22 Coordinate Compression

Introduction

Coordinate compression is a technique to create reduced form of an array or matrix. It is mainly used to map larger values to smaller values to remove the vacant space. So that we can represent the input data in a compressed array or matrix.

Problem 1: Counting frequencies of numbers

Given N integers, find the frequency of all the numbers and print them in the given order. The range of the integers is -10^9 … 10^9 and input may contain duplicates.

Sample Input

10

10 -5 10^9 10 5 -100 10^6 -5 10 -10^6 10

Sample Output

4 2 1 4 1 1 1 2 4 1 10

Solution 1: Compression and Frequency Array

If we could use the input values as indices of an array, we could easily solve this problem using a frequency array. We could initialize the frequency array into zeros and increase the corresponding element by one for each input value. Negative values and too large values in the input prevent us from using this solution.

We can replace the input values with the values in the range 0 to N-1 and still use the same technique.

Algorithm:

0	1	2	3	4	5	6	7	8	9
10	-5	10^9	10	-100	10^6	-5	10	-10^6	-100

Input array

Step 1: Sort the input values and eliminate duplicates. *O(NlgN)*

0	1	2	3	4	5
-10^6	-100	-5	10	10^6	10^9

Reduced array with M elements

Step 2: Declare a frequency array with M elements and initialize it to zeros.

0	1	2	3	4	5
0	0	0	0	0	0

Frequency array with M elements

Step 3: Traverse the input array. Find the position of each element in the reduced array (binary searching) and increase the corresponding position by one in the frequency array. *O(NlgN)*

0	1	2	3	4	5
1	2	2	3	1	1

Frequency array after processing the input array.

Step 4: Re-traverse the input array. Find the position of each element in the reduced array (binary searching) and print the corresponding element in the frequency array. *O(NlgN)*

Solution 2: Using a Map Data Structure

We simply use a **map** or **unordered_map** in C++ or **HashMap** in Java where the input values are the keys and the frequencies are the values.

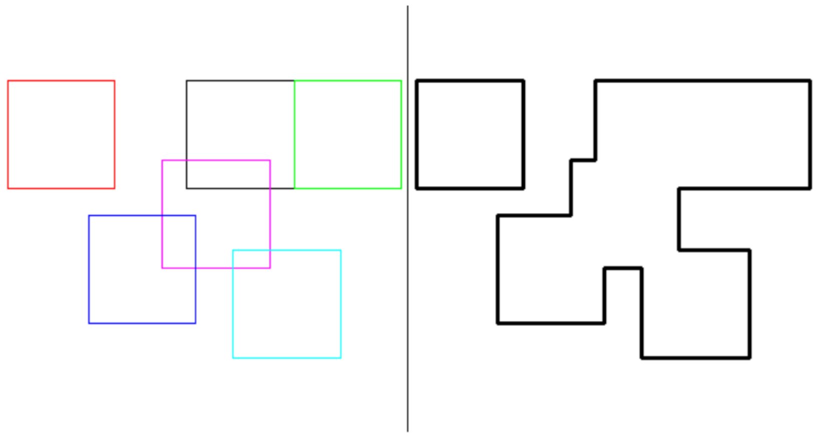
map<int, int> m;

unordered_map<int, int> m;

HashMap<Integer, Integer> freqMap = new HashMap<Integer, Integer>();

Problem 2: Area of Union of N Rectangles

Given N (1 <= N <= 1000) rectangles with edges parallel to axis, calculate the area of the union of all rectangles. The first line of the input contains N. Each of the following N lines contains 4 integers separated by spaces, defining 2 opposite vertices of the rectangle. The range of the integers is -10,000 … 10,000. Output should contain a single integer that is the resulting area of the rectangles' union.



Solution 1: Sweep Line

The area of union of N rectangles can be calculates in *O(NlgN) +* $O(N^2)$ time with the **sweep line** and **set** data structure; and *O(NlgN) + O(NlgN)* time with the **seep line** and **segment tree** data structure. We have already discussed this solution in the previous chapter.

Solution 2: Coordinate Compression

Step 1: Separately compress the x coordinates and y coordinates in to different arrays with X and Y elements. *O(NlgN)*

Step 2: Declare an X by Y reduced matrix and initialize it to zeros.

