Ideal city

Leonardo, ca mulți alți oameni de știință și artiști italieni de vârsta lui, era foarte interesat de planificarea oraşelor si urbanism. El dorea să modeleze un oraş ideal: confortabil, spațios, şi rațional în privința utilizării resurselor, mult diferite de oraşele îngustate şi claustrofobice din evul mediu.

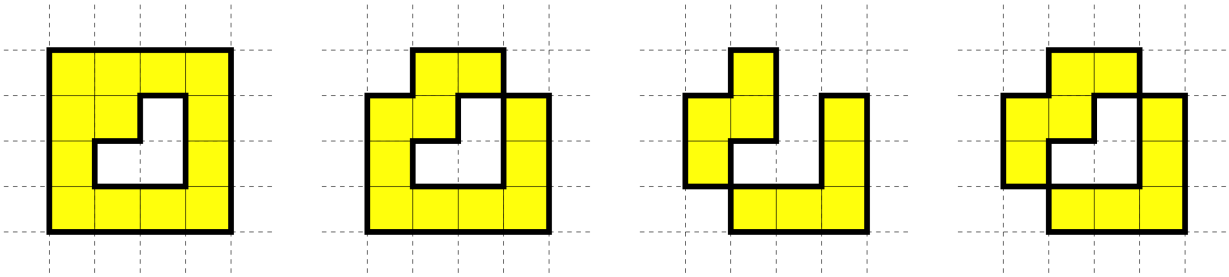
Oraşul ideal

Oraşul era format din N blocuri plasate pe un grid infinit de formă pătrată format din celule. Fiecare celulă se identifică prin perechea de coordonate (rând, coloană). Celula (0, 0) este în colțul din stânga sus a gridului. Pentru celula (i, j), celulele adiacente (dacă există) sunt: (i - 1, j), (i + 1, j), (i, j - 1), şi (i, j + 1). Fiecare block, când este plasat pe grid, acoperă exact o celulă. Un bloc poate fi plasat în celula (i, j) dacă şi numai dacă $1 \leq i, j \leq 2^{31} - 2$. Vom folosi, de asemenea, coordonatele celulelor pentru a ne referi la blocurile plasate pe ele. Două blocuri sunt adiacente dacă sunt plasate pe celule adiacente. Într-un oraş ideal, toate blocurile sale sunt conectate astfel încât nu există "găuri" în interiorul frontierei, adică celulele trebuie să îndeplinească următoarele două condiții.

- Pentru oricare două celule *neacoperite* , există cel puțin o secvență de celule *neacoperite* adiacente care le conectează.
- Pentru oricare două celule *acoperite*, există cel puțin o secvență de celule *acoperite* adiacente care le conectează.

Exemplul 1

Nici una din configurațiile de blocuri de mai jos nu reprezintă un oraș ideal: primele două de la stânga nu respectă prima condiție, a treia nu respectă a doua condiție, iar a patra nu respectă nici una dintre condiții.



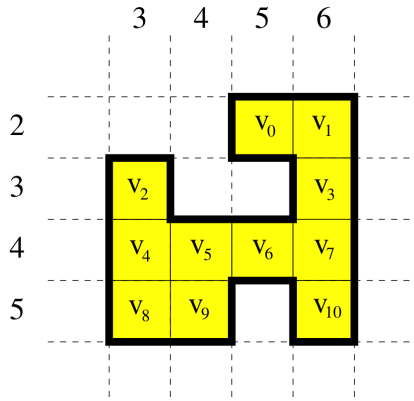
Distanța

Când traversăm oraşul, o *săritură* înseamnă trecerea de la un bloc la un alt bloc adiacent. Celulele neacoperite nu pot fi traversate. Fie v_0, v_1, …, v_{N-1} coordonatele a N blocuri plasate pe grid. Pentru

oricare două blocuri distincte aflate la coordonatele v_i și v_j, distanța dintre $d(v_i, v_j)$ este cel mai mic număr de salturi necesare pentru a ajunge de la un bloc la celălalt.

Exemplul 2

Configurația de mai jos reprezintă un oraș ideal format din N = 11 blocuri la coordonatele v_0 = (2, 5), v_1 = (2, 6), v_2 = (3, 3), v_3 = (3, 6), v_4 = (4, 3), v_5 = (4, 4), v_6 = (4, 5), v_7 = (4, 6), v_8 = (5, 3), v_9 = (5, 4), și v_{10} = (5, 6). De exemplu, $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, și $d(v_9, v_{10}) = 4$.



Enunț

Sarcina ta este să scrii un program care, primind un oraș ideal, să calculeze suma distanțelor dintre oricare perechi de blocuri v_i și v_j cu i < j. Mai exact, programul tău trebuie să calculeze următoarea sumă:

$$\sum d(v_i, v_j), \text{ unde } 0 \le i < j \le N - 1$$

Mai exact, trebuie să implementezi o rutină `DistanceSum`(N, X, Y) care, primind N și doi vectori X și Y care descriu orașul, calculează formula de mai sus. Atât X și Y conțin exact N elemente; blocul i se află la coordonatele (x[i], Y[i]) cu $0 \le i \le N - 1$, și $1 \le X[i], Y[i] \le 2^{31} - 2$. Deoarece rezultatul poate fi prea mare pentru a fi reprezentat pe 32 de biți, el trebuie calculat modulo 1 000 000 000 (un miliard).

În exemplul 2, sunt 11 × 10 / 2 = 55 perechi de blocuri. Suma distanțelor dintre toate perechile este 174.

Subtask 1 [11 puncte]

Se consideră că $N \le 200$.

Subtask 2 [21 de puncte]

Se consideră că $N \le 2\,000$.

Subtask 3 [23 de puncte]

Se consideră că $N \leq 100\,000$.

În plus, următoarele două condiții se îndeplinesc: pentru două celule acoperite i și j astfel încât X[i] = X[j], oricare celulă dintre ele este acoperită; pentru două celul acoperite i și j astfel încât Y[i] = Y[j], oricare celulă dintre ele este acoperită;

Subtask 4 [45 de puncte]

Se consideră că $N \leq 100\,000$.

Detalii de implementare

Trebuie să trimiți exact un fișier, numit `city.c`, `city.cpp` sau `city.pas`. Acest fișier trebuie să implementeze subprogramul descris mai sus având următoarele semnături.

Programe C/C++

```
int DistanceSum(int N, int *X, int *Y);
```

Programe Pascal

```
function DistanceSum(N : LongInt; var X, Y : array of LongInt) : LongInt;
```

Acest subprogram trebuie să se comporte cum a fost descris mai sus. Desigur poți implementa alte subprograme pentru uz intern. Submisia ta nu trebuie să interacționeze în vre-un fel cu intrarea/ieșirea standard, și nici cu alte fișiere.

Exemplu de evaluator

Exemplul de evaluator furnizat așteaptă inputul în următorul format:

- linia 1: N;
- liniile 2, …, N + 1: X[i], Y[i].

Limite de timp și memorie

- Limită de timp: o secundă.
- Limită de memorie: 256 MB.