertion Sort Approach (constant space)

```java
class Solution {

    private void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public void merge(int[] nums1, int m, int[] nums2, int n) {
        for(int i=m;i<m+n;i++) {
            nums1[i] = nums2[i - m];
        }

        for(int i=m;i<m+n;i++) {
            for(int j=i;j>0;j--) {
                if(nums1[j] < nums1[j-1]) {
                    swap(nums1,j,j-1);
                } else {
                    break;
                }
            }
        }
    }
}
```


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
Shell Sort (learn this approach. No proof as why it works)
a1[] = 1 4 7 8 10
a2[] = 2 3 9
gap sequence = N+M/2 , N+M/4 , N+M/8 ...1
= 8/2 , 8/4 , 8/8
gap=4
1 4 7 8 10 2 3 9
1 4 7 8 10 2 3 9
1 2 7 8 10 4 3 9
1 2 3 8 10 4 7 9
1 2 3 8 10 4 7 9
n+m-4 comparisons
gap=2
1 2 3 8 10 4 7 9
1 2 3 8 10 4 7 9
1 2 3 8 10 4 7 9
1 2 3 8 10 4 7 9
1 2 3 4 10 8 7 9
1 2 3 4 7 8 10 9
n+m-2 comparisons
1 2 3 4 7 8 10 9
1 2 3 4 7 8 9 10
n+m-1
Sorted comp

Code using Shell Sort

```
    public void swap(int[] nums, int i, int j) {
        nums[i] = nums[i] ^ nums[j];
        nums[j] = nums[i] ^ nums[j];
        nums[i] = nums[i] ^ nums[j];
    }

    public void merge(int[] nums1, int m, int[] nums2, int n) {
        for(int i=m;i<m+n;i++) {
            nums1[i] = nums2[i-m];
        }

        for(int gap = (m + n + 1)/2;gap >= 1;gap = gap / 2) {
            for(int i=0,j=i+gap;j<m+n;i++,j++) {
                if(nums1[i] > nums1[j]) {
                    swap(nums1,i,j);
                }
            }
        }

        for(int i=0,j=1;j<m+n;i++,j++) {
            if(nums1[i] > nums1[j]) {
                swap(nums1,i,j);
            }
        }
    }
```

swap using XOR

a = a^b
b = a^b
a = a^b

More proper code

```
public void merge(int[] nums1, int m, int[] nums2, int n) {
    for(int i = m; i < m+n; i++){
        nums1[i] = nums2[i - m];
    }

    for(double gap = (m + n) / 2.0; gap > 0; gap = gap/2.0){
        int jStart = (int)Math.ceil(gap);
        for(int i=0, j=jStart; j<m+n; i++, j++){
            if(nums1[i] > nums1[j]){
                swap(nums1, i, j);
            }
        }
        if(jStart == 1) break;
    }
}
```

→ 1 Gap k liye chalana padega roz ho sakta hai end me gap 1 na a ya ha ho.

Stylus Color Line Eraser Backgrounds Undo Redo Pages Previous Next Erase Board Web Documents Show Desktop OpenBoard

```java
public void merge(int[] nums1, int m, int[] nums2, int n) {
    for(int i = m; i < m+n; i++){
        nums1[i] = nums2[i - m];
    }

    for(double gap = (m + n) / 2.0; gap > 0; gap = gap/2.0){
        int jStart = (int)Math.ceil(gap);
        for(int i=0, j=jStart; j<m+n; i++, j++){
            if(nums1[i] > nums1[j]){
                swap(nums1, i, j);
            }
        }
        if(jStart == 1) break;
    }
}
```

$gap = \frac{N}{2}$ $N - \frac{N}{2}$

$\frac{N}{4}$ $N - \frac{N}{4}$

$\frac{N}{8}$ $N - \frac{N}{8}$

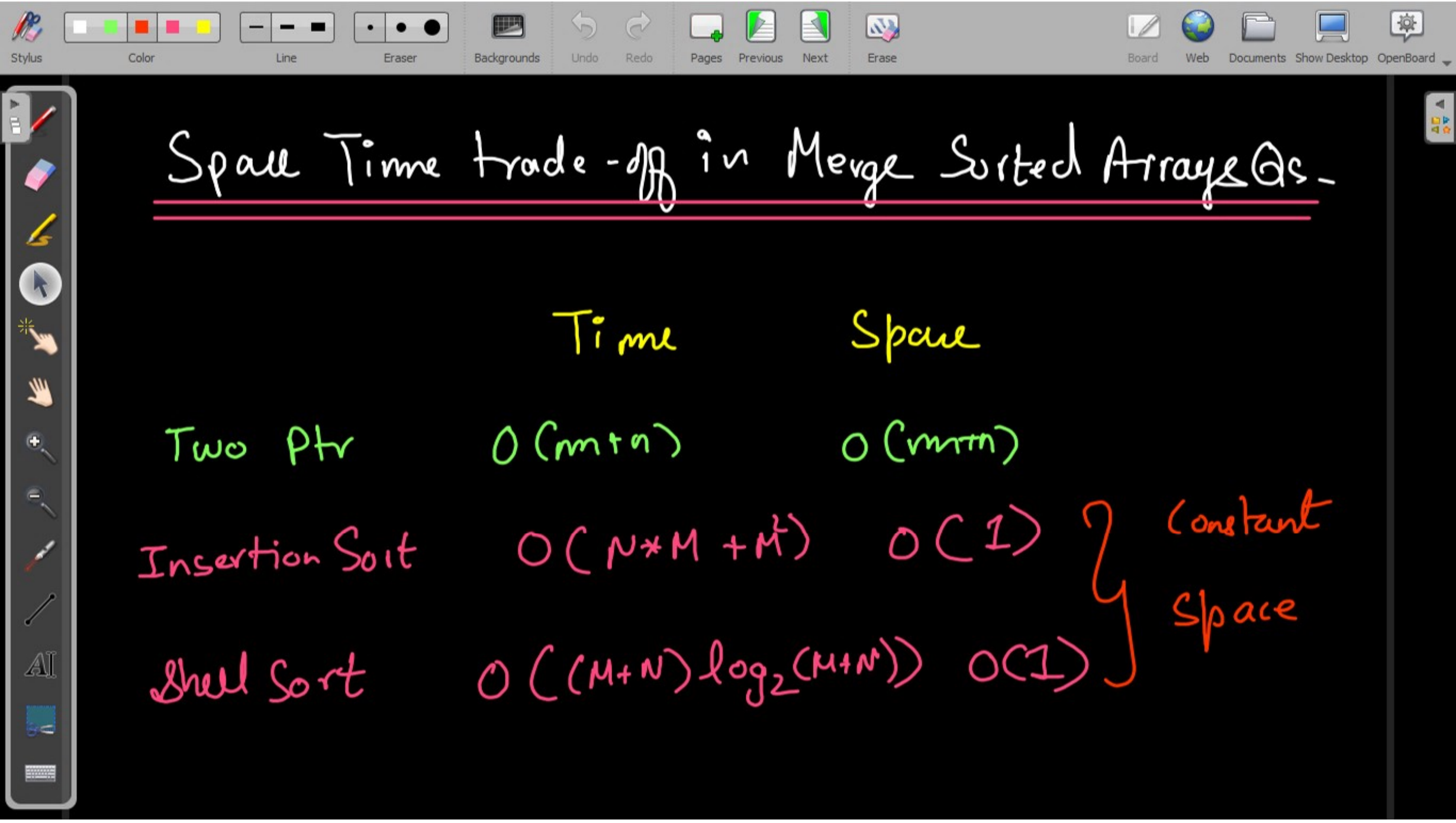
$\vdots$

1 $N-1$

$TC = O(N\log_2 N - N\{\frac{1}{2} + \frac{1}{4} \cdots\})$

$Tc = O(N\log_2 N - 2N)$

$\approx TC = O(N\log_2 N)$


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
Space Time trade-off in Merge Sorted Arrays Qs.
Time
Space
Two Ptr
O(m+n)
O(m+n)
Insertion Sort
O(N*M + M²)
O(1)
Shell Sort
O((M+N) log₂(M+N))
O(1)
Constant Space

Merge Sort (with extra space) {In Place} & Stable}

mergeSort (arr, 0, 7)

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

mid

2 3 4 8 | 1 5 6 7

merge ⇒ Merge technique should be O(1) space technique.

1 2 3 4 5 6 7 8

Code for MergeSort (In-Place) with Extra Space-

```java
public void mergeSort(int[] nums,int lo, int hi) {

    if(lo >= hi) {
        return;
    }

    int mid = (lo + hi) /2;
    //sort the left part
    mergeSort(nums,lo,mid);
    //sort the right part
    mergeSort(nums,mid + 1, hi);

    //merge the sorted parts in place
    merge(nums,lo,mid,mid+1,hi);
}
```

```java
public int[] sortArray(int[] nums) {
    mergeSort(nums,0,nums.length-1);
    return nums;
}
```


```java
public int[] sortArray(int[] nums) {
    mergeSort(nums,0,nums.length-1);
    return nums;
}
```

Time complexity

$$T(n) = T(n/2) + T(n/2) + O\underbrace{\left(\frac{N}{2} + \frac{N}{2}\right)}_{\text{merging}}$$

$x = \log_2 N$

$$T(n) = 2T(n/2) + O\left(\frac{N}{2} + \frac{N}{2}\right)$$

$$2T(n/2) = 2^2 T(n/4) + 2O\left(\frac{N}{4} + \frac{N}{4}\right)$$

$$2^2 T(n/4) = 2^3 T(n/8) + 2^2 O\left(\frac{N}{8} + \frac{N}{8}\right)$$

$$\vdots$$

$$2^x T(1) = 2T(0) + O(1)$$

$$T(n) = O(N) + 2 * O\left(\frac{N}{2}\right) + 2^2 * O\left(\frac{N}{4}\right) + \cdots 2^x * O(1)$$

$O(N * \log_2 N)$

Stylus Color Line Eraser Backgrounds Undo Redo Pages Previous Next Erase Board Web Documents Show Desktop OpenBoard

Time Complexity without Extra Space (Shell sort)

