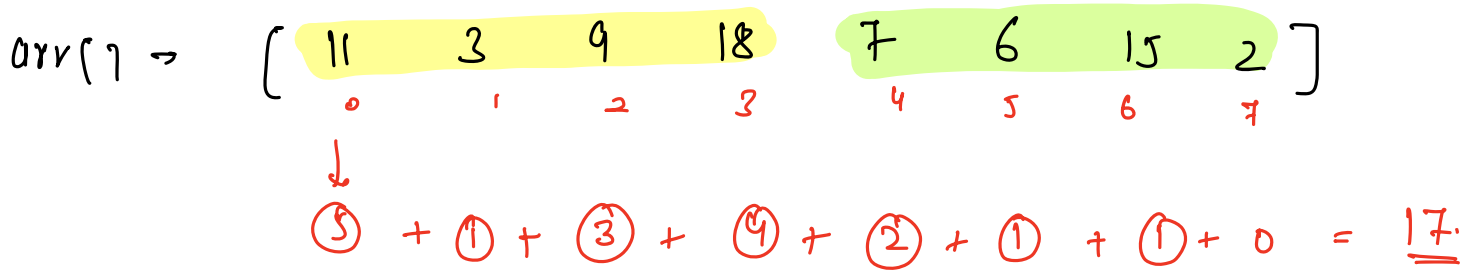
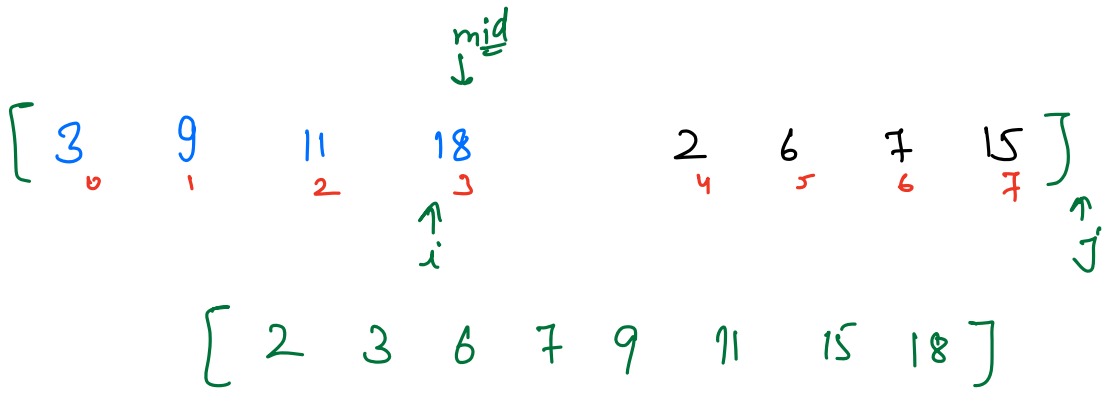
→ Inversion Count

