Count sort

Q Sort an array of digits 0 – 9 in desc order

Eg1 2 4 8 1 9 3 4 0

⇒ 9 8 4 4 3 2 1 0

Brute: inbuilt sort & reverse

TC: nlogn

Idea: Maintain count array.

	0	1	2	3	4	5	6	7	8	9
count[10] =	1	1	1	1	2	0	0	0	1	1

Dry run above eg

9 → count_9

8 → count_8

7 → ... 2

:

0 → ... 0

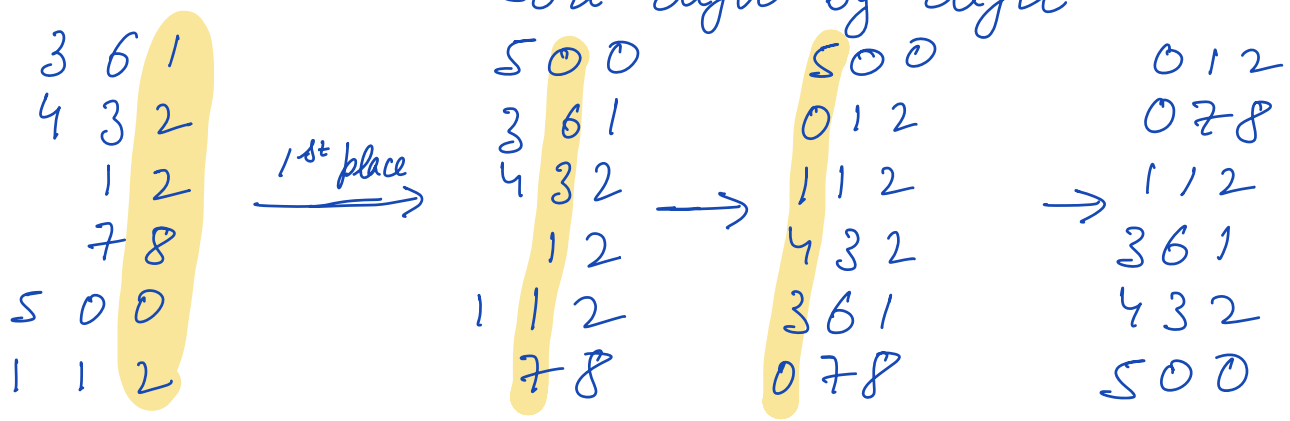
Code

```
int count[10] = {0};
for (i=0; i<n; i++) {
    count[a[i]]++
}
for (i=9; i>=0; i--) {
    for (j=0; j<count[i]; j++) {
        print(i)
    }
}
```

TC: O(n)

SC: O(1)

- **Radix Sort idea**



- When digits of 2 elem are equal, you will keep them in original order.

- When you can place any of the 2 elem before the other, but you choose to keep them in same order.

  ⇒ Stable sorting.

$x \% 10$

$\left(\frac{x}{10}\right) \% 10$

$\left(\frac{x}{100}\right) \% 10$

[361, 432, 12, 78, 500, 112]

Q2 Sum the difference

sum of max - min for all subsequences.

Eg 3, 2, 5

(3-9) % 5

3 - 4 = -1

	max	min
{ }	-	-
3	3	3
2	2	2
5	5	5
3, 2	3	2
2, 5	5	2
3, 5	5	3
3, 2, 5	5	2
	28	19

= 9

Obs: sum (max - min)
= sum (max) - sum (min)

Idea: How many times is A[i] the max

⟹ Contribution technique

$$sum(max) = \sum_{i=0}^{n-1} A_i * (\text{subseq when } a_i \text{ is max})$$

⇒ Sort the array

after sorting

0	1	2
2	3	5
0	1	2

$2 \times 2 \times 2 \times 2 \ldots \times 2 = 2^i$

$$sum(max) = \sum_{i=0}^{n-1} A_i * 2^i$$

$$sum(min) = \sum_{i=0}^{n-1} A_i * 2^{n-i-1}$$

idx	0	1	2
	2	3	5
big	2	1	0

idx + big = n-1

big = n-i-1

TC: nlogn

SC: O(1)

```
sum_max = 0
for( i=0; i<n; i++)
    sum_max = [sum_max +
              (a[i] * pow(2,i,m)) %m] %m
//get sum_min
return (sum_max - sum_min + m) %m
```

$(a-b)\%m = (a\%m - b\%m + m)\%m$

Q3 Max unsorted subarray

Given array, Find min subarray such that sort a[s]...a[e] sorts the whole array. Return the subarray

Eg 1

0	1	2	3	4
10	30	20	40	50

Idea: Compare with sorted array

0	1	2	3	4	
10	30	20	40	50	s = 1
10	20	30	40	50	e = 2

Where does mismatch start from the left?
Where does mismatch start from the right?

