



Сливане на интервали

locked

Problem

Submissions

Leaderboard

Discussions

На входа получавате N на брой интервала. Трябва да слеете всички възможни интервали и да изведете слетите интервали подредени така както биха били подредени върху числовата ос. Интервалите са затворени, т.е ако има интервал $[0, 25]$ и $[25, 100]$ то те може да се слят защото имат обща точка 25.

Input Format

На първият ред получавате число N - броя на интервалите. На следващите N реда получавате по 2 числа - a b , където a е начална точка на интервала, а b - крайна точка на интервала, разделени с пауза. Винаги $a \leq b$.

Constraints

$$1 \leq N \leq 10^6$$

$$-10^9 \leq a \leq b \leq 10^9$$

Output Format

На K реда изведете слетите интервали в нарастващ ред в същия вид както интервалите са подавани на входа, т.е начало на интервал, пауза, край на интервал. K е броят на интервалите след сливането.

Sample Input 0

```
2
5 8
8 10
```

Sample Output 0

```
5 10
```

Explanation 0

$[5,8]$ и $[8,10]$ имат обща точка - 8. Следователно двата интервала могат да бъдат слети до $[5,10]$

Sample Input 1

```
5
0 5
4 8
12 12
13 18
14 16
```

Sample Output 1

```
0 8
12 12
13 18
```

Explanation 1

[0,5] и [4,8] имат обща част - [4,5], следователно могат да се слоят до [0,8]. [12,12] няма обща част с никой друг интервал, следователно не се променя. [13,18] напълно обхваща [14,16], така че може да се обединят в [13,18].

[f](#) [t](#) [in](#)



Submissions: 105

Max Score: 100

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)Current Buffer (saved locally, editable)  

C++14



```
1 #include <cmath>
2 #include <cstdio>
3 #include <vector>
4 #include <iostream>
5 #include <algorithm>
6 using namespace std;
7
8 int leftArr[500000];
9 int rightArr[500000];
10
11 int newLeft[500000];
12 int newRight[500000];
13
14 int compare(const void* a, const void* b)
15 {
16     const int* x = (int*) a;
17     const int* y = (int*) b;
18
19     if (*x > *y)
20         return 1;
21     else if (*x < *y)
22         return -1;
23
24     return 0;
25 }
26
27 int main() {
28     int n;
29     cin >> n;
30
31     for (int i = 0; i < n; i++)
32         cin >> leftArr[i] >> rightArr[i];
33
34     //sort intervals with respect to left bound
35     mergeSort(leftArr, rightArr, 0, n - 1);
36
37     int k = 0; //counter for result arrays
38     int j = 1; //counter to iterate over all elements
39
40     newLeft[0] = leftArr[0];
41     newRight[0] = rightArr[0]; //setting result arrs with first interval
42
43     while (j < n)
44     {
45         //if next interval is entirely in the previous interval: [3 13] and [5 10]
46         if (rightArr[j] <= newRight[k]) //left bound is for sure >= current left bound
47         {
48             j++;
49             continue;
50         }
51
52         //if next left is <= current right but next right is > current right: [5 10] and [8 12]
53         if (leftArr[j] <= newRight[k] && rightArr[j] > newRight[k])
54         {
55             newRight[k] = rightArr[j]; //change current right bound to the bigger value
56             j++;
57             continue;
58         }
59     }
```

```
58     }
59
60     //if next left is > current right - make new interval: [8 12] and [20 25]
61     if (leftArr[j] > newRight[k])
62     {
63         k++;
64         newLeft[k] = leftArr[j];
65         newRight[k] = rightArr[j];
66         j++;
67     }
68 }
69
70 for (int i = 0; i<=k; i++)
71     cout << newLeft[i] << ' ' << newRight[i] << endl;
72 return 0;
73 }
74
```

Line: 1 Col: 1

 [Upload Code as File](#) ☐ [Test against custom input](#)[Run Code](#)[Submit Code](#)