$O(N \log_2^2 N)$

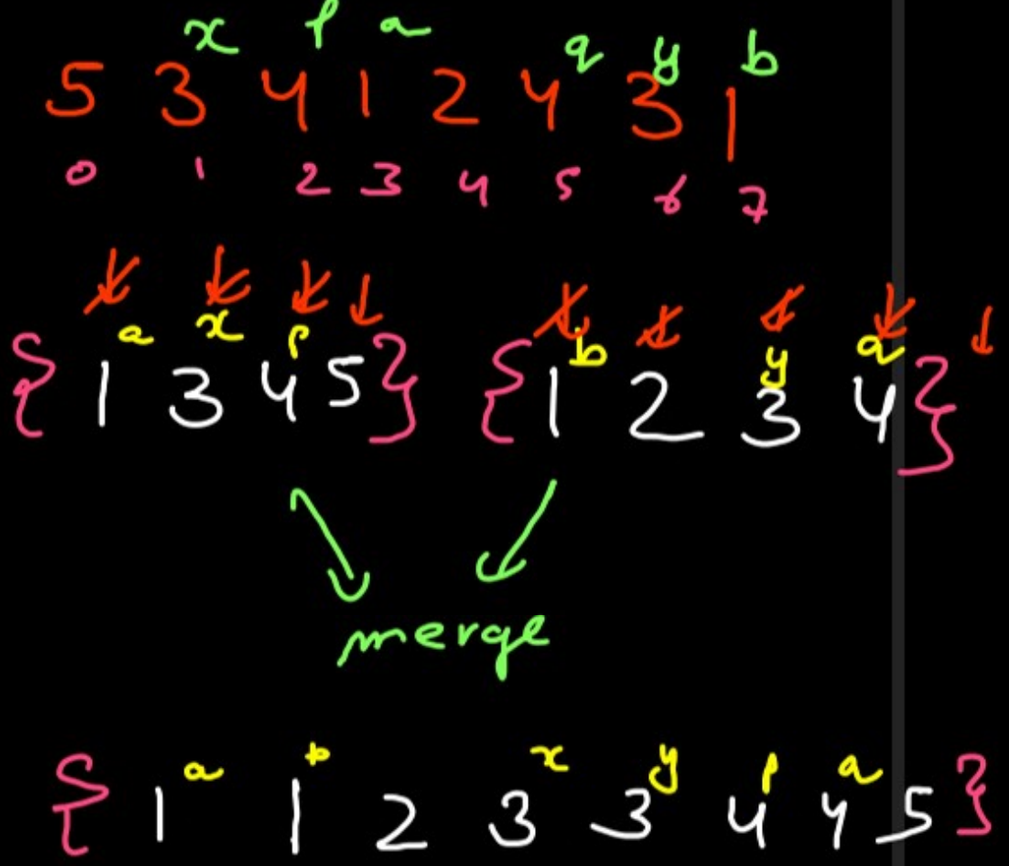
#RR

$$T(n) = 2T(n/2) + O(N \log_2 N)$$

Merge Sort is Stable

```java
public void merge(int[] nums, int a1l, int a1h, int a2l, int a2h) {
    int size = (a1h-a1l+1) + (a2h-a2l+1);
    int[] res = new int[size];
    int i=a1l,j=a2l,k=0;

    while(i <= a1h && j<= a2h) {
        if(nums[i] <= nums[j]) {
            res[k++] = nums[i++];
        } else {
            res[k++] = nums[j++];
        }
    }

    while(i<= a1h) {
        res[k++] = nums[i++];
    }

    while(j<= a2h) {
        res[k++] = nums[j++];
    }

    for(int x=a1l;x<=a2h;x++) {
        nums[x] = res[x-a1l];
    }
}
```

Stable coz of this line.



Quick Sort { Partition an Array }

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

l r

4 2 1 5 9 6 5 9

3 4

pi = left - 1

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

<= pivot > pivot

0 to l-1 → ≤ pivot

l to r-1 → > pivot

r to N → unexplored

```
if(arr[right] > pivot)
    right++
else {
    swap(arr[left], arr[right])
    left++;
    right++;
}
```

variations

→ -ve, +ve

→ 0, 1

→ 3ven, non-3ven

→ odd, even

→ <=pivot, >pivot

DNf Sort / Sort 012/ Three Way Partitioning

// travel using j pointer

```
if (arr[j] == 0)
{ swap(i,j);
  i++; j++;
}
else if(arr[j] == 1)
{
  j++;
}
else {
  swap(j,k);
  k--;
}
```

Inversion Count {Count Inversion}

count the No of pairs (i,j) such that i<j and a[i] > a[j].

Brute force : $O(N^2)$

(8,5) (8,3) (8,4) (8,1) (8,6) (8,2)

(5,3) (5,4) (5,1) (5,2)

(3,1) (3,2)

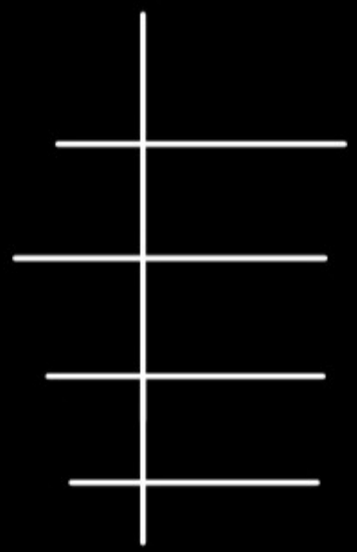
(4,1) (4,2)

(6,2)

Ans = 15

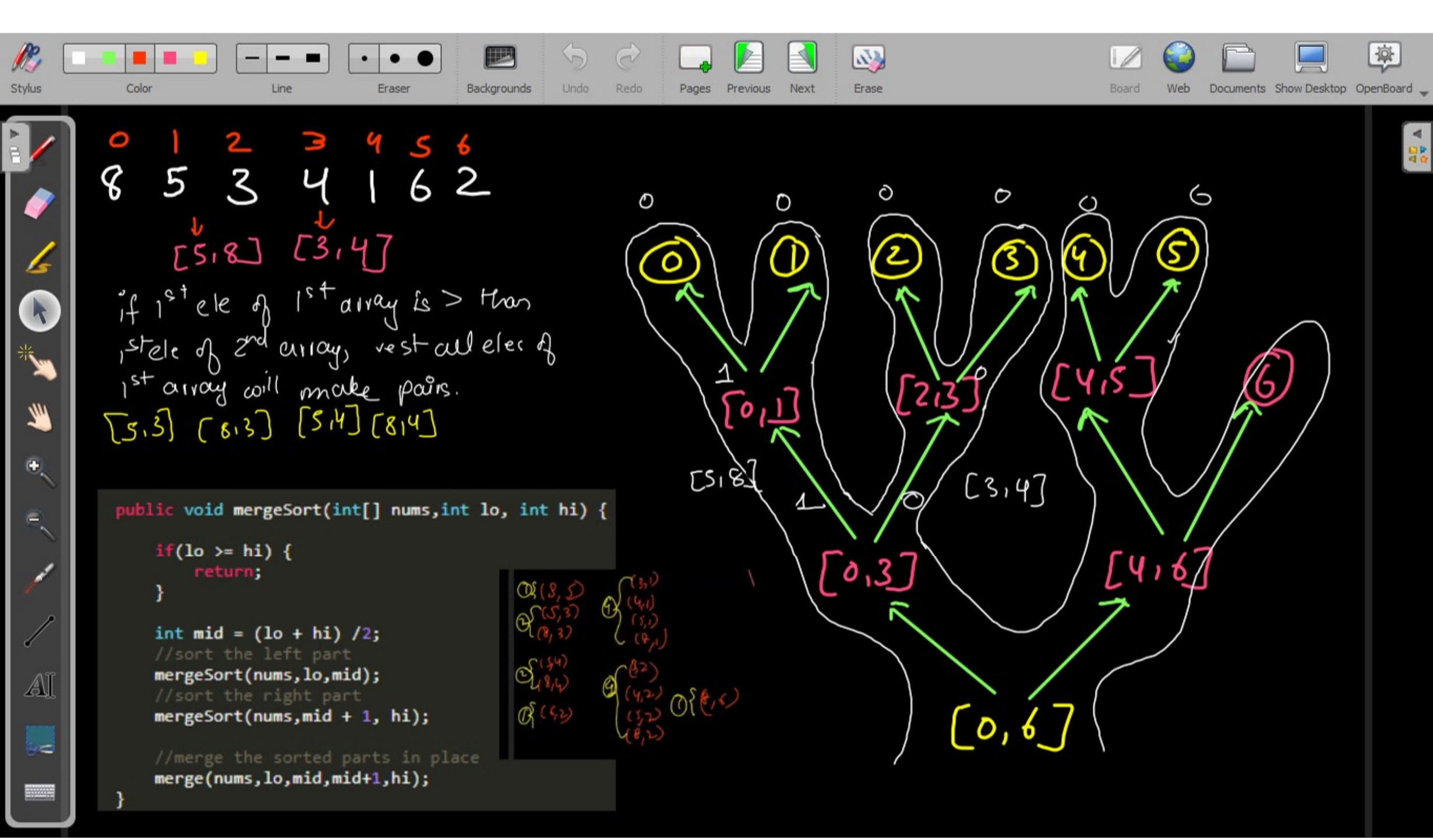
The Qs is to return the count only.

There are various optimizations possible for this DS.



- Merge Sort ✓
- fenwick Tree
- Policy Based DS (ordered set + Indexing)
- Self Balancing BST (Red Black Tree)

We are studying using MS.


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
0 1 2 3 4 5 6
8 5 3 4 1 6 2
[5,8] [3,4]
if 1st ele of 1st array is > than 1st ele of 2nd array, rest all elec of 1st array will make pairs.
[5,3] [8,3] [5,4] [8,4]
[0,1] [2,3] [4,5] 6
[5,8] [3,4]
[0,3] [4,6]
[0,6]
public void mergeSort(int[] nums,int lo, int hi) {
if(lo >= hi) {
return;
}
int mid = (lo + hi) /2;
//sort the left part
mergeSort(nums,lo,mid);
//sort the right part
mergeSort(nums,mid + 1, hi);
//merge the sorted parts in place
merge(nums,lo,mid,mid+1,hi);
}

Code for Inversion Count

```java
public static long mergeSort(long[] nums,int lo, int hi) {

    if(lo >= hi) {
        return 0l;
    }

    int mid = (lo + hi) /2;
    //sort the left part
    long left = mergeSort(nums,lo,mid);
    //sort the right part
    long right = mergeSort(nums,mid + 1, hi);

    //merge the sorted parts in place
    long ans = merge(nums,lo,mid,mid+1,hi);

    return left + right + ans;
}
```

```java
static long inversionCount(long arr[], long N)
{
    return mergeSort(arr,0,(int)N-1);
}
```

```java
public static long merge(long[] nums, int a1l, int a1h, int a2l, int a2h) {
    int size = (int)(a1h-a1l+1l) + (int)(a2h-a2l+1l);
    long[] res = new long[size];
    int i=a1l,j=a2l,k=0;

    long invCount = 0l;
    while(i <= a1h && j<= a2h) {
        if(nums[i] <= nums[j]) {
            res[k++] = nums[i++];
        } else {
            res[k++] = nums[j++];
            invCount += (a1h - i + 1l);
        }
    }

    while(i<= a1h) {
        res[k++] = nums[i++];
    }

    while(j<= a2h) {
        res[k++] = nums[j++];
    }

    for(int x=a1l;x<=a2h;x++) {
        nums[x] = res[x-a1l];
    }

    return invCount;
}
```

→ 1st array k Saare rem element.

