Lecture : Count and merge sort

Agenda

- count sort
- Merge 2 sorted arrays
- Merge sort
- Calculate no. of pairs such that A[i] > B[j]
- Inversion count
- stable sort and inplace sort.

Count sort

Qu find the smallest number that can be formed by rearranging the digits of given number in an array. Return the smallest number in form of an array.

6	3	4	2	7	2	1

⟶ 1, 2, 2, 3, 4, 6, 7.

4	2	7	3	9	0

⟶ 0, 2, 3, 4, 7, 9.

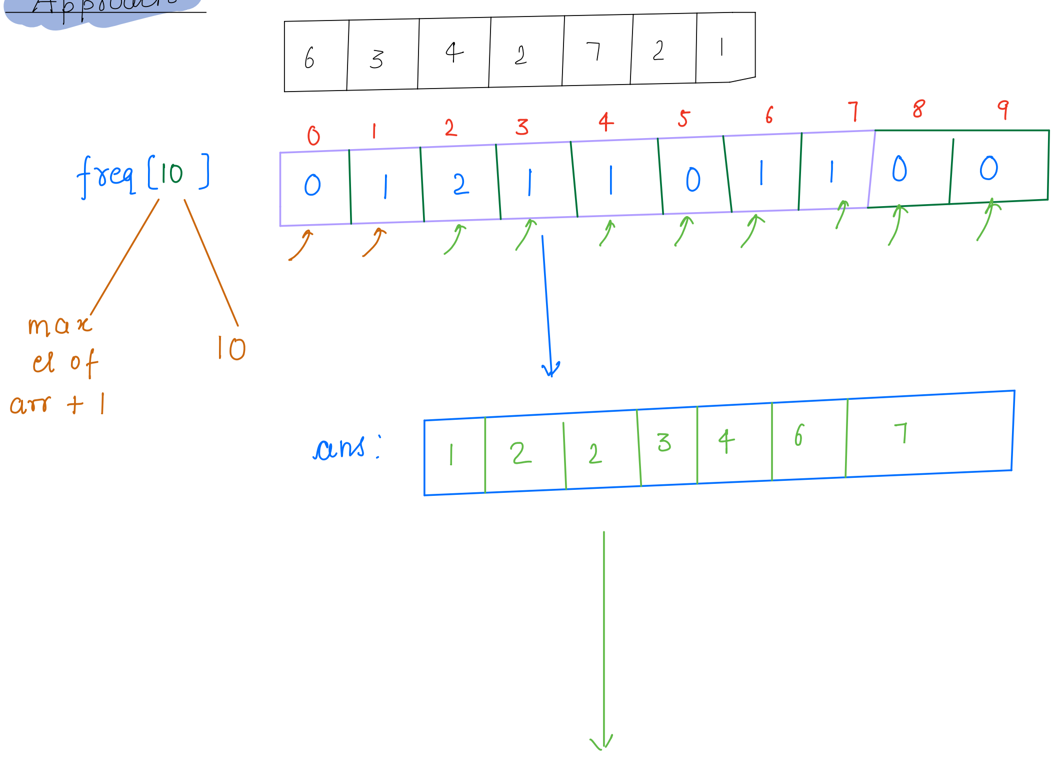
Brute force

```
Arrays.sort(arr);
```

TC: O(nlogn)

SC: O(1)

Approach



6	3	4	2	7	2	1

freq [10]

max el of arr + 1

10

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

ans :

1	2	2	3	4	6	7

Count sort

Algorithm

```
    void rearrange( int[] A) {

        int freq[10];

        for(int el: A) {
              freq[el]++;
        }
        int idx =0;

        for(int i=0; i< 10; i++) {
              int fr = freq[i];

                for(j=0; j< fr; j++) {
                     A[idx] =i;

                     idx++;
                }
      }
  }
```

TC: O(n)

SC: O(1)

Qu Will count sort work if range of A[i] is more than 10^9? [no]

10	29	10^9+7	10^8	10^9	10^9+2

freq[max el +1] ⟶ MLE

```
for(i=0; i<freq.length; i++){        ⟶ TLE

        - - - - - - - - - - ....

}
```

Qu Count sort on negative numbers. $[-9, +9] \longrightarrow n <= 10^5$

$-9 <= A[i] <= 9$

Approach

ip

0	1	2	3	4	5
−2	3	8	3	−2	3

op

−2	−2	3	3	3	8

$freq[max - min + 1] \longrightarrow freq[8 - (-2) + 1]$

$freq[11]$

0	1	2	3	4	5	6	7	8	9	10
~~1~~ 2					3 ~~1~~ ~~2~~					1
freq[-2]	f[-1]	f[0]	f[1]	f[2]	f[3]	- - - -	↑	- - - -		f[8]

