```
int m = A.length;
int[] prefStep = new int[m];
for(int[] ops:ops){
  int l = ops[0];
  int r = ops[1];
  prefStep[l]+=1;
  if(r+1<m) prefStep[r+1]-=1;
}
//convert prefix step array into prefix sum array
for(int i= 1;i<m;i++){
  prefStep[i] += prefStep[i-1];
}

Arrays.sort(prefStep);
Arrays.sort(A);

int  score = 0;
int mod = 1000000007;
for(int i = 0;i<q;i++){
  score = (score%mod +  (A[i] * prefStep[i])%mod)%mod;
}

return score%mod;
```

A: [5 2 3 1 4 6]

ps: [1 2 5 3 2 1]

1 2 3 4 5 6

1 1 2 2 3 5

ops) 1 3

2 4

2 2

2 5

0 2

0+1+2+6

You are given an array `A` of `N` integers. You are also given another array `ops` describing operations where `ops[i] = (l, r)`. **l and r are 0-indexed**. For every pair in `ops` calculate the sum of `A[l] + A[l+1] +...+ A[r-1] + A[r]`.

Your score is the sum of all the answers from `ops`. You have to find the maximum score possible, if you can arrange the array `A` in any order.

The answer can be very large, so return it modulo `1000000007`.

Note Complete the given function. The input and output would be handled by the driver code.

5 2 3 1 4 6

A= 1 2 3 5 4 6

1 2 5 3 2 1

1 2 3 4 5 6

1 1 2 2 3 5

ops) 1 3

2 4

2 2

2 5

0 2

+1

-(r+1)

1 2 5 3 2 1

=) 6 13

=) 8 15

=) 3 6

=) 14 18

=) 10 9

Score= 41 13

[1 3 6 10 6 2 9 3]

Range queries

5, 3, 6, 2, 1, 7, 9, 10

0 1 2 3 4 5 6 7

L	R	
1	5	=>19
6	7	=> 19
4	6	=> 17
2	5	=> 16

$n = 10^5$

$q = 10^5$

```
int[] prefStep = new int[n];

for(int[] trip:trips){
  int numPass = trip[0];
  int source = trip[1];
  int destination = trip[2];
  prefStep[source] += numPass;
  prefStep[destination] -= numPass;
}

//now i will take the prefix sum of the pref step array
for(int i = 1;i<n;i++){
  prefStep[i] += prefStep[i-1];
}
//now check if we exceed the number of passengers on any
for(int ps:prefStep) if(ps > capacity) return false;
return true;
```

trips [

[3 2 5]

[2 3 7]

[2 5 8]

[3 6 8]

]

0 0 3 5 5 4 7 5 0 0

0 1 2 3 4 5 6 7 8 9

9, 0, 1

3, 3, 7

9

0 1 2 3 4 5 6 7

3 3 3 2 2 2 2

2 2 2 3 2

0 3 5 5 4 7 5

0-4

Prefix Sum

5 3 6 2 1 7 9 10

0 1 2 3 4 5 6 7

=> 5 8 14 16 17 24 33

0 1 2 3 4 5 6 7

pref[0] = arr[0];
for(int i = 1;i<arr.length;i++){
  pref[i] = pref[i-1] + arr[i];
}

[3]

pref(3)

4-6 ≠ 17

33-16 = 17

pref[r] - pref[l-1]

1) PS arr => O(n)

2) Q → O(1)

O(n+Q)

A - B - C + D

O(1)

5 7 8 11 17

8 12 14 19 29

9 19 28 37 50

18 32 44 55 72

28 44 59 74 95