Reverse Pairs (leetcode Hard : 493)

$i < j, \quad a[i] > 2 * a[j]$

0	1	2	3	4	5	6
40	25	19	12	9	6	2

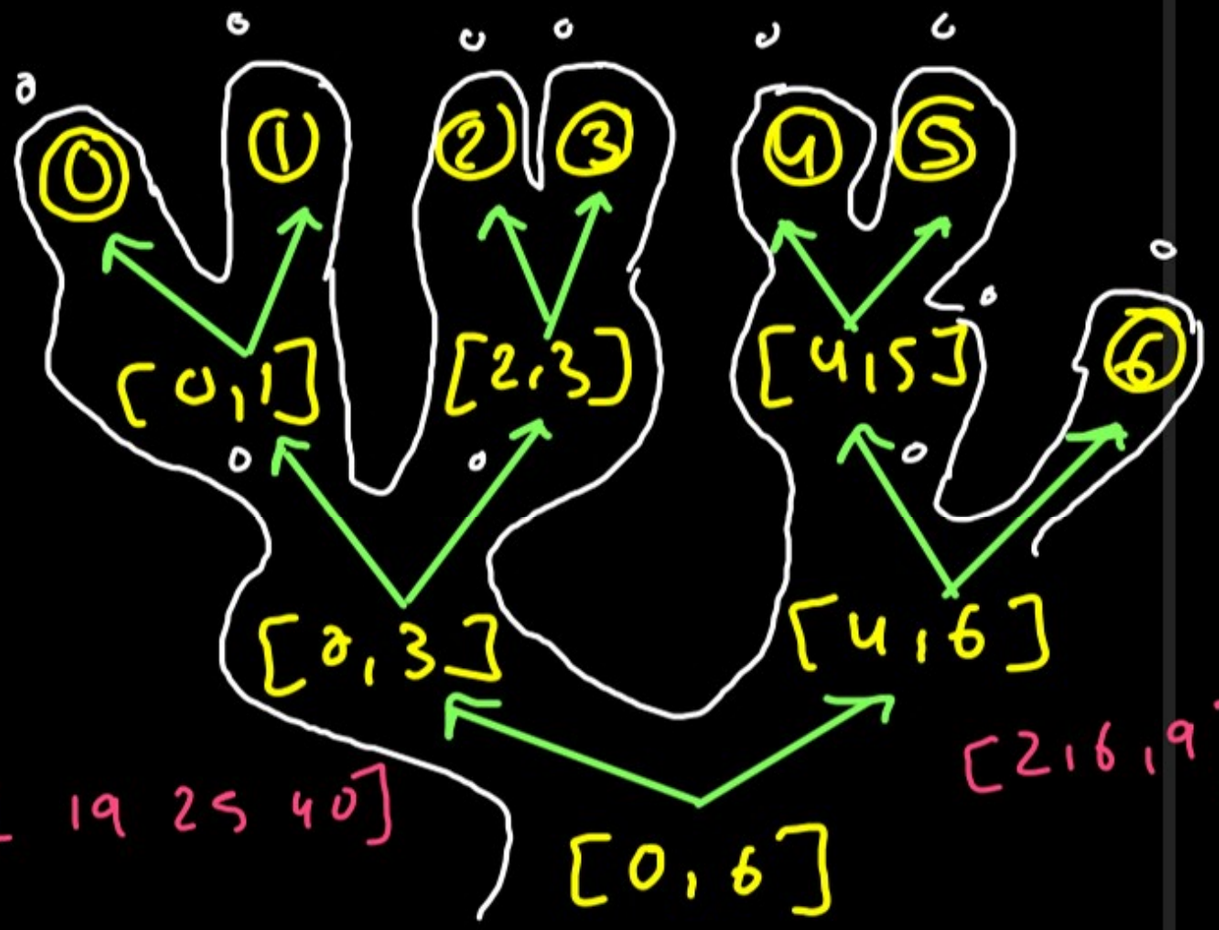
[12,19,25,40] [2,6,9]

{4, 12, 18}

[12,2] [19,2] [25,2] [40,2]

[19,6] [25,6] [40,6]

[19,9] [25,9] [40,9]



Why should we merge & count the reverse pairs seperately?

⇒ If we do merging and counting simultaneously:

{25, 40} {12, 19}

[12]

+2 (25, 12) (40, 12)

{25, 40} {12, 19}

No pair added However {40, 19}

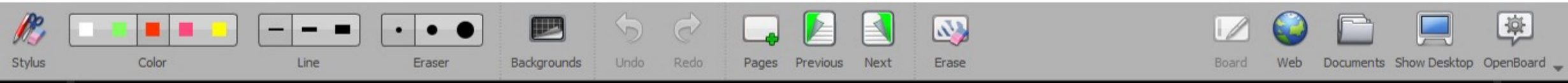
[12, 19]

Because while merging we only see a[i] > a[j] & not a[i] > 2 * a[j]

Now only merging will occur

{25, 40}

[12, 19, 25, 40] → (40, 19) pair was missed



```java
public int mergeSort(int[] nums,int lo, int hi) {

    if(lo >= hi) {
        return 0;
    }

    int mid = (lo + hi) /2;
    //sort the left part
    int left = mergeSort(nums,lo,mid);
    //sort the right part
    int right = mergeSort(nums,mid + 1, hi);

    //merge the sorted parts in place
    int ans = merge(nums,lo,mid,mid+1,hi);

     return left + right + ans;
}
```

```java
public int findInversionCount(int[] nums, int a1l, int a1h, int a2l, int a2h) {
    int invCount = 0;
    int i=a1l,j=a2l,k=0;
    while(i <= a1h && j<= a2h) {
        if((long)nums[i] <= 2l* (long)nums[j]) {
            k++;i++;
        } else {
            invCount += (a1h - i + 1);
            k++;j++;
        }
    }
    return invCount;
}
```

```java
public int merge(int[] nums, int a1l, int a1h, int a2l, int a2h) {
    int size = (a1h-a1l+1) + (a2h-a2l+1);
    int[] res = new int[size];

    int invCount = findInversionCount(nums,a1l,a1h,a2l,a2h);

    int i=a1l,j=a2l,k=0;

    while(i <= a1h && j<= a2h) {
        if(nums[i] <= nums[j]) {
            res[k++] = nums[i++];
        } else {
            res[k++] = nums[j++];
        }
    }

    while(i<= a1h) {
        res[k++] = nums[i++];
    }

    while(j<= a2h) {
        res[k++] = nums[j++];
    }

    for(int x=a1l;x<=a2h;x++) {
        nums[x] = res[x-a1l];
    }

    return invCount;
}
```

```java
public int reversePairs(int[] nums) {
    return mergeSort(nums,0,nums.length - 1);
}
```

Stylus Color Line Eraser Backgrounds Undo Redo Pages Previous Next Erase Board Web Documents Show Desktop OpenBoard

Count Winning Streaks (NADOS)

