Curtain 別解

By tokoharu

@tokoharu_sakura(鍵)

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef pair<int,int> PII;
const int MX = 200000;
typedef long long LL;
int main() {
    int N;
    while(cin >> N, N) {
        vector<PII> input(N);
        for(int i=0; i<N; i++) {
            cin >> input[i].first >> input[i].second;
        }
        LL ymin = MX, ymax = -MX, xmin = MX, xmax = -MX;
        for(int i=0; i<4; i++) {
            LL x,y;
            cin >> x >> y;
            ymin = min(ymin, y);
            ymax = max(ymax, y);
            xmin = min(xmin, x);
            xmax = max(xmax, x);
        }
        LL ans1 = 0, ans2 = 0;
        input.push_back(input[0]);
        for(int i=0; i<N; i++) {
            if(input[i].second == input[i+1].second) {
                LL x1 = input[i].first;
                LL x2 = input[i+1].first;
                LL y = input[i].second;
                ans2 += y * (x1 - x2);

                x1 = max(xmin, min(xmax, x1));
                x2 = max(xmin, min(xmax, x2));
                y  = max(ymin, min(ymax, y));
                ans1 += y * (x1 - x2);
            }
        }
        cout << ans2 - ans1 << endl;
    }

    return 0;
}
```

別解

- https://ideone.com/ac7mZX
- みじかい！

- 一方、「なんだこれ」との声
- どういうことなのかを概説します

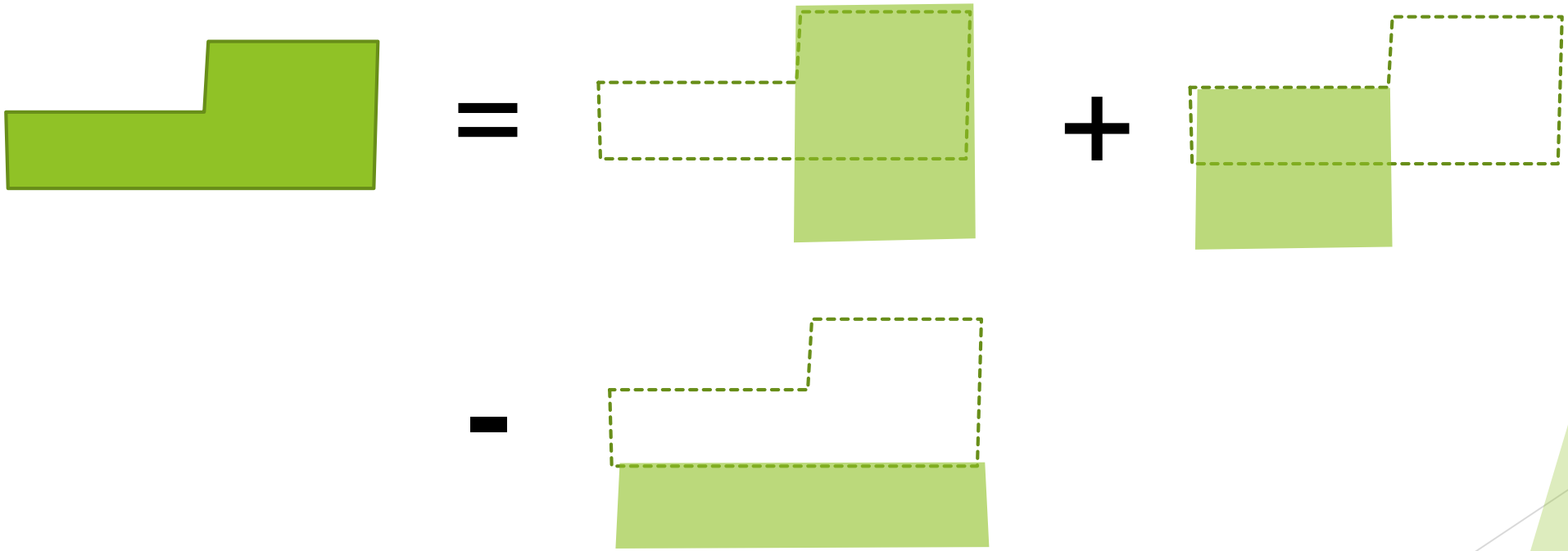
- 本番中この解法で解いたチームはいなさそうに見えました
  - すべてのコードを確認してはいません

多角形の面積を求めるアルゴリズム(1)

- 図形を足し引きしてうまいこと面積を求めるやり方が有名
  - http://imagingsolution.net/math/calc_n_point_area/
  - https://ja.wikipedia.org/wiki/%E5%BA%A7%E6%A8%99%E6%B3%95
  - 座標法と呼ばれるらしい
  - 原点と図形の辺からなる三角形の面積を足し引きする

多角形の面積を求めるアルゴリズム(2)

- 同様の手法で、原点に替えてx軸と図形の辺を見ていくやり方がある
- 次のイメージ



- 今回の問題ではこちらのほうが都合がよさそうなのでこれを基に考える

