**4 kyu**

**Sum of Intervals**

1612391% of 3364 of2,501[xDranik](https://www.codewars.com/users/xDranik" \o "This kata's Sensei)

C++

* [TRAIN AGAIN](https://www.codewars.com/kata/sum-of-intervals/train/cpp)
* [NEXT KATA](https://www.codewars.com/trainer/cpp)

Details

[Solutions](https://www.codewars.com/kata/sum-of-intervals/solutions/cpp)

[Forks (6)](https://www.codewars.com/kata/sum-of-intervals/forks/cpp)

[Discourse (65)](https://www.codewars.com/kata/sum-of-intervals/discuss/cpp)

* Add to Collection
* |
* Share this kata:

Write a function called sumIntervals/sum\_intervals() that accepts an array of intervals, and returns the sum of all the interval lengths. Overlapping intervals should only be counted once.

**Intervals**

Intervals are represented by a pair of integers in the form of an array. The first value of the interval will always be less than the second value. Interval example: [1, 5] is an interval from 1 to 5. The length of this interval is 4.

**Overlapping Intervals**

List containing overlapping intervals:

[

[1,4],

[7, 10],

[3, 5]

]

The sum of the lengths of these intervals is 7. Since [1, 4] and [3, 5] overlap, we can treat the interval as [1, 5], which has a length of 4.

**Examples:**

sum\_intervals( {

{1,2},

{6, 10},

{11, 15}

} ); // => 9

sum\_intervals( {

{1,4},

{7, 10},

{3, 5}

} ); // => 7

sum\_intervals( {

{1,5},

{10, 20},

{1, 6},

{16, 19},

{5, 11}

} ); // => 19

<https://www.codewars.com/kata/sum-of-intervals/cpp>

#include <iostream>

#include <stdio.h>

#include <vector>

#include <utility>

#include <algorithm>

#include <map>

#include <math.h>

using namespace std;

vector<pair<int,int> > merge( vector<std::pair<int, int> >& intervals )

{

if ( intervals.size() <= 1 ) return intervals;

// start\_or\_end save the point position and start or end imformation

map<int, int> start\_or\_end;

//for ( const auto& in : intervals ) {

for(int i =0; i<intervals.size(); i++) {

start\_or\_end[intervals[i].first] ++;

start\_or\_end[intervals[i].second] --;

}

vector<pair<int,int> > res;

pair<int,int> inter( INT\_MAX, INT\_MIN );

int cnt = 0;

//for ( const auto& se : start\_or\_end ) {

for(std::map<int,int>::iterator se = start\_or\_end.begin(); se != start\_or\_end.end(); se++)

{

// it is start point

if ( se->second > 0) {

inter.first = std::min( inter.first, se->first );

cnt += se->second;

}

// it is end point

else if ( se->second < 0) {

cnt += se->second;

inter.second = std::max( inter.second, se->first );

if ( cnt == 0 ) {

res.push\_back( inter );

inter.first = INT\_MAX;

inter.second = INT\_MIN;

}

}

// it is both start and end point

else if( cnt == 0){

res.push\_back( make\_pair( se->first, se->first ) );

}

}

return res;

}

int sum\_intervals(std::vector<std::pair<int, int> > intervals) {

vector<pair<int, int> > mezclados = merge(intervals);

int sum =0;

for(int i =0; i<mezclados.size(); i++) {

sum += mezclados[i].second - mezclados[i].first;

}

return sum;

}

int main() {

/\*

std::vector<std::pair<int, int> > intervals; //= {{1, 5}};

//{{1, 5}, {6, 10}};Equals(8))

//intervals.push\_back(make\_pair(1,5));

//intervals.push\_back(make\_pair(6,10));

//assertEquals(78, sumIntervals(new int[][]{{-245, -218}, {-194, -179}, {-155, -119}}));

intervals.push\_back(make\_pair(-245, -218));

intervals.push\_back(make\_pair(-194, -179));

intervals.push\_back(make\_pair(-155, -119));

cout << sum\_intervals(intervals) << endl;

// Equals(4));

\*/

std::vector<pair<int,int> > intervals;

intervals.push\_back(make\_pair(-245, -218));

intervals.push\_back(make\_pair(-194, -179));

intervals.push\_back(make\_pair(-155, -119));

vector<pair<int,int> > mer = merge(intervals);

for(int i =0; i<mer.size(); i++) {

cout << mer[i].first << endl;

cout << mer[i].second << endl;

}

cout << sum\_intervals(mer) << endl;

return 0;

}