{LC 2031 count subarrays with more 1s than 0s} {Premium}

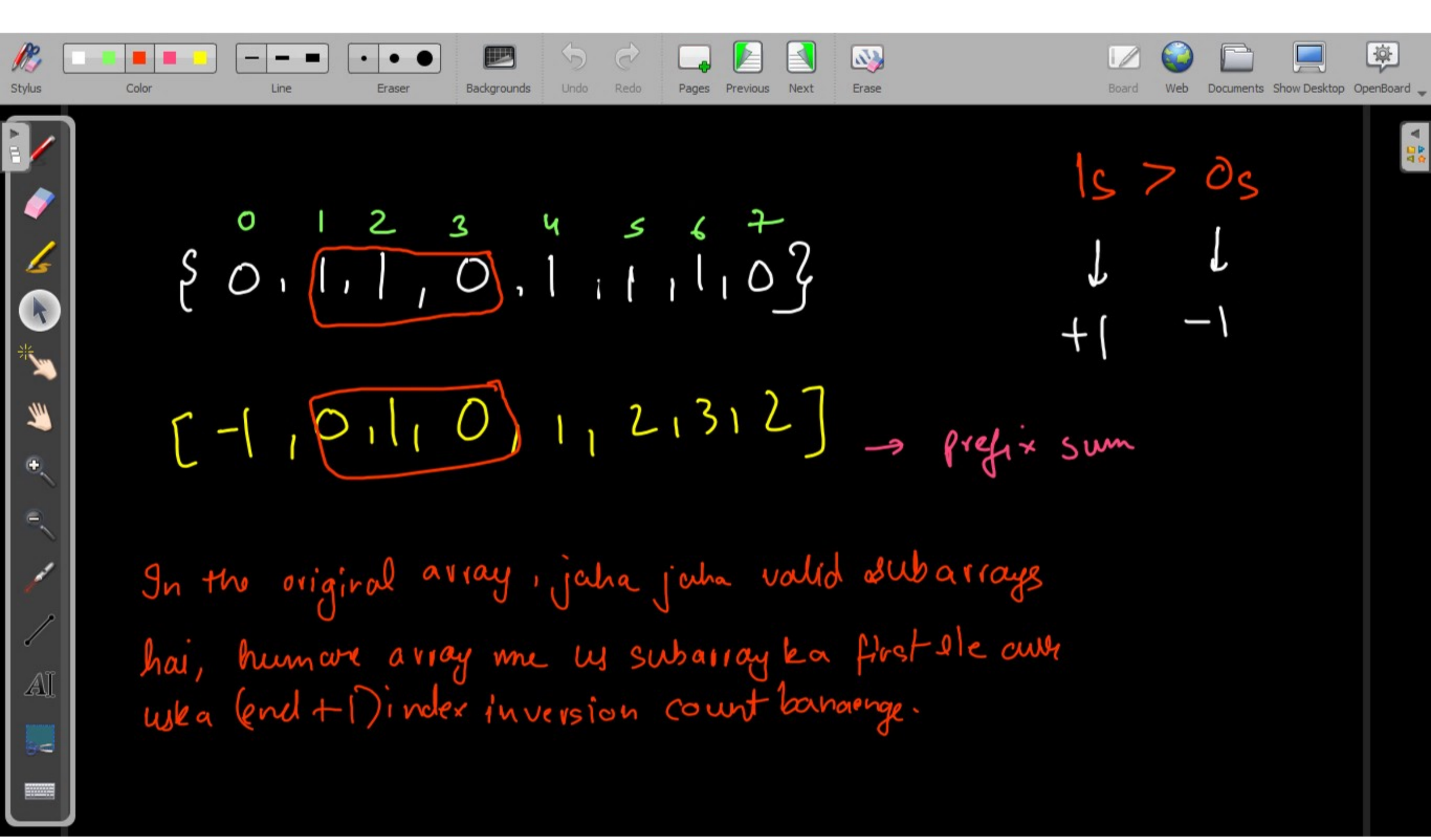
0 1 2 3 4 5 6 7

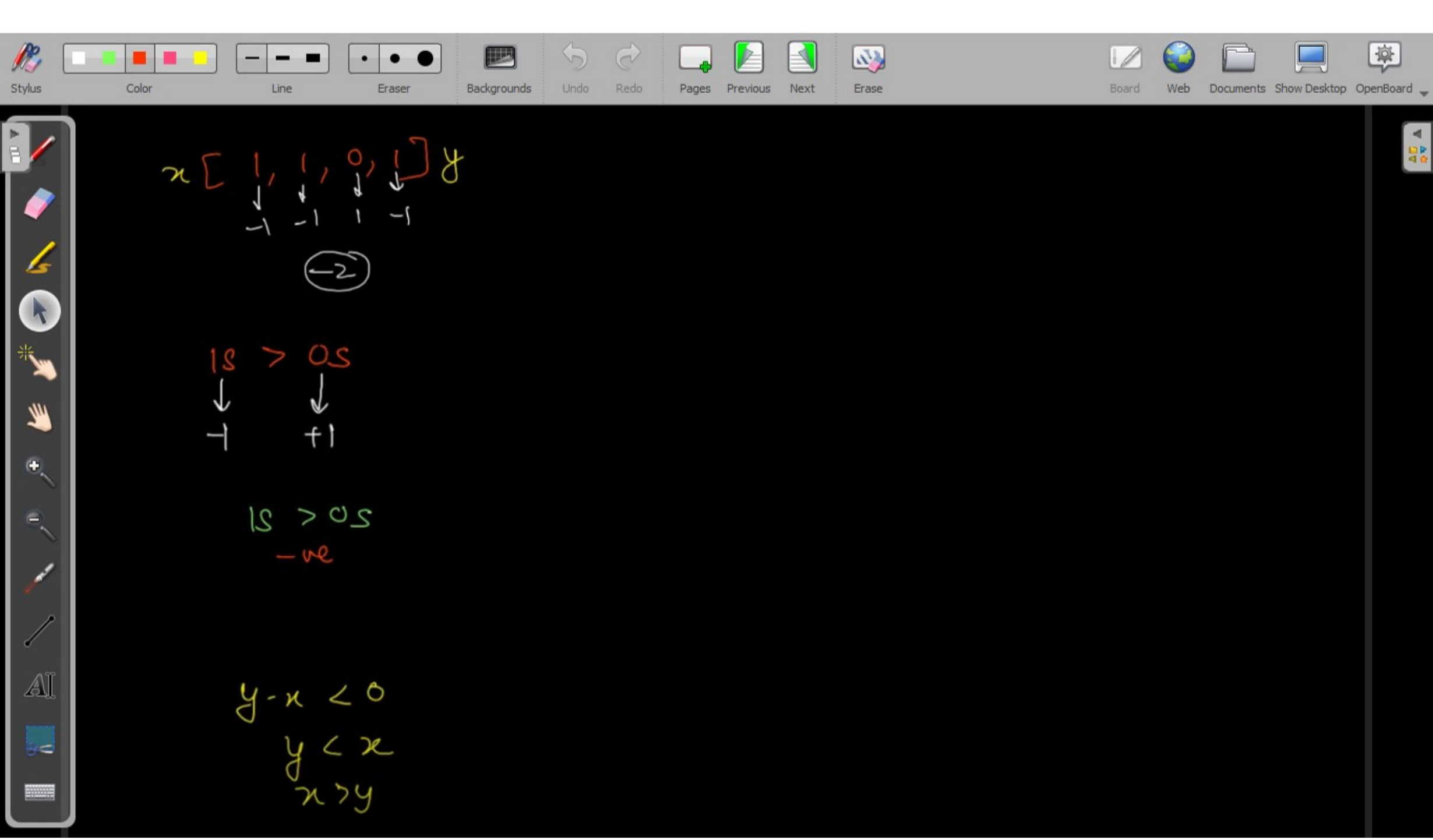
{0, 1, 1, 0, 1, 0, 1, 0}

```java
public static int winningStreak(int[] arr){
    int ans = 0;

    for(int st=0;st<arr.length;st++) {
        int countOf0s = 0, countOf1s = 0;
        for(int end=st;end < arr.length;end++) {
            if(arr[end] == 0) {
                countOf0s++;
            } else {
                countOf1s++;
            }

            if(countOf1s > countOf0s) {
                ans++;
            }
        }
    }

    return ans;
}
```

$O(N^2)$ approach

Gives TLE


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
0 1 2 3 4 5 6 7
{0, 1, 1, 0, 1, 1, 1, 0}
1s > 0s
+1
-1
[-1, 0, 1, 0, 1, 2, 3, 2]
→ prefix sum
In the original array, jaha jaha valid subarrays
hai, humare array me us subarray ka first ele
uska (end+1) index inversion count banaenge.


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
x [1, 1, 0, 1] y
-1 -1 1 -1
-2
1s > 0s
-1
+1
1s > 0s
-ve
y - x < 0
y < x
x > y

Minimum Swaps to Sort (Swapping only adjacent eles)

(MinSwaps II)

Minimum Swaps to Sort

Medium Accuracy: 50.0% Submissions: 89710 Points: 4

Given an array of n distinct elements. Find the minimum number of swaps required to sort the array in strictly increasing order.

{ 8, 5, 3, 4, 1, 6, 2 }

{ 1, 2, 3, 4, 5, 6, 8 }

So, if I want to take 8 to the last pos, I need to swap 8 with all the elements that are greater than 8 and are to its right. (Inversion count for 8)

{8, 5, 3, 4, 1, 6, 2}

↓ After 8 reaches its correct position

{(5, 3, 4, 1, 6, 2) 8}

↓

Ans for 5 is in remaining array. This means that the ans for 5 does not concern 8. This implies (in original array) that $i < j$ { for $a[i] > a[j]$ }.

bubble sort-

Min adjacent swaps to Sort

≡

Inversion Count

```java
public static int merge(int[] nums, int a1l, int a1h, int a2l, int a2h) {
    int size = (int)(a1h-a1l+1l) + (int)(a2h-a2l+1l);
    int[] res = new int[size];
    int i=a1l,j=a2l,k=0;

    int invCount = 0;
    while(i <= a1h && j<= a2h) {
        if(nums[i] <= nums[j]) {
            res[k++] = nums[i++];
        } else {
            res[k++] = nums[j++];
            invCount += (a1h - i + 1l);
        }
    }

    while(i<= a1h) {
        res[k++] = nums[i++];
    }

    while(j<= a2h) {
        res[k++] = nums[j++];
    }

    for(int x=a1l;x<=a2h;x++) {
        nums[x] = res[x-a1l];
    }

    return invCount;
}
```

```java
public static int mergeSort(int[] nums,int lo, int hi) {

    if(lo >= hi) {
        return 0;
    }

    int mid = (lo + hi) /2;
    //sort the left part
    int left = mergeSort(nums,lo,mid);
    //sort the right part
    int right = mergeSort(nums,mid + 1, hi);

    //merge the sorted parts in place
    int ans = merge(nums,lo,mid,mid+1,hi);

    return left + right + ans;
}
```

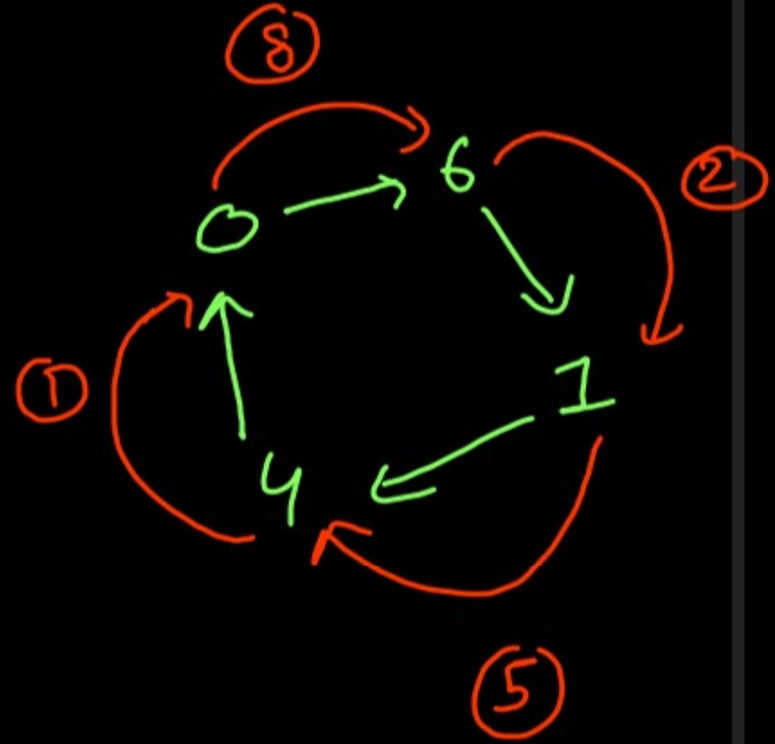
```java
int countSwaps(int nums[], int n) {
    // code here
    return mergeSort(nums,0,n-1);
}
```

Minimum Swaps to Sort -I

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

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

	src, dest
8	(0, 6)
5	(1, 4)
3	(2, 2)
4	(3, 3)
1	(4, 0)
6	(5, 5)
2	(6, 1)

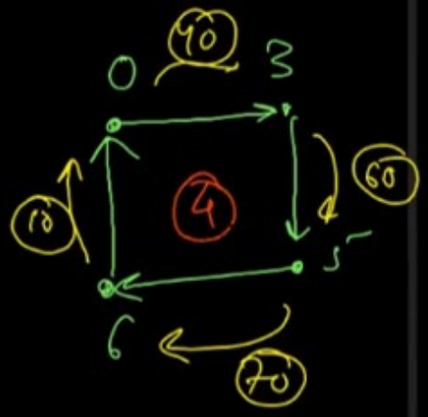


Min Swaps = 4

Stylus Color Line Eraser Backgrounds Undo Redo Pages Previous Next Erase Board Web Documents Show Desktop OpenBoard

Multiple Cycles are also Possible. (Add all cycles length for ans)

40	50	30	60	20	70	10	80
0	1	2	3	4	5	6	7
10	20	30	40	50	60	70	80



One node cannot be a part of 2 cycles as one node will always have only one incoming & one outgoing edge. (Except self loops)

```java
public static class Pair implements Comparable<Pair>{
    int ele;
    int idx;

    Pair(int ele, int idx) {
        this.ele = ele;
        this.idx = idx;
    }

    public int compareTo(Pair other) {
        return this.ele - other.ele;
    }
}

public int cycleSize(List<Pair> list, int start, boolean[] vis) {
    int res = 0;
    int j = start;

    while(vis[j] == false) {
        vis[j] = true;
        j = list.get(j).idx;
        res++;
    }

    return res - 1;
}
```

```java
    public int minSwaps(int nums[])
    {
        int n = nums.length;
        List<Pair> list = new ArrayList<>();

        for(int i=0;i<n;i++) {
            Pair p = new Pair(nums[i],i);
            list.add(p);
        }

        Collections.sort(list);
        boolean[] vis = new boolean[n];
        int res = 0;
        for(int i=0;i<n;i++) {
            if(vis[i] == false)
                res += cycleSize(list,i,vis);
        }

        return res;
    }
}
```

Count Sort (Without Stability)

9	6	3	5	3	4	3	9	6	4	6	5	8	2	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

① 1st iteration → find Max & Min.