```cpp
// O(N) solution for problem C of 3rd Domestic D
#include <bits/stdc++.h>
using namespace std;
typedef pair<int,int> PII;
const int MX = 200000;
typedef long long LL;
int main() {
    int N;
    while(cin >> N, N) {
        vector<PII> input(N);
        for(int i=0; i<N; i++) {
            cin >> input[i].first >> input[i].second;
        }
        LL ymin = MX, ymax = -MX, xmin = MX, xmax = -MX;
        for(int i=0; i<4; i++) {
            LL x,y;
            cin >> x >> y;
            ymin = min(ymin, y);
            ymax = max(ymax, y);
            xmin = min(xmin, x);
            xmax = max(xmax, x);
        }
        LL ans1 = 0, ans2 = 0;
        input.push_back(input[0]);
        for(int i=0; i<N; i++) {
            if(input[i].second == input[i+1].second) {
                LL x1 = input[i].first;
                LL x2 = input[i+1].first;
                LL y = input[i].second;
                ans2 += y * (x1 - x2);

                x1 = max(xmin, min(xmax, x1));
                x2 = max(xmin, min(xmax, x2));
                y  = max(ymin, min(ymax, y));
                ans1 += y * (x1 - x2);
            }
        }
        cout << ans2 - ans1 << endl;
    }

    return 0;
}
```

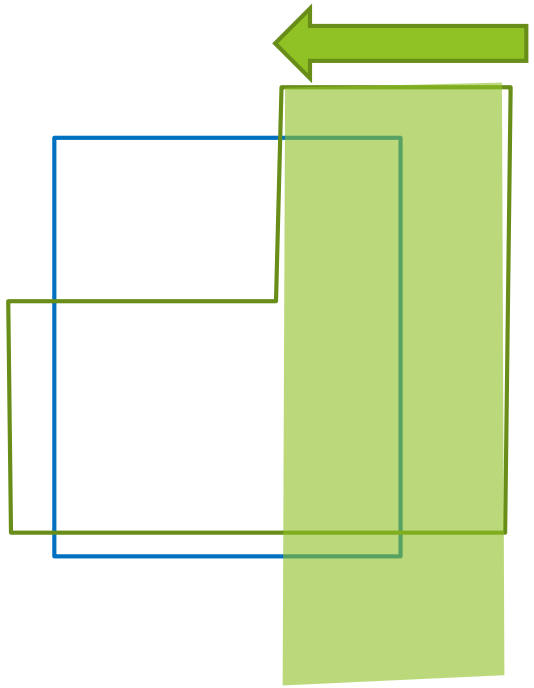
プログラム再掲

- ans2で窓の面積を求めている
- 前頁のx軸から見た時のアルゴリズムを利用していることがわかる

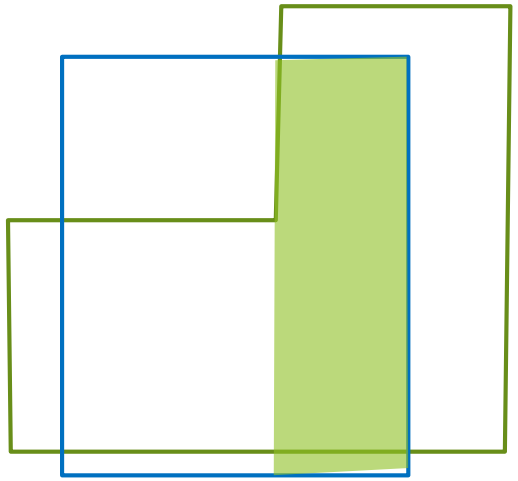
- であるなら、ans1は窓とカーテンの共通部分になるはず
- これをどうやって求めているのか

考え方

通常ここを足しこむところ、



今回このくらい足したい



- 青がカーテンの長方形をあらわす
- 共通部分を足しこみたいときは すべての座標をカーテン内に持ってくるようにうまく調整すればよさそう

具体的にやること

- 前頁の調整はx座標を 次の区間[(カーテンのx座標の最小値), (カーテンのx座標の最大値)] に持ってくるとよい
- y座標も同様
- この操作をしてから多角形の面積を求めればよい
- これがans1を求めるときの操作になる

- 注意：
- 「カーテンは長方形のまま、窓の形状が一般の多角形」になったときはこのままの適用ができない。
- 台形の一部の面積を求めるような操作が必要になる