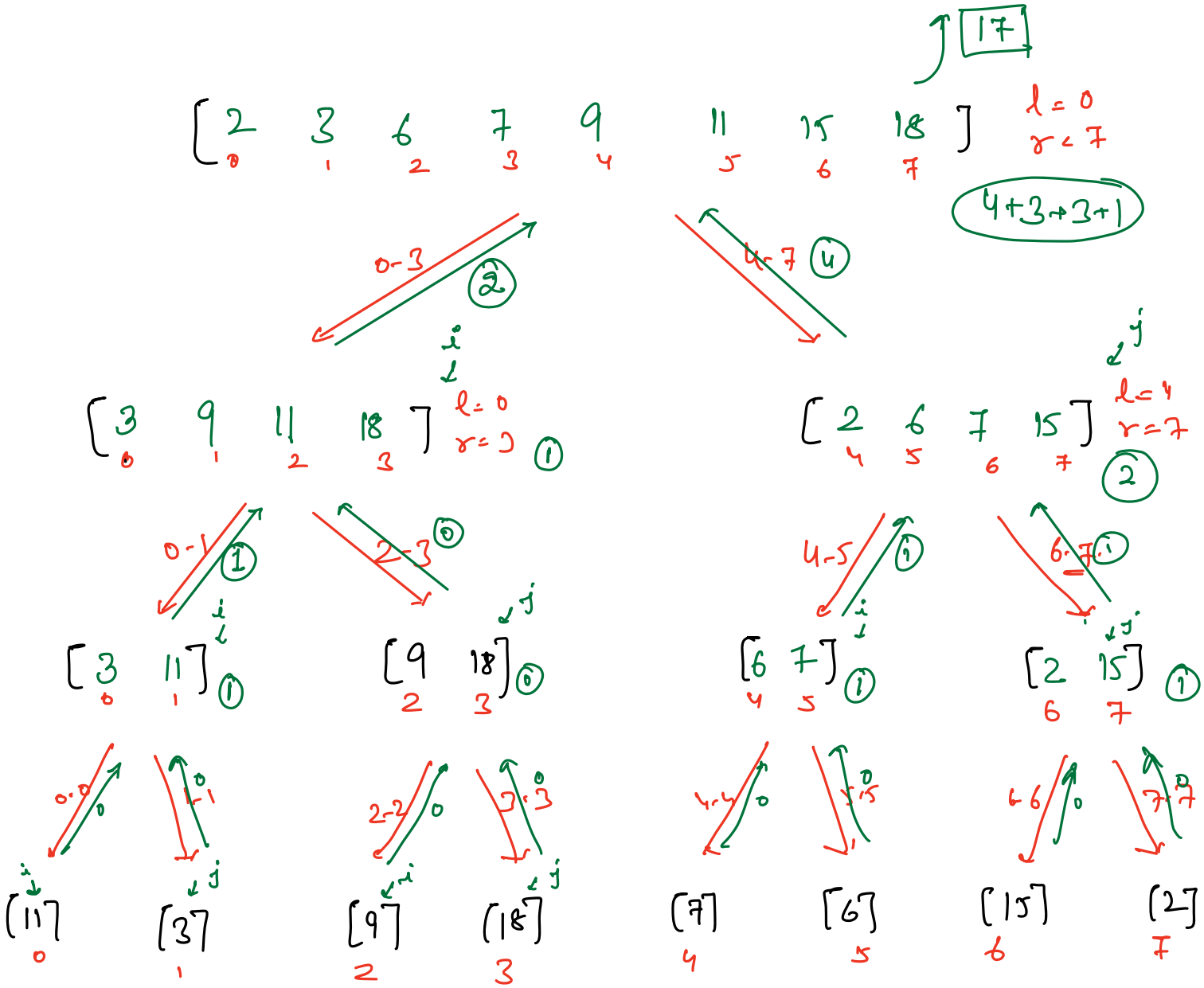
Given an array of integers A. If i < j and A[i] > A[j], then the pair (i, j) is called an inversion of A. Find the total number of inversions of A modulo (109 + 7).





ip += (mid-i+1)

ip = 0 4 7 10 11


17
2 3 6 7 9 11 15 18
0 1 2 3 4 5 6 7
l = 0
r = 7
4+3+3+1
0-3
2
4-7
4
3 9 11 18
l = 0
r = 3
1
2 6 7 15
l = 4
r = 7
2
0-1
1
2-3
0
4-5
1
6-7
1
3 11
1
9 18
0
6 7
1
2 15
1
0-0
0
1-1
0
2-2
0
3-3
0
4-4
0
5-5
0
6-6
0
7-7
0
11
0
3
1
9
2
18
3
7
4
6
5
15
6
2
7

code →

$1 \le N \le 10^5$

$1 \le A[i] \le 10^9$

```
int inversionPairs( arr, l, r){
    if( l == r) { return 0}

    mid = (l+r)/2;
    f1 = inversionPairs (arr, l, mid);
    f2 = inversionPairs ( arr, mid+1, r);
    f3 = mergeTwoSorkedSubarrays (arr, l, mid+1, r);
    return (int)((f1 + f2 + f3) % (10^9+7));
}
```

TC → O(NlogN)
SC → O(N)