Hence we have our start & end

Code

```
pair<int, int> sub(int A[]){
    int B[] = A
    sort(B)
    start = -1
    end = -1

}
```

```
for (i=0 ; i<n ; i++ ){
    if (A[i] != B[i]) {
        start = i
        break
    }
}

for (i=n-1 ; i>=0 ; i--){
    if (A[i] != B[i]) {
        end = i
        break
    }
}

return {start, end}
}
```

TC: O(nlogn)
SC: O(n)

Q4 K closest points to origin (0,0)

Given list of points ⇒
return K closest points to origin

Eg - 1,3 $\sqrt{10}$ -2,2 $\sqrt{8}$ K = 1
ans = [{-2,2}]

Eg 1,3 $\sqrt{10}$ 1,-1 $\sqrt{2}$ 2,-1 $\sqrt{5}$ K = 2

ans = [{1,-1} , {2,-1}]

Distance from origin of point x, y

sqrt (x^2+y^2)

We need to sort on basis of distance.

Ques If I compare x^2+y^2 instead of sqrt (x^2+y^2) ⇒ this works

Idea : Sort using comparator

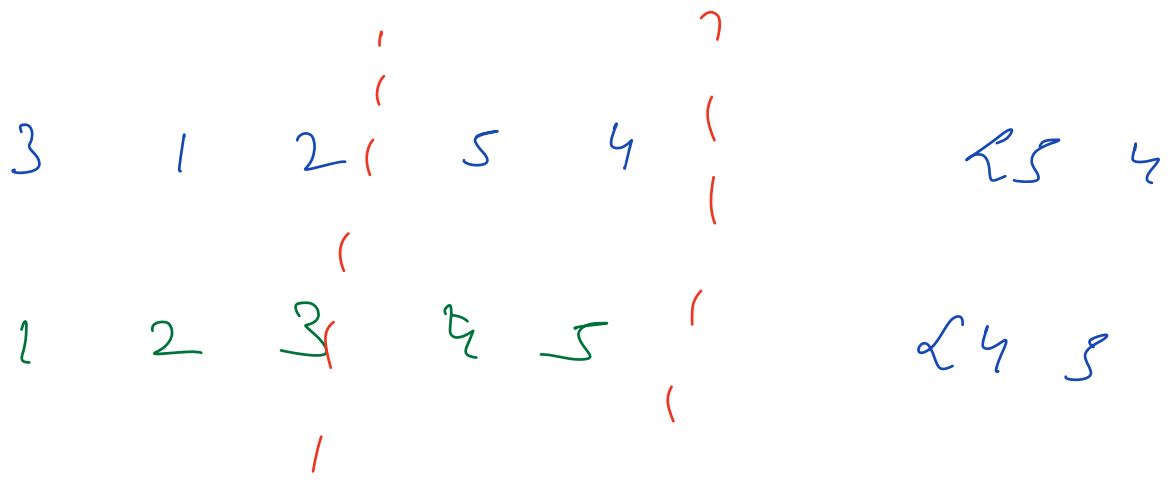
Code

```
bool cmp ( pair<int, int> a, pair<int, int> b){

    long d1 = (long) a.first * a.first +
              (long) a.second * a.second

    long d2 = (long) b.first * b.first +
              (long) b.second * b.second

    if ( d1 < d2 )
        return true
    else
        return false
}
```

```
list< pair<int,int> > closest ( list< pair<int,int> > A, int K ){

    list< pair<int,int> > ans
    sort ( A, cmp )
    for( i=0 ; i<K ; i++ ){
        ans.insert (A[i])
    }
    return ans
}
```

TC: O(nlogn)
SC: O(n)

{done}



Max chunks to make sorted

Observe the prefix max of array.

	0	1	2	3	4
A[5] =	1	0	2	4	3
mx	1	1	2	4	4

Obs: mx = i where chunk can be created.

1) Create pf_max array

2) if (pf_max[i] == i) count++