$-2 - min$

$-2 - (-2)$

0

$x - min = 7$

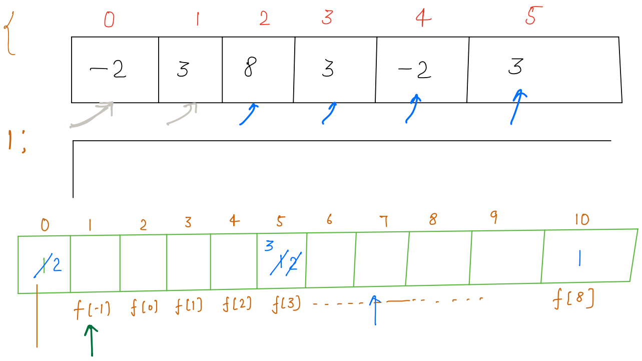
$x - (-2) = 7$

$x + 2 = 7$

$\boxed{x = 5}$

Algorithm

```
void rearrange( int[] A) {
    int freq[~~10~~];  max-min +1;
    for(int el: A) {
        freq[~~el~~] ++;
             el-min
    }
    int idx =0;
                  freq.length
    for(int i=0; i< ~~10~~; i++) {
        int fr = freq[i];
        for(j=0; j< fr; j++) {
            A[idx] = ~~i~~;   i+min
            idx++;
        }
    }
}
```



i=0 , fr=2

j=0 A[0] = 0+(-2)

j=1 A[1] = 0+(-2)

i=1. fr=0

i=2, fr=0

i=3 fr=0

i=4 fr=0

i=5 fr=3

j=0 A[2] = 5+(-2) =3

j=1 A[3]=3

j=2 A[4]=3

-2	-2	3	3	3	. . . .

TC: O(n)

SC: O(1)

Qu Given A[] where all odd elements are sorted and all even elements are sorted. sort the entire array.

0	1	2	3	4	5	6	7	8
2	5	4	8	11	13	10	15	21
2	4	5	8	10	11	13	15	21

Approach1 Arrays.sort(A);

Approach2

even[] =

0	1	2	3
2	4	8	10
i			

odd[] =

0	1	2	3	4
5	11	13	15	21
j				

→ $even[\underset{0}{i}] < odd[\underset{0}{j}]$ → 2 , i=1

even[1] < odd[0] → 4 i=2

even[2] > odd[0] → 5 j=1

even[2] < odd[1] → 8 i=3

even[3] < odd[1] → 10 i=4 {11, 13, 15, 21}

⋮

11

13

15

21

Copy remaining el.

⋮

i< even.length && j< odd.length

TC: O(n)

SC: O(n)

Algorithm

```
void merge(int[] A) {
    int n = A.length;
    even[n1];   } h|w
    odd[n2];    }

    int i=0, j=0, idx=0;
    while(i< n1  &&  j<n2) {                    // O(min(n1,n2))
        if(even[i] < odd[j]) {                     O(n)
            A[idx] = even[i];
            i++;
            idx++;
        } else {
            A[idx] = odd[j];
            j++;
        }   idx++;
    }

    while (i<n1) {                 } O(n)
        A[idx] = even[i];          }
        idx++;                     }
        i++;                       }
    }

    while (j<n2) {  copy remaining el of odd   } O(n)
        A[idx] = odd[j];                       }
        j++;                                   }
        idx++;
    }
}
```

