* 0-1 sorting :

ip : [1, 0, 1, 1, 0, 0, 0, 1]
0 1 2 3 4 5 6 7

op : [0, 0, 0, 0, 1, 1, 1, 1]
0 1 2 3 4 5 6 7

#1 : insertion, bubble, selection - O(N^2)

#2 : arr.sort (compare) - O(N log N)

#3 : counting / count sort Technique

1. cnt0 = 0 ✓ ✗ ✗ 4
   cnt1 = 0 ✗ ✗ ✗ 4

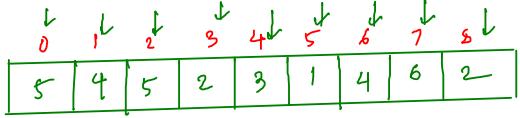
2. fill cnt0 no. of zeros
   fill cnt1 no. of ones

[0, 0, 0, 0, 1, 1, 1, 1]
0 1 2 3 4 5 6 7

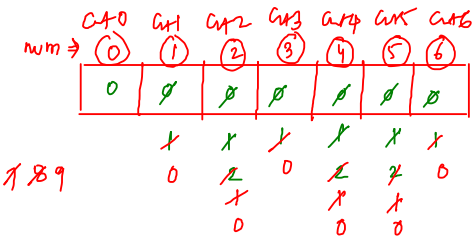
```
int idx = 0
while (cnt0 > 0) {
    arr[idx] = 0
    idx++
    cnt0--;
}
while (cnt1 > 0) {
    arr[idx] = 1
    idx++
    cnt1--;
}
```

Expanding count sort 1, 2, 3, 4, 5, 6 sorting

* this can be used only if elements are in range $0 \rightarrow 10^5$

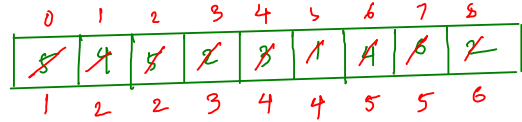


(count Array)

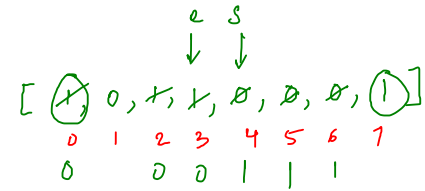



```
for (0 → n) {
    num = arr[i]
    cnt[num]++;
}
idx = 0;
for (num = 0 → 6) {
    while (cnt[num] > 0) {
        arr[idx] = num;
        idx++;
        cnt[num]--;
    }
}
```

idx = 0 1 2 3 4 5 6 7 8



#4 : Two-pointer Technique




```
while (s != e) {

    if (arr[s] == 1 && arr[e] == 0) {
        swap(arr[s], arr[e]);
        s++; e--;
    }
    else if (arr[s] == 0) s++;
    else if (arr[e] == 1) e--;
}
```

a(s) == 1
⇒ not correct

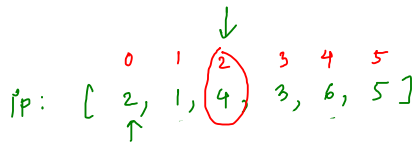
a(s) == 0
⇒ correct (start++)

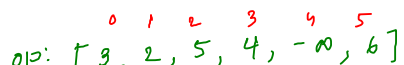
a(e) == 1
⇒ correct (end--)

a(e) == 0
⇒ not correct

* Both are not correct
a(s) == 1 and a(e) == 0
swap(a(s), a(e))
s++, e--;

Smallest greater Element :





* find minimum in ele > arr[i]

curr_num = 2

smallest_greater = ∞ # 3

4 > 2 3 > 2 6 > 2 5 > 2

4 < ∞ 3 < 4 6 < 3 5 < 3

curr_num = 1

smallest_greater = ∞ 2

2 > 1 4 > 1 3 > 1 6 > 1 5 > 1

2 < ∞ 4 < 2 ✗ 3 < 2 ✗ 6 < 2 ✗ 5 < 2

curr_num = 4

smallest_greater = ∞ # 5

6 > 4 5 > 4

6 < ∞ 5 < 6

* AS sorting 5 : $(1 \rightarrow 10^5)$ (+ve)

ip: [5, 6, 2, 7, 4]

2nd 1st 1st 2nd

0 1 2 3 4

op: w, x, y, z

⇒ w * x - y * z

⇒ maximum

w = 7 y = 4

x = 6 z = 2

7 * 6 - 4 * 2

⇒ 42 - 8

⇒ 34

op ⇒ (1st max * 2nd max) - (1st min * 2nd min)

when will get max diff ?

(a, b)

a - b = max diff

a↑ - b↓

maxele - minele = max diff

(w * x) - (y * z)

a↑ b↓

maxProd - minProd = maxProddiff

w↑ * x↑ = maxProd

y↓ * z↓ = minProd

what if the range of ele $(-10^5 \to 10^5)$:

ip: [-200 , -100 , -50 , -60 , 1 , 2 , 3 , 4]

-60

0 1 2 3 4 5 6 7

op:

$(-200 * -100) - (-50 * 0)$

$20,000 - 0$

$\Rightarrow 20,000$

1st min -ve * 2nd min -ve

-ve * -ve → +ve

arr(0) * arr(1)

arr(n-1) * arr(n-2)

} max prod

+ve * +ve → +ve

1st max +ve * 2nd max +ve

max prod − min prod

arr(n-1) * arr(n-2) − arr(0) * arr(1)

3 * 4 − (-200 * -100)

3 * 4 − (20,000)

⇒ 12 − 20000

⇒ − 19988

X Failed

(negative)

max Negative * min positive → -ve

1st min positive * 2nd min positive → +ve

(no negative)

} min prod

★ As Sorting 4 :

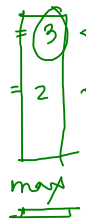
① ip: $[\underset{0}{2}, \underset{1}{1}, \underset{2}{3}, \underset{3}{4}]$ $L = 3$

a, b, c ⇒ max ele − min ele <= L

(1, 2, 4) ⇒ 4 − 1 = 3 <= L
(1, 2, 3) ⇒ 3 − 1 = 2 <= L

} ? max possible = 3

op: 3



② if L = 2,

(1, 2, 3) ⇒ 3 − 1 = 2 <= L
X (1, 2, 4) ⇒ 4 − 1 = 3 <= L X

op: 2

#2 : Sorting — O(N Log N)

[2, 1, 3, 4] L = 3

- ✗ Count Sort Technique
- ✗ Storing In Objects Technique
- ✓ Moving Stream Technique
- e.g. Two pointer Technique (when arr is sorted)

→ [1, 2, 3, 4] , L = 3

min = 1, max = 4

arr(e) − arr(s) <= L

⇒ e − s − 1 (possible ways)

count = (e − s − 1)
= 3 − 0 − 1
= 2

start

1 x 4
1 y 4
1 z 4

*

[1, 2, 3, 4]

3 − 1 <= L
2 <= 2

4 − 1 <= L
3 <= 3

count += (2 − 0 − 1)
= 1
= 1 + (3 − 0 − 1)
= 1 + 2 = 3
= 3 + (3 − 1 − 1)
= 3 + 1 = 4

4 − 2 <= L
2 <= L

1 x 4 x y 4 y z 4 z 4 4
1 y 4 x z 4 y y 4
1 z 4 x 4 4
1 4 4

*

[1 x y z 4 4 4 4]

return = e − s − 1

r − 0 − 1 = 2

count = ((return * (return + 1)) / 2
(2 + 2 + 1)

1 x 4 x y 4 z 4
1 y 4 x z 4
1 z 4