题目描述

绘图机器的绘图笔初始位置在原点(0,0)机器启动后按照以下规则来进行绘制直线。

1. 尝试沿着横线坐标正向绘制直线直到给定的终点E

2. 期间可以通过指令在纵坐标轴方向进行偏移，offsetY为正数表示正向偏移,为负数表示负向偏移

给定的横坐标终点值E 以及若干条绘制指令，

请计算绘制的直线和横坐标轴以及x=E的直线组成的图形面积。

输入描述

- 首行为两个整数 N 和 E
- 表示有N条指令,机器运行的横坐标终点值E
- 接下来N行 每行两个整数表示一条绘制指令**x offsetY**
- 用例保证横坐标x以递增排序的方式出现
- 且不会出现相同横坐标x

取值范围

- 0<N<=10000
- 0<=x<=E<=20000
- -10000<=offsetY<=10000

输出描述

- 一个整数表示计算得到的面积 用例保证结果范围在0到4294967295之内。

用例

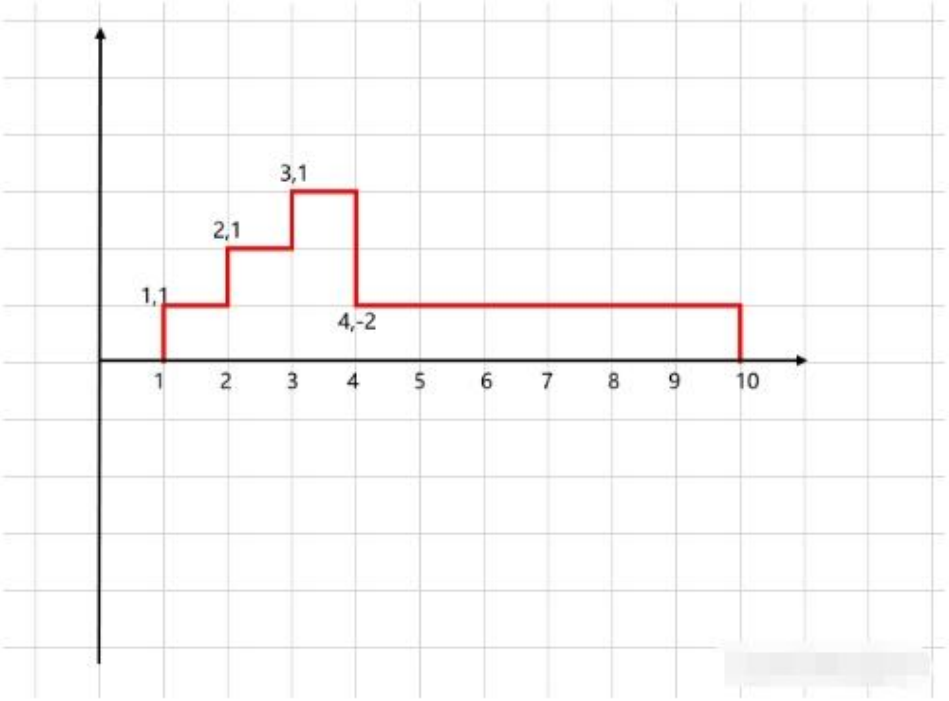
输入	4 10 1 1 2 1 3 1 4 -2
输出	12
说明	无

输入	2 4 0 1 2 -2
输出	4
说明	无

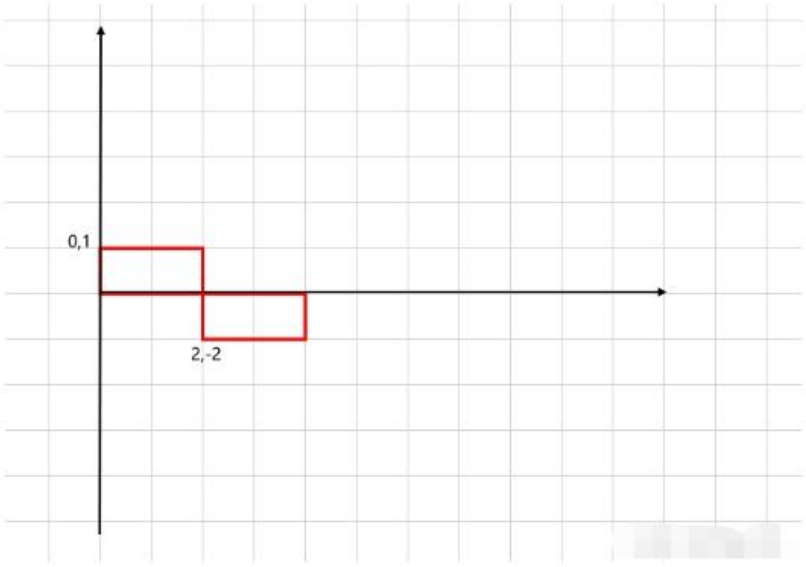
题目解析

注意下面每个拐点上标记不是坐标信息，而是 (x,offsetY)，其中offsetY是偏移

示例1图示



示例2图示



这题将图画出来后，可能大家的思路就打开了。

我的解题思路是这样的，将上面红色线框对应的复杂图形的面积求解，切割为横轴上每个单位长度的矩形面积求解，而每单位长度的矩形面积就等于对应的高度，即纵轴坐标的绝对值，因此我们只需要将offsetY偏移转为纵坐标的即可。

而题目描述中：用例保证横坐标x以递增排序的方式出现。

这里只强调递增没有强调连续，因此我们需要考虑不连续的offsetY转纵坐标的场景，其实也很简单，断档的offsetY其实默认集成前面的offsetY，这其实就是动态规划的状态转移。

Java算法源码

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();
    int e = sc.nextInt();

    // 求出每个横轴单位上的offsetY偏移值，如果输入未给定offsetY，则相当于offsetY=0
    int[] offsets = new int[e];
    for (int i = 0; i < n; i++) {
      int x = sc.nextInt();
      int offsetY = sc.nextInt();
      offsets[x] = offsetY;
    }

    System.out.println(getResult(n, e, offsets));
  }

  public static int getResult(int n, int e, int[] offsets) {
    if (e == 0) return 0;

    int[] dp = new int[e];