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

2nd iteration

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

2 3 3 3 4 4 5 5 6 6 8 9 9 9

3rd iteration

TC = O(N)

SC = O(max - min + 1)

Bucket Sort

① Characters → Sorting a String

Lower case → 26 buckets

Uppercase → 52 buckets

All ASCII characters → 256 buckets

② Marks → (0, 100)

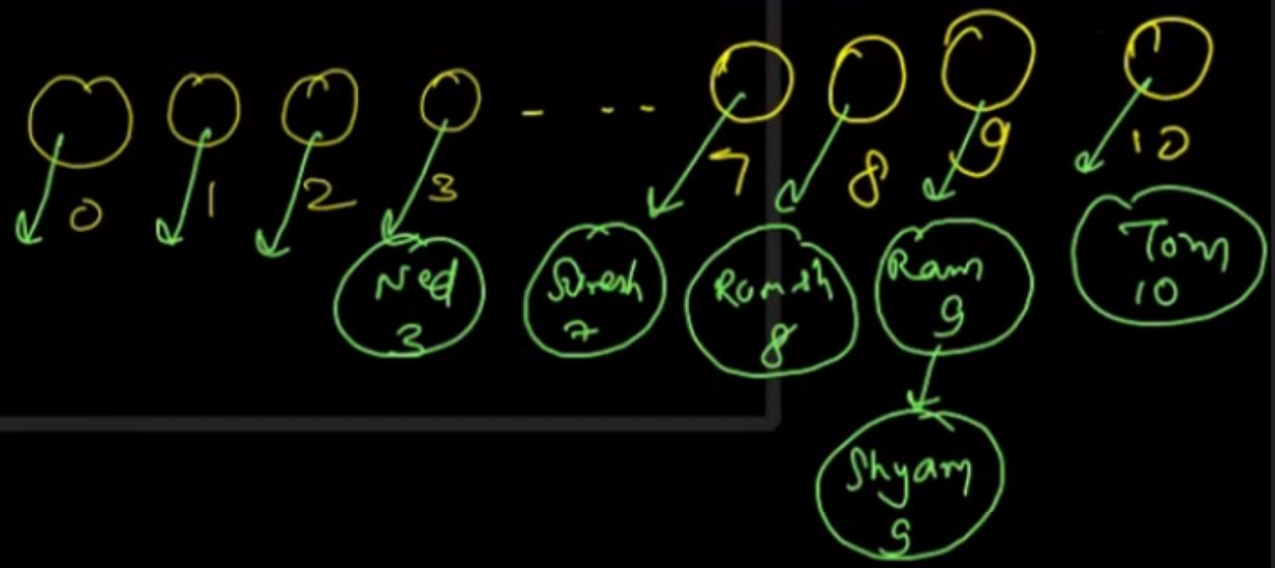
③ Values corresponding to index: Nelements & values are 1 to N or 0 to N-1

HashMap kind of chaining. This is Stable.

(2) Marks → (0, 100)

Ram → 9
Ramesh → 8
Shyam → 9
Suresh → 7

Sorting objects based

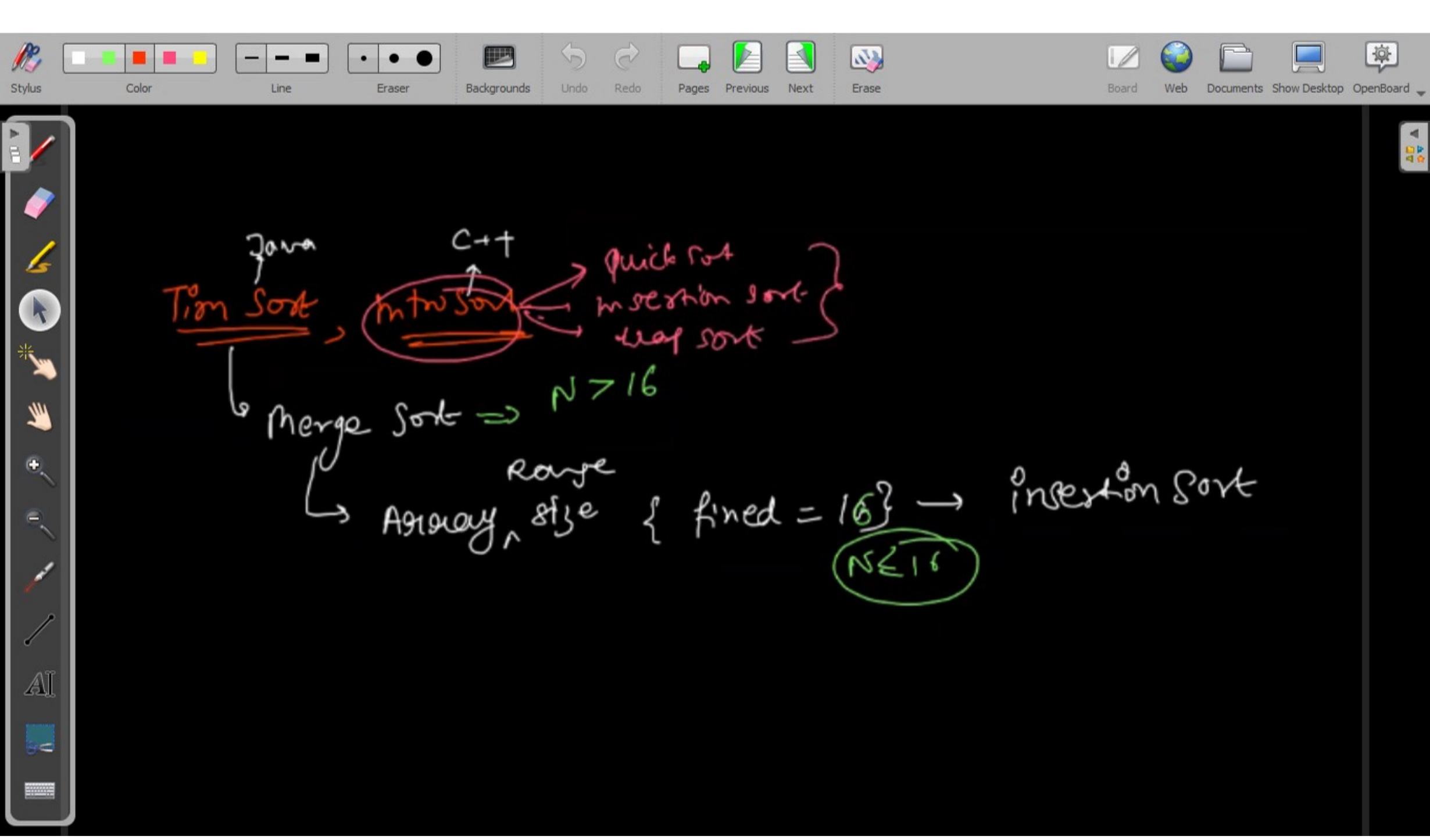


Tom → 10
Ned →

No. of buckets = no of distinct elements

Total space = No of buckets + (N) → No of elements

if N is very greater than No of buckets. SC = O(N)


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
Java
C++
Tim Sort
Intro Sort
Quick Sort
Insertion Sort
Heap Sort
Merge Sort ⇒ N > 16
Range
Array size { fixed = 16 } → Insertion Sort
N ≤ 16

Stylus Color Line Eraser Backgrounds Undo Redo Pages Previous Next Erase Board Web Documents Show Desktop OpenBoard

451. Sort Characters By Frequency

Medium 4156 181 Add to List Share

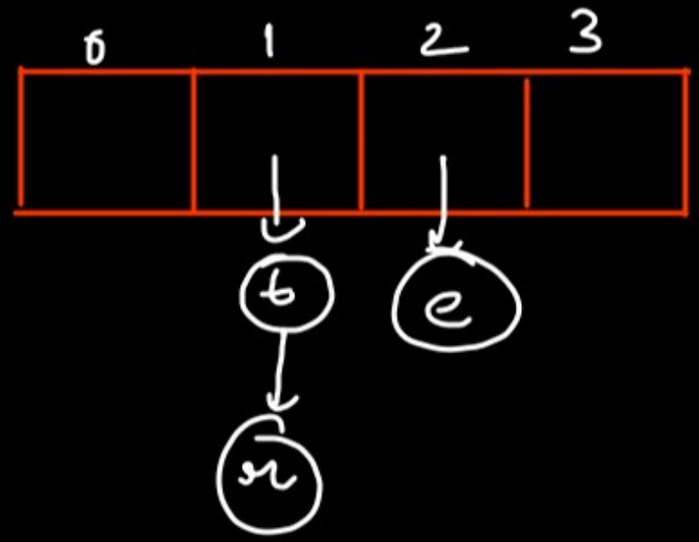
Given a string `s`, sort it in **decreasing order** based on the **frequency** of the characters. The **frequency** of a character is the number of times it appears in the string.

Return *the sorted string*. If there are multiple answers, return *any of them*.

tree

1

make an array of size s.length()+1 as max freq of a character can be s.length() & min 0.



} Now traverse each bucket But traverse this array of buckets in reverse order as we want max freq 1st.

eetr ⇒ Ans.

```java
class Solution {
    public String frequencySort(String s) {
        HashMap<Character,Integer> freq = new HashMap<>();

        for(int i=0;i<s.length();i++) {
            char ch = s.charAt(i);
            int oFreq = freq.getOrDefault(ch,0);
            freq.put(ch, oFreq + 1);
        }

        ArrayList<Character>[] buckets = new ArrayList[s.length() + 1];
        for(int i=0;i<buckets.length;i++) {
            buckets[i] = new ArrayList<>();
        }

        for(Character ch : freq.keySet()) {
            int val = freq.get(ch);
            buckets[val].add(ch);
        }

        StringBuilder sb = new StringBuilder("");
        for(int b=buckets.length-1;b>=0;b--) {
            for(Character ch : buckets[b]) {
                for(int f=0;f<b;f++) {
                    sb.append(ch);
                }
            }
        }

        return sb.toString();
    }
```

1636. Sort Array by Increasing Frequency

Easy 1400 55 Add to List Share

Given an array of integers `nums`, sort the array in **increasing** order based on the frequency of the values. If multiple values have the same frequency, sort them in **decreasing** order.

Return the *sorted array*.

Same as prev ques. Just, the buckets will point to a treeSet instead of ArrayList because of this reason.