n^3

n^2

TC: $O(n)$ $2n$ ✓

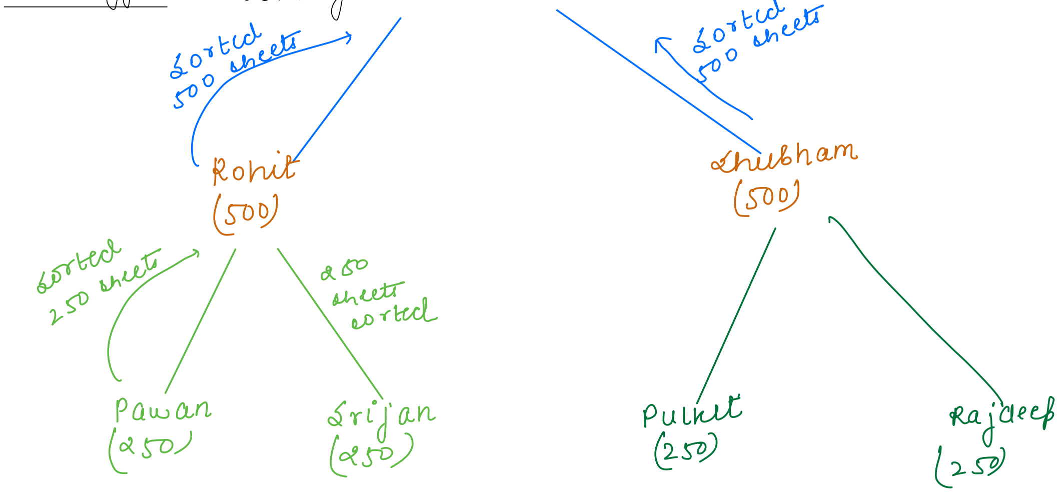
SC: $O(n)$ $O(1)$

Break: 8:32 - 8:45

Merge sort

1000 students

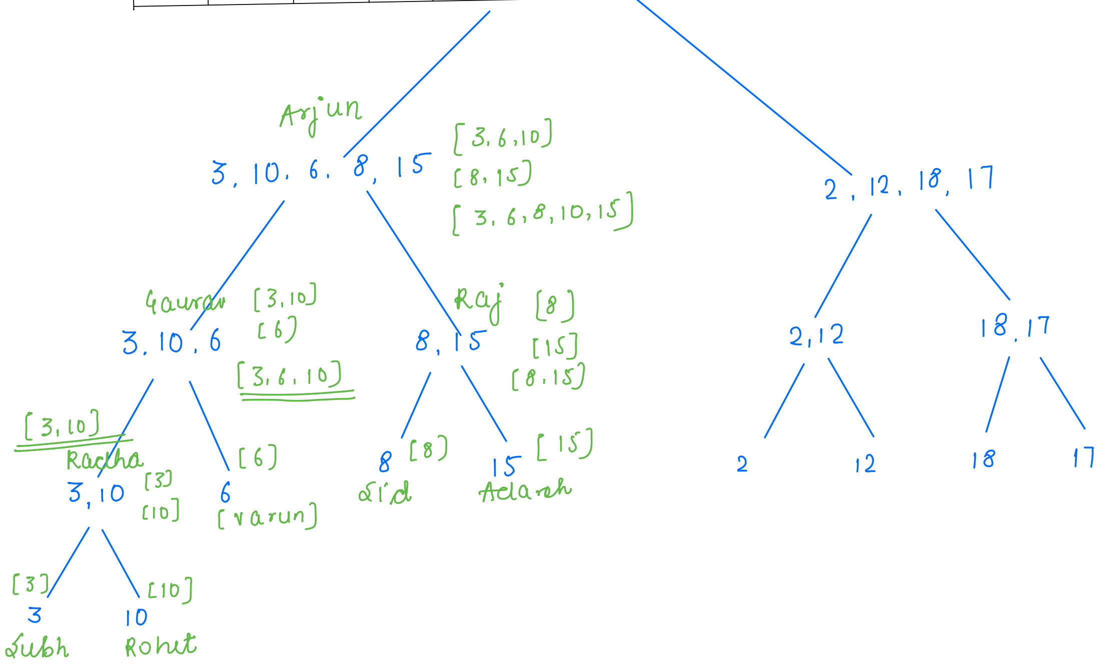
Analogy: sorting examination sheets. (roll no)



Example

Sorting numbers

0	1	2	3	4	5	6	7	8
3	10	6	8	15	2	12	18	17



Merging

↓

Merge 2 sorted arrays (Q).

