Idealni grad

Leonardo, kao i mnogi drugi italijanski naučnici i umjetnici njegovog doba, bio je izuzetno zainteresovan za planiranje gradova i urbani dizajn. Ciljao je da modelira idealan grad: udoban, prostran i racionalan u korištenju svojih resursa, daleko od uskih, klaustrofobičnih gradova srednjeg vijeka.

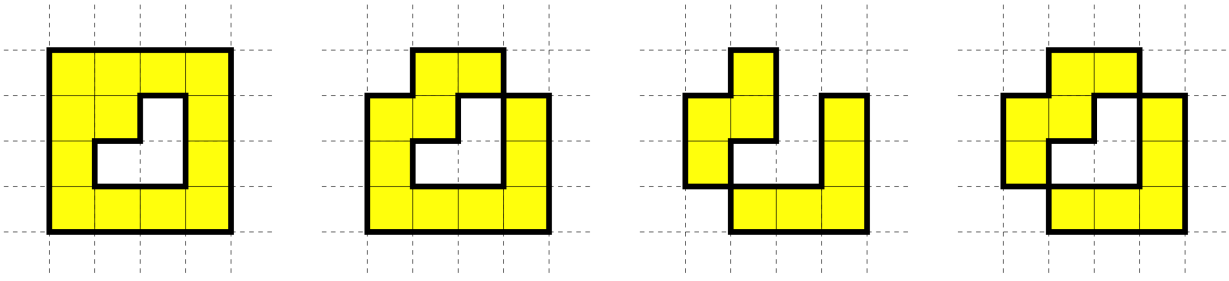
Idealni grad

Grad se sastoji od N blokova postavljenih na zamišljenu beskonačnu kvadratnu mrežu ćelija. Svaku ćeliju identifikuje par koordinata (red, kolona). Za datu ćeliju (i, j), susjedne ćelije su: (i - 1, j), (i + 1, j), (i, j - 1), i (i, j + 1). Svaki blok, kada se postavi na mrežu, pokriva tačno jednu ćeliju. Blok može biti postavljen na ćeliju (i, j) ako i samo ako $1 \leq i, j \leq 2^{31} - 2$. Koristićemo koordinate ćelija i kada mislimo na blokove na njima. Dva bloka su susjedna ako su postavljeni na susjedne ćelije. U idealnom gradu, svi njegovi blokovi su povezani na takav način da nema "rupa" unutar njegovih granica, odnosno, ćelije moraju da zadovolje oba uslova navedena ispod.

- Za svake dvije *prazne* ćelije, postoji bar jedan niz susjednih *praznih* ćelija koje ih povezuju.
- Za svake dvije *neprazne* ćelije, postoji bar jedan niz susjednih *nepraznih* ćelija koje ih povezuju.

Primjer 1

Ni jedna konfiguracija blokova ispod ne predstavlja idealan grad: prve dvije lijevo ne zadovoljavaju prvi uslov, treća ne zadovoljava drugi uslov, a četvrta ne zadovoljava ni jedan od uslova.



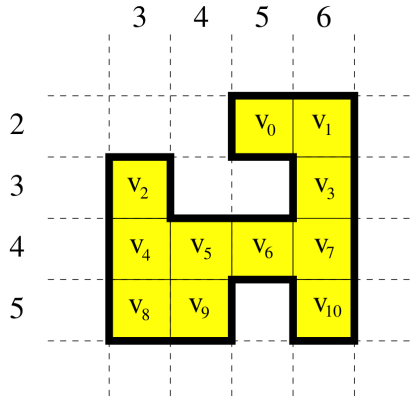
Razdaljina

Kada se kreće kroz grad, *skok* predstavlja kretanje od nekog bloka do nekog drugog bloka susjednog njemu. Preko praznih ćelija se ne može kretati. Neka su $v_0, v_1, \ldots, v_{N-1}$ koordinate N blokova postavljenih na mrežu. Za bilo koja dva različita bloka na koordinatama v_i i v_j, njihova razdaljina $d(v_i, v_j)$ je najkraći broj skokova koji su potrebni da se dođe od jednog od ovih blokova

do drugog.

Primjer 2

Konfiguracija ispod predstavlja idealan grad sastavljen od N = 11 blokova na koordinatama 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), and v_{10} = (5, 6). Na primjer, d(v_1, v_3) = 1, d(v_1, v_8) = 6, d(v_6, v_{10}) = 2, i d(v_9, v_{10}) = 4.



Postavka

Vaš zadatak je da napišete program koji, za idealni grad, računa sumu svih razdaljina između blokova v_i i v_j, po parovima, za koje važi i < j. Formalno, Vaš program treba da izračuna vrijednost sljedeće sume:

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

Specifično, treba da implementirate funkciju `DistanceSum`(N, X, Y) koja, kada je dato N i dva niza X i Y koji opisuju grad, računa formulu iznad. I X i Y su veličine N; blok i je na koordinatama (X[i], Y[i]) za $0 \le i \le N - 1$, i $1 \le X[i], Y[i] \le 2^{31} - 2$. Kako rezultat može biti prevelik da bi se predstavio sa 32 bita, trebate ga objaviti po modulu 1 000 000 000 (milijardu).

U primjeru 2, postoji 11 × 10 / 2 = 55 parova blokova. Suma svih razdaljina, po parovima, je 174.

Podzadatak 1 [11 bodova]

Možete smatrati da je $N \le 200$.

Podzadatak 2 [21 bod]

Možete smatrati da je $N \le 2\,000$.

Podzadatak 3 [23 boda]

Možete smatrati da je $N \le 100\,000$.

Dodatno, sljedeća dva uslova važe: ako su date dvije neprazne ćelije i i j takve da X[i] = X[j], svaka ćelija između njih je neprazna; ako su date dvije neprazne ćelije i i j takve da Y[i] = Y[j], svaka ćelija između njih je takođe neprazna.

Podzadatak 4 [45 bodova]

Možete smatrati da je $N \leq 100\,000$.

Detalji implementacije

Trebate poslati tačno jednu datoteku, pod nazivom `city.c`, `city.cpp` ili `city.pas`. Ova datoteka treba da implementira opisani potprogram sa sljedećim deklaracijama.

C/C++ programi

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

Pascal programi

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

Ova funkcija treba da se ponaša u skladu sa ranije datim opisima. Naravno, možete implementirati i druge pomoćne funkcije. Vaš program ne smije ni na koji način koristiti standardni ulaz i izlaz niti bilo koju drugu datoteku.

Probni tester

Probni tester dostavljen sa takmičarskim okruženjem očekuje ulazne podatke u sljedećem formatu:

- linija 1: N;
- linije 2, …, N + 1: X[i], Y[i].

Vremensko i memorijsko ograničenje

- Vremensko ograničenje: 1 sekunda.
- Memorijsko ograničenje: 256 MiB.