TreeSet (Java)
Ordered set (C++)
↳ Self Balancing BST

```java
public int[] frequencySort(int[] nums) {
    HashMap<Integer,Integer> freq = new HashMap<>();
    for(int i=0;i<nums.length;i++) {
        int oFreq = freq.getOrDefault(nums[i],0);
        freq.put(nums[i],oFreq + 1);
    }

    TreeSet<Integer>[] buckets = new TreeSet[nums.length + 1];
    for(int i=0;i<buckets.length;i++) {
        buckets[i] = new TreeSet<>();
    }

    for(Integer x : freq.keySet()) {
        int freqVal = freq.get(x);
        buckets[freqVal].add(x); //x will be added in ordered manner itself
    }

    int[] res = new int[nums.length];
    int idx = 0;

    for(int b=0;b<buckets.length;b++) {
        for(Integer x : buckets[b].descendingSet()) {
            for(int f=0;f<b;f++) {
                res[idx++] = x;
            }
        }
    }

    return res;
}
```

WC TC : $O(N \log_2 N)$

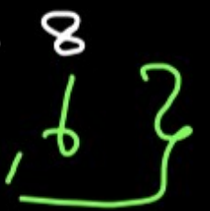
BC TC : $O(N)$

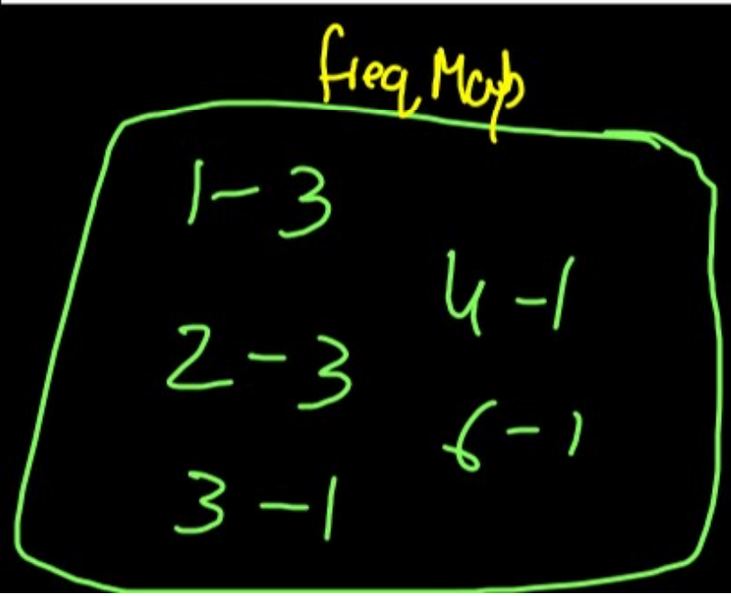
Top K Elements {LC:347}

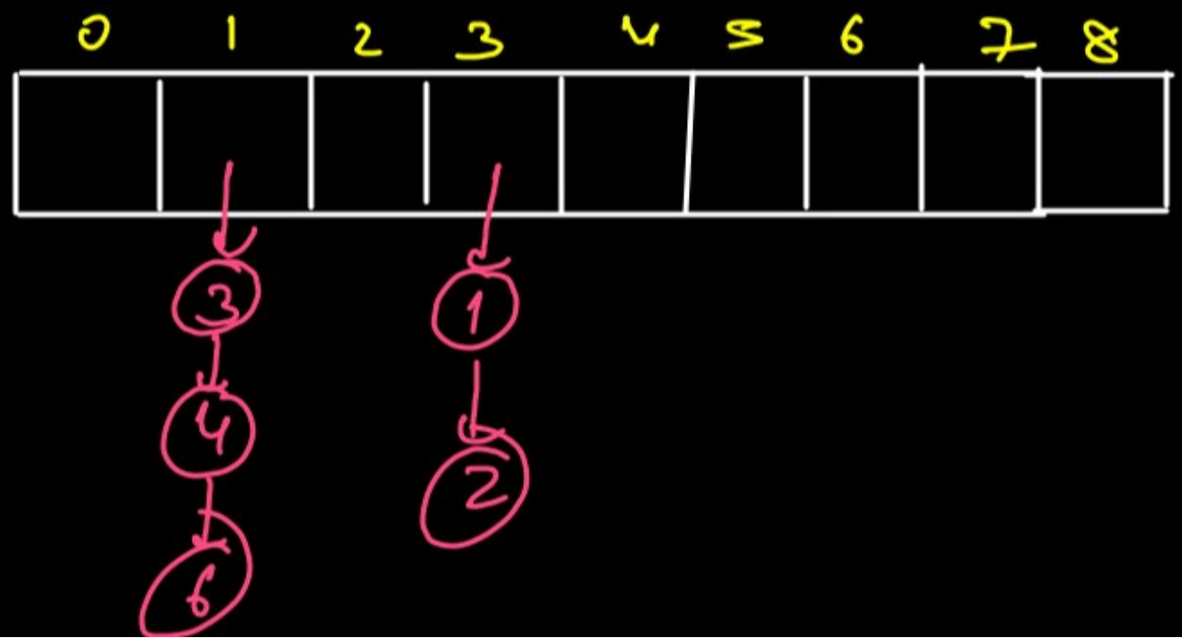
347. Top K Frequent Elements

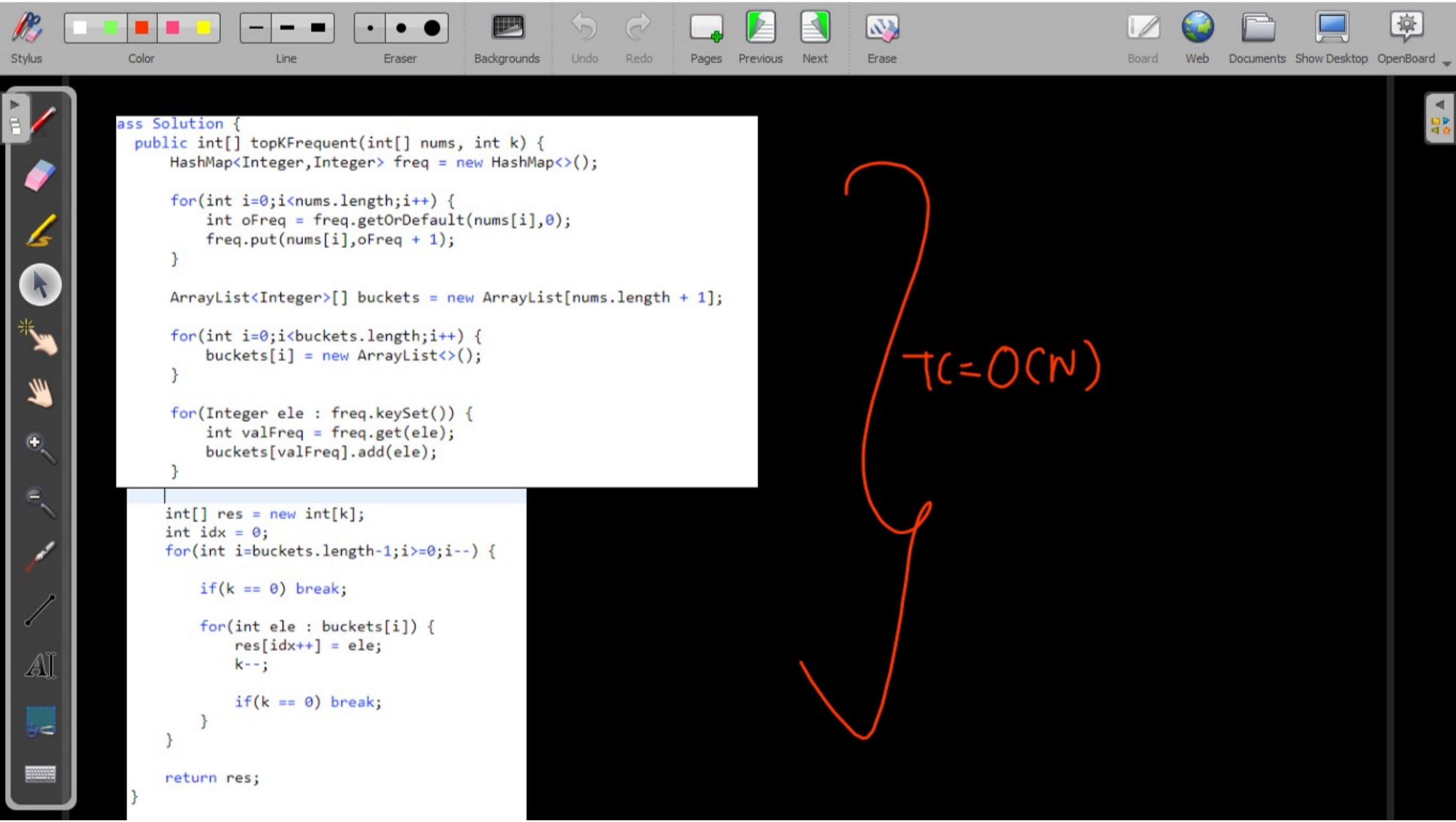
Medium 9285 374 Add to List Share

Given an integer array `nums` and an integer `k`, return *the* `k` *most frequent elements*. You may return the answer in **any order**.








Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
ass Solution {
public int[] topKFrequent(int[] nums, int k) {
HashMap<Integer,Integer> freq = new HashMap<>();
for(int i=0;i<nums.length;i++) {
int oFreq = freq.getOrDefault(nums[i],0);
freq.put(nums[i],oFreq + 1);
}
ArrayList<Integer>[] buckets = new ArrayList[nums.length + 1];
for(int i=0;i<buckets.length;i++) {
buckets[i] = new ArrayList<>();
}
for(Integer ele : freq.keySet()) {
int valFreq = freq.get(ele);
buckets[valFreq].add(ele);
}
int[] res = new int[k];
int idx = 0;
for(int i=buckets.length-1;i>=0;i--) {
if(k == 0) break;
for(int ele : buckets[i]) {
res[idx++] = ele;
k--;
if(k == 0) break;
}
}
return res;
}
TC=O(N)

Top k Words (LC: 692)