Algorithm

0 A.length-1;(8)

0	1	2	3	4	5	6	7	8
3	10	6	8	15	2	12	18	17

$mid = \frac{0+8}{2} = 4$

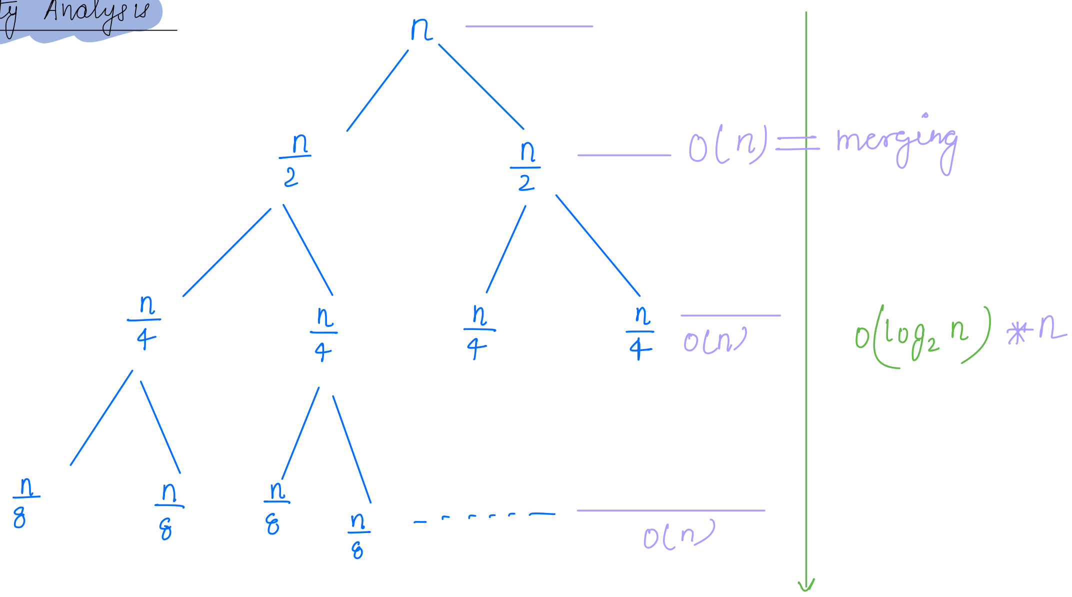
3, 10, 6, 8, 4

2, 12, 18, 17

```
void mergeSort( A[], l, r) {
    if( l==r) {
        return;
    }
    mid = (l+r)/2;
    mergesort( A, l, mid);
                  0   4
    merge sort( A, mid+1, r);
                    5     8
    merge( A, l, mid, r);
              0   4   8
}

void merge( int[] A , l, mid. r) {
    int n = A. length;
    left[mid-l+1] ;   }
    right[ r-mid  ]   }  Create this   Mandatory → h/w   Try it in O(1)

    int i=0, j=0, idx=0;

    while( i< n1  &&  j<n2) {
                                          // O(min(n1,n2))
                                             O(n)
        if( ~~even~~ left[i]  < ~~odd~~ right[j]) {
            A[idx] = ~~even~~ left[i];
            i++;
            idx++;
        } else {
            A[ idx] = ~~odd~~ right[j];
            j++;
            idx++;
        }
    }

 while  (i <n1)  {
        A[idx] = ~~even~~ left [i];        }  O(n)
        idx++;
        i++;
 }

 while  (j<n2) {  Copy remaining el of odd
        A[idx] = ~~odd~~ right[j];         }  O(n)
        j++;
        idx++;

 }
}
```

SC: ______________

Complexity Analysis



$$SC: O(1) + \underset{\substack{\uparrow \\ \text{recursive} \\ \text{space}}}{O(\log_2 n)}$$

Qu Given A[n] and B[m]. calculate number of pairs i, j such that A[i] > B[j]

A =

0	1	2
7	3	5

B =

0	1	2
2	0	6

Ans = (7, 2) (7, 0), (7, 6) (3, 2) (3, 0) (5, 2) (5, 0) ⇒ 7 pairs

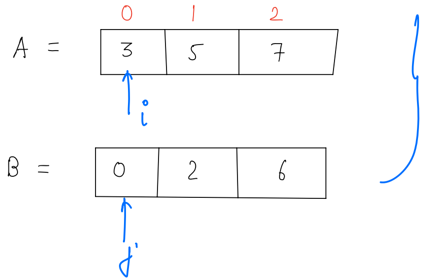
Brute force approach

2 loops

TC: O(n*m)

SC: O(1)

Approach 2



Sort both arrays

$i=0, j=0$ $B[0] < A[0]$

$A[0] > B[0]$ $\underbrace{(3,0)\ (5,0)\ (7,0)}_{\text{length of } A-i}$ 3 pairs

$i=0, j=1$ $A[0] > B[1]$ 3 pairs

$\underbrace{(3,2)\ (5,2)\ (7,2)}_{\text{length of } A-i}$

TC: $O\,n\log n + O\,m\log m + O(n+m)$

SC: $O(1)$

Qu Given A[n], calculate no. of pairs [i,j] such that $i<j$ and $A[i] > A[j]$, i and j are index of array.

A =

0	1	2	3	4
10	3	8	15	6

Ans = {
10, 3
10, 8
10, 6
8, 6
15, 6

0	1	2	3
5	2	6	1

$A[i] > A[j]$
$i < j$

5, 2
5, 1
2, 1
6, 1

0	1	2	3	4
5	3	1	4	2

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

Brute force approach

Approach

B.f ⇒ $O(n^2)$

Optimised: Merge sort + Above problem

Algorithm

Stable sort and inplace

Stable sort : relative order of eq el should not change while sorting

A =

0	1	2	3
6	5	3	5

↓

A =

3	5	5	6

Scenario   Airport check line

→ first come first serve, whoever comes first should be allowed first to checkin.

→ But acc. to airline, all members are not same. Some would be economic, business class, priviledged | priority .–––....

→ Srijan (economy)
Abhinav (")

Uppula (Business)

Thankyou 🙂