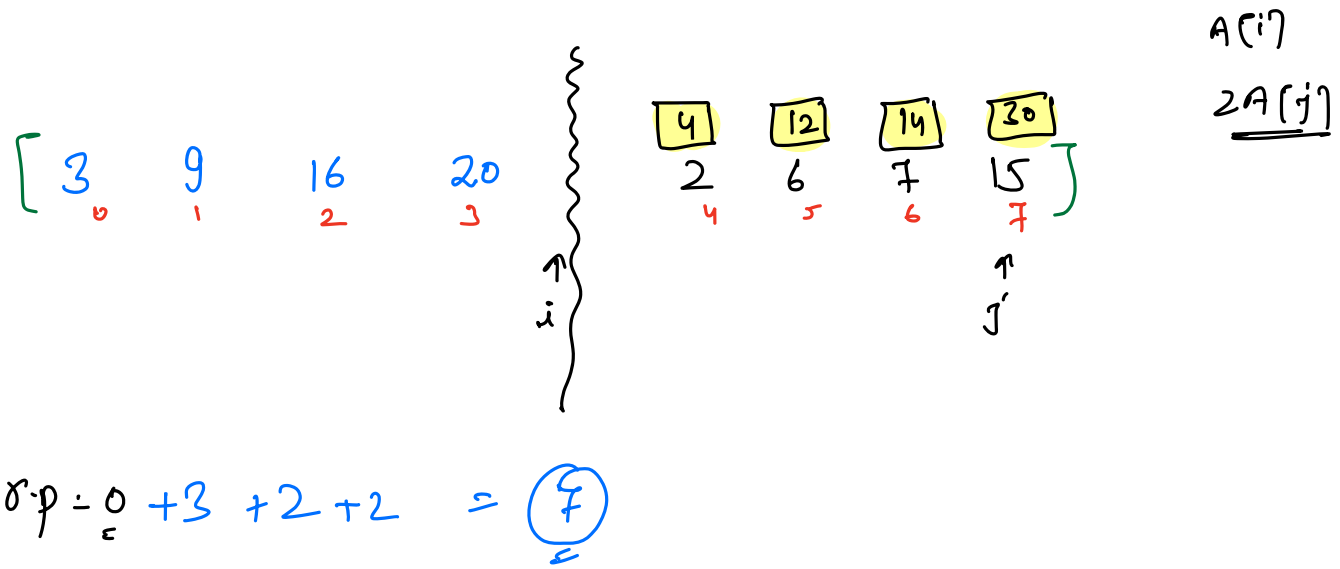
```
long mergedTwoSortedSubArrays( int[] A, int l, int y, int r){
    int C[r-l+1]; long ip = 0
    i = l, j = y, k = 0;
    while( i < y && j <= r){
        if( A[i] <= A[j]){
            C[k] = A[i]
            i++ ; k++
        }
        else{
            C[k] = A[j];
            j++, k++ ;
            ip = ip + (y-i);
        }
    }
    while( i < y){
        C[k] = A[i]
        i++, k++
    }
    while( j <= r){
        C[k] = A[j];
        j++, k++
    }
    // copy the elements from C[] to A[]
    k = 0
    for( i = l; i <= r; i++){
        A[i] = C[k]
        k++
    }
    return ip;
}
```

{T.C → O(N)
S.C → O(N)}

Reverse Pair

Given an array of integers A, we call (i, j) an important reverse pair if i < j and A[i] > 2*A[j].

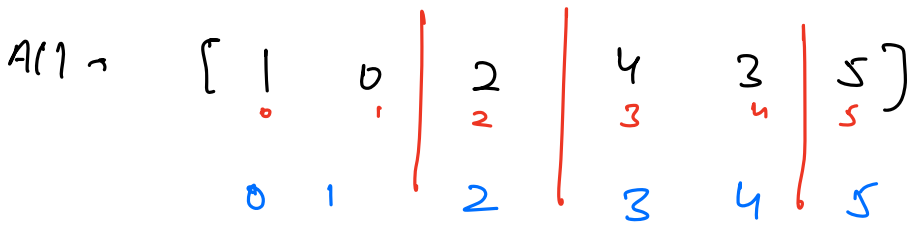
Return the number of important reverse pairs in the given array A.