692. Top K Frequent Words

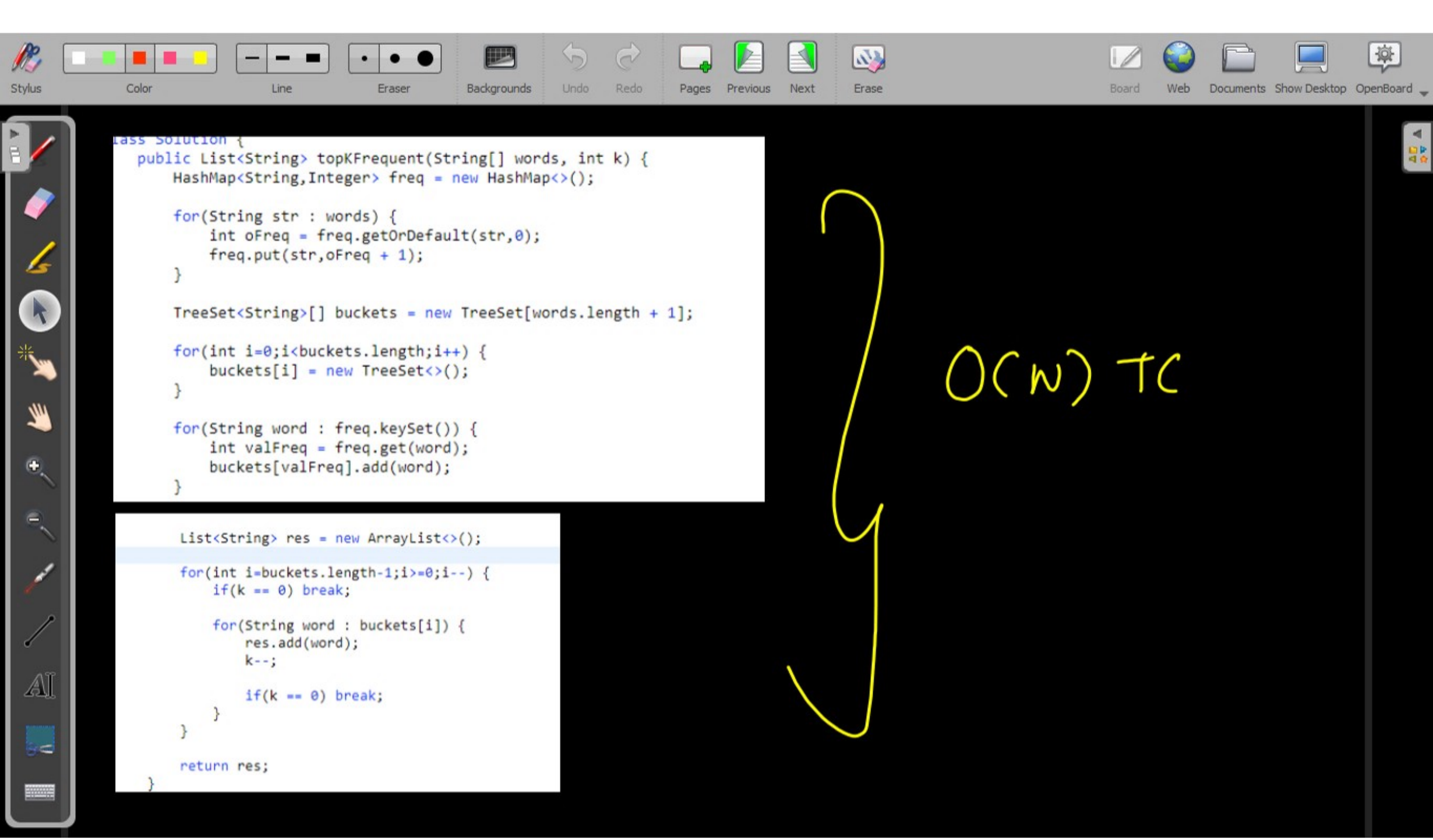
Medium 4484 252 Add to List Share

Given an array of strings `words` and an integer `k`, return *the* `k` *most frequent strings.*

Return the answer **sorted** by **the frequency** from highest to lowest. Sort the words with the same frequency by their **lexicographical order**.

Example 1:

Same as prev ques. But use Treeset instead of Arraylist to maintain lexicographical order


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
class Solution {
public List<String> topKFrequent(String[] words, int k) {
HashMap<String,Integer> freq = new HashMap<>();
for(String str : words) {
int oFreq = freq.getOrDefault(str,0);
freq.put(str,oFreq + 1);
}
TreeSet<String>[] buckets = new TreeSet[words.length + 1];
for(int i=0;i<buckets.length;i++) {
buckets[i] = new TreeSet<>();
}
for(String word : freq.keySet()) {
int valFreq = freq.get(word);
buckets[valFreq].add(word);
}
List<String> res = new ArrayList<>();
for(int i=buckets.length-1;i>=0;i--) {
if(k == 0) break;
for(String word : buckets[i]) {
res.add(word);
k--;
if(k == 0) break;
}
}
return res;
}
O(N) TC

Coordinate Compression.

Convert an array to reduced form

Medium Accuracy: 69.45% Submissions: 4206 Points: 4

Given an array with **N** distinct elements, convert the given array to a reduced form where all elements are in range from **0** to **N-1**. The order of elements is same, i.e., **0** is placed in place of smallest element, **1** is placed for second smallest element, ... **N-1** is placed for largest element.

{10, 40, 20}

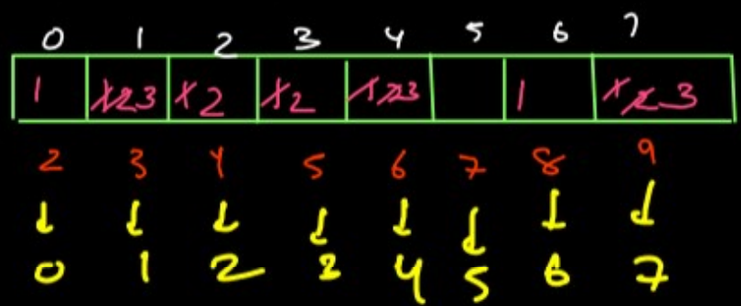
{10-0 40-1 20-2}

↓ Sort

{10-0 20-2 40-1}

0 2 1

Count sort was also coordinate compression



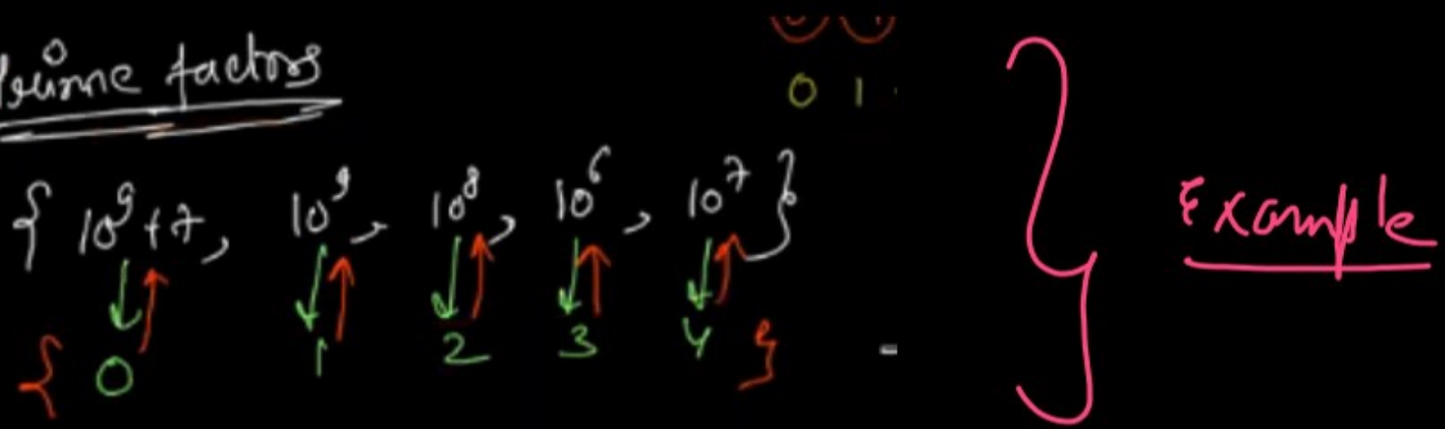
2 ko value 0

3 ko value 1

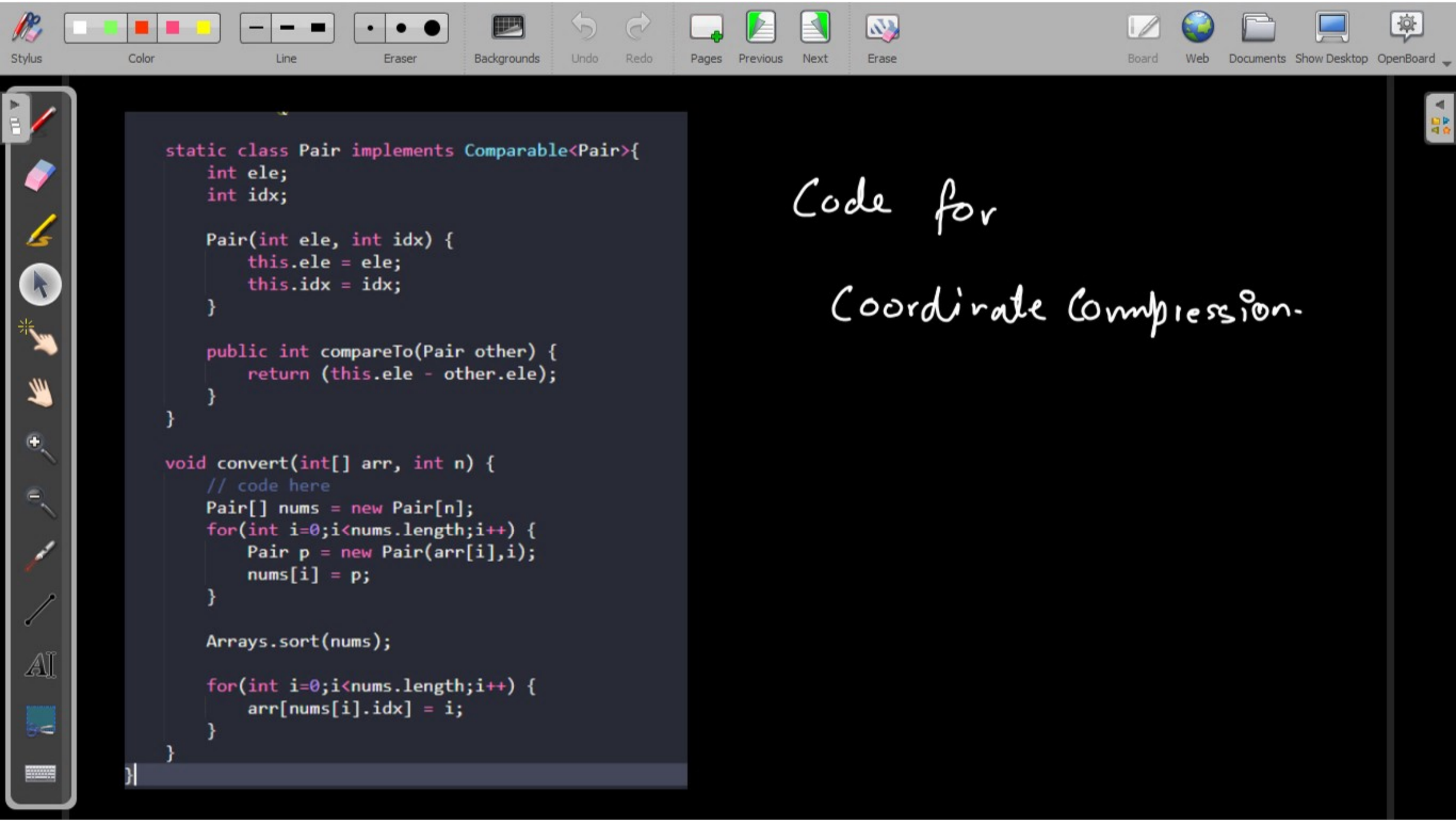
& soon

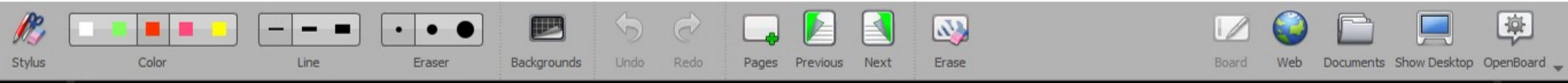
Coordinate compression can be used in many questions

* Prime factors



Example


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
static class Pair implements Comparable<Pair>{
int ele;
int idx;
Pair(int ele, int idx) {
this.ele = ele;
this.idx = idx;
}
public int compareTo(Pair other) {
return (this.ele - other.ele);
}
}
void convert(int[] arr, int n) {
// code here
Pair[] nums = new Pair[n];
for(int i=0;i<nums.length;i++) {
Pair p = new Pair(arr[i],i);
nums[i] = p;
}
Arrays.sort(nums);
for(int i=0;i<nums.length;i++) {
arr[nums[i].idx] = i;
}
}
}
Code for
Coordinate Compression.