Step 3: Mark four corners of each rectangle in the reduced matrix. *O(N)*

Step 4: Perform a 2D prefix sum of the reduced matrix and count number of squares that are not zero. *O(N*N)*

Solution 3: Mapping

We can declare the grid as a map of map and then solve the problem as if the grid has been declared as a 2D array and then solve problem with the 2D prefix sum technique. Accessing an element of the grid will take *O(lg(lgN))* time instead of *O(1)*.

```
map<int, map<int, int>> grid;   //instead of vector<vector<int>> grid;
```

Problem 3: Grazing Area

Farmer John's farm is divided into an N x M unit square grid of pastures (1 <=N, M <= 10^6). K (0 <= K <= 1000) of the horizontal or vertical strip of pastures contain rocks. Cows cannot graze in a rocky pasture and cannot pass through it.

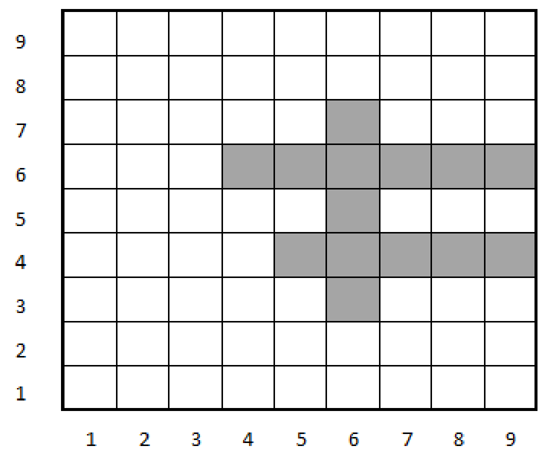
Every morning cows start grazing in the first grid pasture (1, 1). What is the total grazing area for the cows in the farm? Of course, the cows only move horizontally or vertically on the X or Y axis, they never go diagonally.

Sample Input

```
9 9 3
4 6 9 6
6 3 6 7
5 4 9 4
```

Sample Output

```
64
```



Solution:

- Compress the grid at most *O(KxK)* in size and initialize it to zeros.
- Mark the rocky places to ones.
- Use the Flood Fill algorithm to count the empty cells on the compressed grid and calculate their total area on the original grid.

Problem 4: Delivery Route (USACO 2012 January, Silver)

After several years of record milk production, Farmer John now operates an entire network of N farms (1 <= N <= 100). Farm i is located at position (x_i, y_i) in the 2D plane, distinct from all other farms, with both x_i and y_i being integers.

FJ needs your help planning his daily delivery route to bring supplies to the N farms. Starting from farm 1, he plans to visit the farms sequentially (farm 1, then farm 2, then farm 3, etc.), finally returning to farm 1 after visiting farm N. It tames FJ one minute to make a single step either north, south, east, or west. Furthermore, FJ wants to visit each farm exactly once during his entire journey (except farm 1, which he visits twice of course).

Please help FJ determine the minimum amount of time it will take him to complete his entire delivery route.

PROBLEM NAME: delivery

INPUT FORMAT:

* Line 1: The number of farms, N.

* Lines 2..1+N: Line i+1 contains two space-separated integers, x_i and y_i (1 <= x_i, y_i <= 1,000,000).

SAMPLE INPUT (file delivery.in):

```
4
2 2
2 4
2 1
1 3
```

OUTPUT FORMAT:

* Line 1: The minimum number of minutes FJ needs to complete his delivery route, or -1 if it is impossible to find a feasible delivery route that visits every farm exactly once (except farm 1).

SAMPLE OUTPUT (file delivery.out):

```
12
```

OUTPUT DETAILS:

FJ can complete his delivery route in 12 minutes: 2 minutes to go from farm 1 to farm 2, 5 minutes to go from farm 2 to farm 3 (circumventing farm 1), 3 minutes to go from farm 3 to farm 4, and then 2 minutes to return to farm 1.

Problem Credits: Brian Dean

Useful Links and References

https://www.hackerearth.com/practice/math/geometry/line-sweep-technique/tutorial/
https://www.educative.io/edpresso/the-skyline-problem-in-cpp
https://www.sciencedirect.com/topics/earth-and-planetary-sciences/convex-hull
https://www.cs.auckland.ac.nz/software/AlgAnim/convex_hull.html