Idealan grad

Boris, kao i mnogi drugi italijanski naučnici i umetnici njegovog doba, bio je izuzetno zainteresovan za planiranje gradova i urbani dizajn (suprotno od njegovog brata ratara Demjana koji je bio zainteresovan samo za obradu njiva). Njegov san je bio da modelira idealan grad: udoban, prostran i racionalan u korišćenju svojih resursa, daleko od uskih i klaustrofobičnih gradova srednjeg veka.

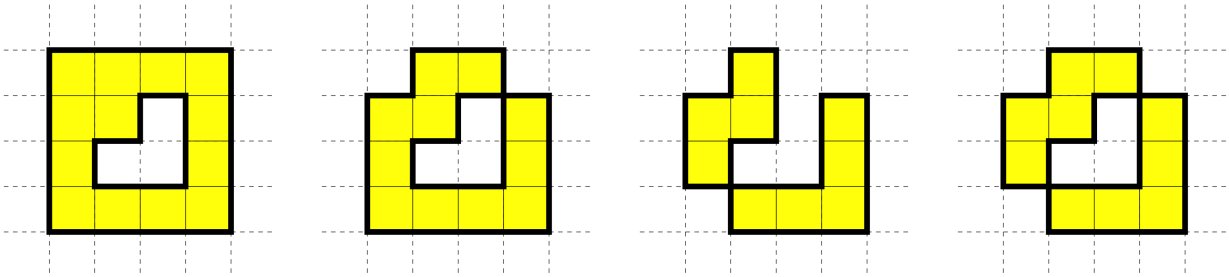
Idealan grad

Grad se sastoji od N blokova postavljenih na beskonačnu kvadratnu mrežu polja. Svako polje je određeno parom koordinata (red, kolona). Za dato polje (i, j), njegova susedna polja su (ako postoje): (i - 1, j), (i + 1, j), (i, j - 1), i (i, j + 1). Svaki blok, kada se postavi na mrežu, pokriva tačno jedno polje. Blok može biti postavljen na polje (i, j) ako i samo ako je $1 \leq i, j \leq 2^{31} - 2$. Koristićemo koordinate polja i kada mislimo na blokove koji su postavljeni na njima. Dva bloka su susedna ako su postavljena na susedna polja. U idealnom gradu, svi njegovi blokovi su povezani na takav način da nema "rupa" unutar njegovih granica, odnosno polja moraju da zadovolje oba uslova navedena ispod.

- Za svaka dva *prazna* polja, postoji bar jedan niz susednih *praznih* polja koji ih povezuju.
- Za svaka dva *neprazna* polja, postoji bar jedan niz susednih *nepraznih* polja koji ih povezuju.

Primer 1

Nijedna konfiguracija blokova ispod ne predstavlja idealan grad: prve dve levo ne zadovoljavaju prvi uslov, treća ne zadovoljava drugi uslov, a četvrta ne zadovoljava ni jedan od uslova.



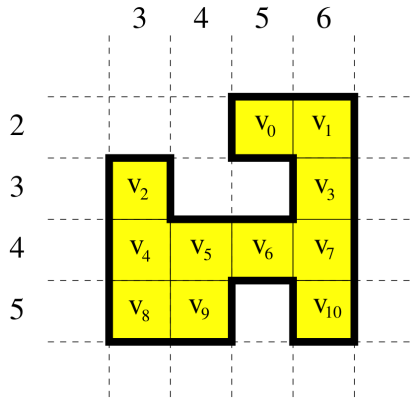
Rastojanja

Kada se krećete kroz grad, *korak* predstavlja kretanje od jednog bloka do nekog drugog susednog bloka. Preko praznih polja se ne može kretati. Neka su v_0, v_1, …, v_{N-1} koordinate N blokova

postavljenih na mrežu. Za bilo koja dva različita bloka na koordinatama v_i i v_j, njihovo rastojanje $d(v_i, v_j)$ je najkraći broj koraka koji su potrebni da se dođe od jednog od ovih blokova do drugog.

Primer 2

Konfiguracija ispod predstavlja idealan grad sastavljen od N = 11 blokova sa koordinatama $v_0 = (2, 5)$, $v_1 = (2, 6)$, $v_2 = (3, 3)$, $v_3 = (3, 6)$, $v_4 = (4, 3)$, $v_5 = (4, 4)$, $v_6 = (4, 5)$, $v_7 = (4, 6)$, $v_8 = (5, 3)$, $v_9 = (5, 4)$, i $v_{10} = (5, 6)$. Na primer, $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, i $d(v_9, v_{10}) = 4$.



Postavka problema

Vaš zadatak je da napišete program koji za dati idealan grad računa sumu svih razdaljina između blokova v_i i v_j, po svim mogućim parovima indeksa za koje važi $i < j$. Formalno, Vaš program treba da izračuna vrednost sledeće sume:

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

Potrebno je implementirati funkciju `DistanceSum`(N, X, Y) koja za dato N i dva niza X i Y koji opisuju grad, računa gornju formulu. Oba niza X i Y su veličine N; blok i je na koordinatama (X[i], Y[i]) za $0 \leq i \leq N - 1$, i $1 \leq X[i], Y[i] \leq 2^{31} - 2$. Kako rezultat može biti prevelik da bi se predstavio sa 32 bita, potrebno ga je izračunati po modulu 1 000 000 000 (jedna milijarda).

U Primeru 2, postoji $11 \times 10 / 2 = 55$ parova blokova. Suma svih rastojanja po parovima je 174.

Podzadatak 1 [11 bodova]

Možete pretpostaviti da je $N \leq 200$.

Podzadatak 2 [21 bodova]

Možete pretpostaviti da je $N \leq 2\,000$.

Podzadatak [23 bodova]

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

Dodatno važe sledeća dva uslova: ako su data dva neprazna polja i i j takva da je X[i] = X[j], svako polje između njih je neprazno; ako su data dva neprazna polja i i j takva da Y[i] = Y[j], svako polje između njih je takođe neprazno.

Podzadatak 4 [45 bodova]

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

Detalji implementacije

Morate predati tačno jedan fajl, pod nazivom `city.c`, `city.cpp` ili `city.pas`. Ovaj fajl mora da sadržati implementaciju gore opisane funkcije koristeći sledeća zaglavlja:

C/C++ programi

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

Pascal programi

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

Ove funkcije se moraju ponašati tačno kako je gore opisano. Naravno, možete implementirati i druge funkcije. Vaš program koji predate ne sme ni na koji način da koristi standardni ulaz/izlaz, niti bilo koji drugi fajl.

Program za testiranje (sample grader)

Program za testiranje koji se nalazi u okviru zadatka očekuje ulaz u sledećem formatu:

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

Vremenska i memorijska ograničenja

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