Idealni grad

Leonardo je bio jako zainteresiran za urbanističke probleme svog vremena i pokušao je dizajnirati idealan grad.

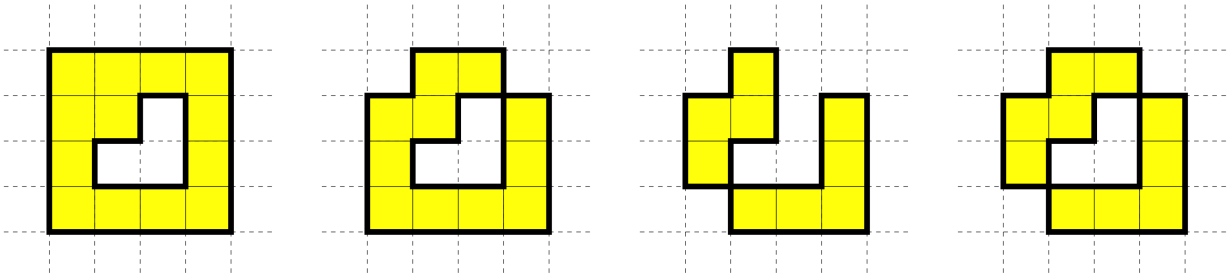
Grad

Grad se sastoji od N blokova koji se nalaze na beskonačnoj mreži polja. Svako polje je određeno koordinatama (redak, stupac). Za svako polje (i, j), susjedna polja su: (i - 1, j), (i + 1, j), (i, j - 1), i (i, j + 1). Svaki blok na mreži zauzima točno jedno polje. Blok se može nalaziti na polju (i, j) jedino ako je $1 \leq i, j \leq 2^{31} - 2$. Koordinate polja ćemo također koristiti i za koordinate blokova koji su na njima. Dva bloka su susjedna ako su smješteni na susjednim poljima. U idealnom gradu, svi blokovi su međusobno povezani na način da ne postoje "rupe" unutar blokova tj. polja moraju zadovoljavati sljedeće uvjete.

- Za svaka dva *prazna* polja, postoji barem jedan niz susjednih *praznih* polja kojih ih povezuje.
- Za svaka dva *ne-prazna* polja, postoji barem jedan niz susjednih *ne-praznih* polja koji ih povezuje.

Prvi primjer

Niti jedna od dolje prikazanih konfiguracija ne predstavlja idealan grad: prvi i drugi ne zadovoljavaju prvi uvjet, treći ne zadovoljava drugi uvjet, a četvrti ne zadovoljava niti jedan uvjet.

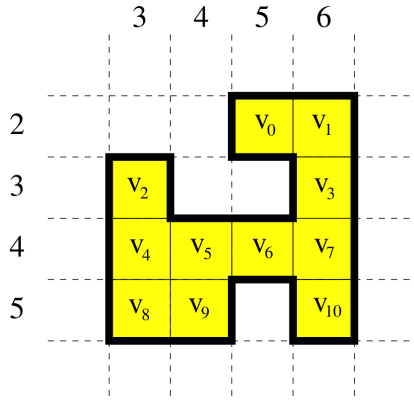


Udaljenost

Prilikom putovanja gradom, *korak* predstavlja pomak iz jednog bloka u drugi susjedni blok. Ne može se putovati praznim poljima. Neka su $v_0, v_1, \ldots, v_{N-1}$ koordinate od N blokova na mreži. Za svaka dva međusobno različita bloka na koordinatama v_i i v_j, njihova udaljenost $d(v_i, v_j)$ je najmanji potreban broj koraka da bi se doputovalo od jednog do drugog bloka.

Drugi primjer

Konfiguracija ispod predstavlja idealan grad koji se sastoji 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), i v_{10} = (5, 6). Primjerice, $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, i $d(v_9, v_{10}) = 4$.



Zadatak

Napišite program koji će izračunati sumu udaljenosti svih međusobno različitih blokova v_i i v_j za koje vrijedi i < j. Preciznije, vaš program mora izračunati sljedeću sumu:

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

Potrebno je napisati funkciju `DistanceSum`(N, X, Y) koja će, za zadani broj N i 2 niza X i Y koji opisuju grad, izračunati gornju formulu. Oba niza, X i Y su veličine N; blok i je na koordinatama (X[i], Y[i]) za sve for $0 \le i \le N - 1$, and $1 \le X[i], Y[i] \le 2^{31} - 2$. Kako rezultat može biti prevelik da bi mogao biti predstavljen varijablom od 32 bits, rezultat mora biti prikazan modulo 1 000 000 000 (milijarda).

In primjeru 2, postoji 11 × 10 / 2 = 55 porova blokova. Tražena suma je 174.

Podzadatak 1 [11 bodova]

- $N \le 200$.

Podzadatak 2 [21 bod]

- $N \le 2\,000$.

Podzadatak 3 [23 boda]

- $N \le 100\,000$.

Dodatno, vrijede sljedeća dva uvjeta: za svaka dva neprazna polja i i j takvi da je X[i] = X[j], svako polje između njih je također neprazno; za svaka dva neprazna polja i i j takvi da je Y[i] = Y[j], svako polje između njih je također neprazno.

Podzadatak 4 [45 bodova]

- $N \leq 100\,000$.

Implementacija

Morate predati točno jednu datoteku, pod nazivom `city.c`, `city.cpp` ili `city.pas`. Datoteka mora implementirati gore opisani potprogram koristeći sljedeće deklaracije.

C/C++ programi

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

Pascal programi

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

Vaš potprogram se mora ponašati kao što je gore opisano. Naravno, možete slobodno implementirati druge pomoćne funkcije. Ne smijete koristiti standardni ulaz ili izlaz, kao i pisanje ili čitanje po bilo kakvim datotekama.

Primjer sustava za testiranje

Sustav za testiranje na raspologanju prima ulaz kako slijedi:

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

Vremenska i memorijska ograničenja

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