164. Maximum Gap

Hard 2127 287 Add to List Share

Given an integer array `nums`, return *the maximum difference between two successive elements in its sorted form.* If the array contains less than two elements, return `0`.

You must write an algorithm that runs in linear time and uses linear extra space.

Radix Sort

```java
class Solution {

//     Radix Sort karna hai
//     Yaani har digit pe count sort
//     Since integer me max 10 digits honge
//     TC -> O(10 * N)
//     SC -> O(N)

    void countSort(int[] nums,int exp) {
        int n = nums.length;
        int[] output = new int[n];
        int[] count = new int[10];

        for(int i=0;i<n;i++)
            count[(nums[i]/ exp) % 10]++;

        for(int i=1;i<10;i++) {
            count[i] += count[i-1];
        }

        for(int i=n-1;i>=0;i--) {
            output[count[(nums[i] / exp) % 10] - 1] = nums[i];
            count[(nums[i] / exp) % 10]--;
        }

        for(int i=0;i<n;i++)
            nums[i] = output[i];
    }
```

```java
    public int maximumGap(int[] nums) {
        if(nums.length <= 1) return 0;

        //Radix Sort
        int n = nums.length;
        int max = Integer.MIN_VALUE;
        for(int i=0;i<nums.length;i++) {
            max = Math.max(max,nums[i]);
        }

        for(int exp=1;max/exp > 0; exp*= 10) {
            countSort(nums,exp);
        }

        int res = 0;
        for(int i=0;i<n-1;i++) {
            res = Math.max(res,nums[i+1] - nums[i]);
        }

        return res;
    }
}
```

H-Index

274. H-Index

Medium 1239 1857 Add to List Share

Given an array of integers `citations` where `citations[i]` is the number of citations a researcher received for their `i^th` paper, return compute the researcher's `h` **-index.**

According to the definition of h-index on Wikipedia: A scientist has an index `h` if `h` of their `n` papers have at least `h` citations each, and the other `n - h` papers have no more than `h` citations each.

If there are several possible values for `h`, the maximum one is taken as the `h` **-index.**

{3, 0, 6, 1, 5}
0 1 2 3 4

↓ Sort

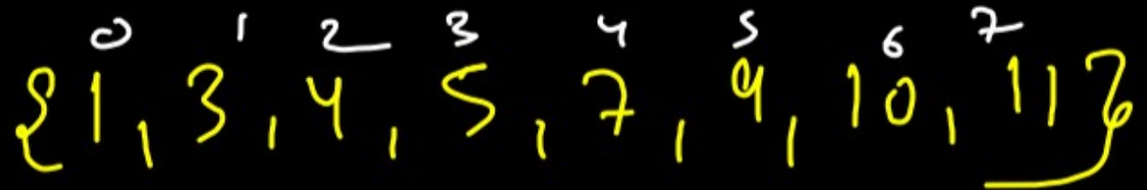
{0, 1, 3, 5, 6}

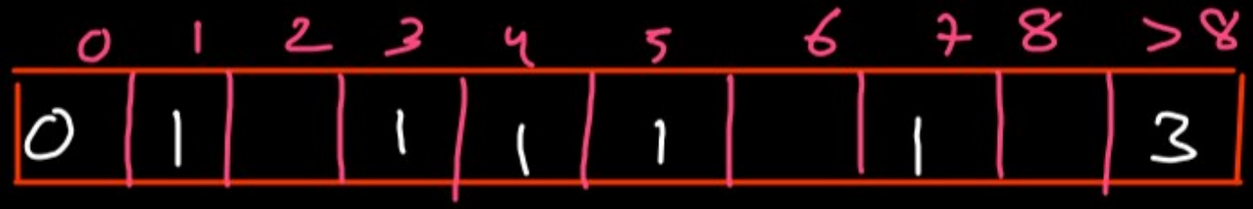
There are not atleast 5 papers with value ≥ 5.

kamse kam 0 papers with value ≥ 0

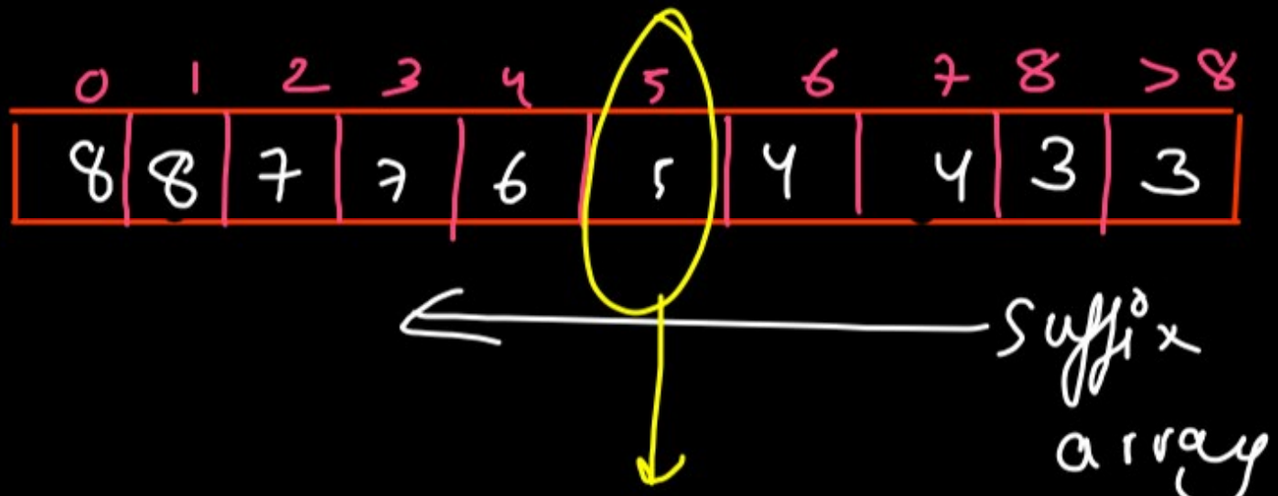
Highest such value is an

kamse kam 5 papers with value ≥ 5

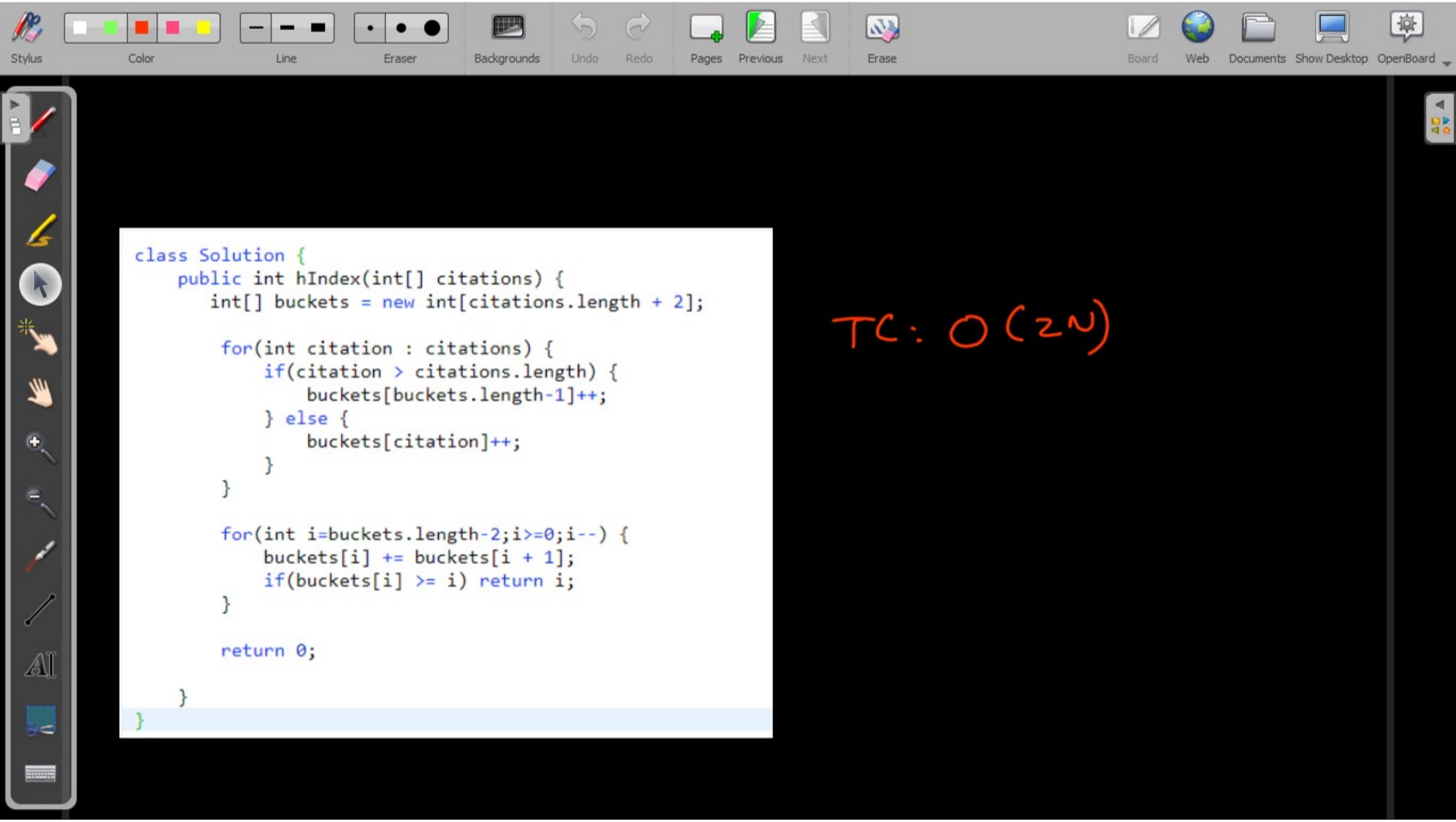




→ freq



ans = 5


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Show Desktop
OpenBoard
class Solution {
public int hIndex(int[] citations) {
int[] buckets = new int[citations.length + 2];
for(int citation : citations) {
if(citation > citations.length) {
buckets[buckets.length-1]++;
} else {
buckets[citation]++;
}
}
for(int i=buckets.length-2;i>=0;i--) {
buckets[i] += buckets[i + 1];
if(buckets[i] >= i) return i;
}
return 0;
}
}
TC: O(2N)