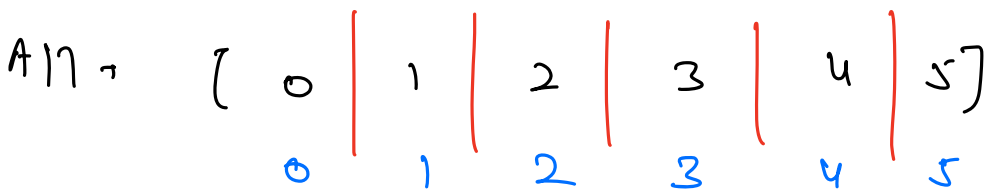


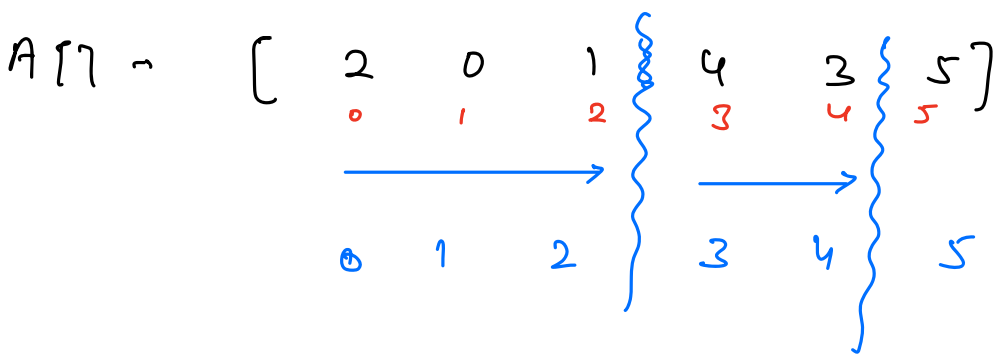
Max Chunks To Make Array Sorted

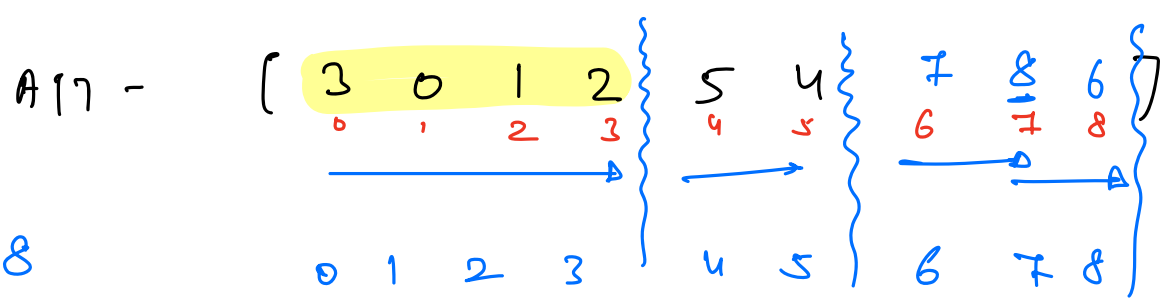
Given an array of integers A of size N that is a permutation of [0, 1, 2, ..., (N-1)], if we split the array into some number of "chunks" (partitions), and individually sort each chunk. After concatenating them in order of splitting, the result equals the sorted array.

What is the most number of chunks we could have made?









code -->

https://pastecode.io/s/y0bvhf3o

Re-arrange Array

Given an array A of size N. Rearrange the given array so that A[i] becomes A[A[i]] with O(1) extra space.

$0 \leq A[i] \leq N-1$

arr = [1 3 0 4 2]
0 1 2 3 4

a[i] → a[a[i]]

ans = [3 4 1 2 0]
0 1 2 3 4

#code...

```
ans[N];
for( i = 0; i<N; i++){
        val = arr[i]
        ans[i] = arr[val];
}
return ans[];
```

arr = [1 3 0 4 2]
indices: 0 1 2 3 4

3 4

arr[arr[i]]

prev value at index → 0 is also required.

idea → store prev & new value at ith index.

value → [0, N-1]. ⇒ use % → remainder

[dvd = quo * div + remainder]

[arr[i] → old value * N + new value]

{ old value = arr[i] / N
  new value = arr[i] % N }

arr = [1 3 0 4 2]
indices: 0 1 2 3 4

N = 5.

[5+3 15+4 0+1 20+2 10+0]
[5 15 0 20 10]
indices: 0 1 2 3 4

[8 19 1 22 10]
[3 4 1 2 0]

code

```
for (i=0; i<N; i++){
    arr[i] += N
}
```

```
for( i=0; i < N; i++){
    idx = arr[i]/N
    old value = arr[idx] /N ;
    arr[i] += oldvalue ;
}
```

[T.C → O(N)
S.C → O(1)]

```
for( i=0; i<N; i++){
    arr[i] = arr[i] %. N
}
```

{ Storing 2 values at 1 index ⇒ /. % }