    int ans = dp[0] = offsets[0];
    for (int i = 1; i < offsets.length; i++) {
      dp[i] = dp[i - 1] + offsets[i];
      ans += Math.abs(dp[i]);
    }

    return ans;
  }
}
```

JS算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
let n, e;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    [n, e] = lines[0].split(" ").map(Number);
  }

  if (n && lines.length === n + 1) {
    lines.shift();

    // 求出每个横轴单位上的offsetY偏移值，如果输入未给定offsetY，则相当于offsetY=0
    const offsets = new Array(e).fill(0);
    lines.forEach((line) => {
      let [x, offsetY] = line.split(" ").map(Number);
      offsets[x] = offsetY;
    });

    console.log(getResult(e, offsets));

    lines.length = 0;
  }
});

function getResult(e, offsets) {
  let dp = new Array(e).fill(0);

  let ans = (dp[0] = offsets[0]);

  for (let i = 1; i < offsets.length; i++) {
    dp[i] = offsets[i] + dp[i - 1];
    ans += Math.abs(dp[i]);
  }

  return ans;
}
```

Python算法源码

```python
# 输入获取
n, e = map(int, input().split())
tmp = [list(map(int, input().split())) for _ in range(n)]

offsets = [0]*e
for x, offsetY in tmp:
    offsets[x] = offsetY


# 算法入口
def getResult():
    if e == 0:
        return 0

    dp = [0]*e

    ans = dp[0] = offsets[0]
    for i in range(1, e):
        dp[i] = dp[i-1] + offsets[i]
        ans += abs(dp[i])

    return ans


# 算法调用
print(getResult())
```