Città ideale

Leonardo, come molti altri scienziati ed artisti italiani del suo tempo, si interessava parecchio alla progettazione di città e ambienti urbani. Egli mirava a disegnare una città ideale: confortevole, spaziosa e razionale nell'uso delle risorse, nettamente diversa dalle anguste e claustrofobiche città del Medioevo.

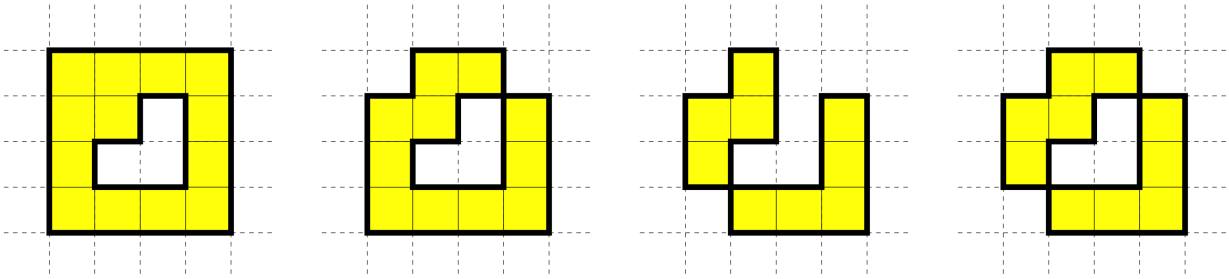
La città ideale

La città è formata da N blocchi posizionati su una griglia infinita di celle. Ogni cella è identificata da una coppia di coordinate (riga, colonna). Data una cella (i, j), le celle ad essa adiacenti sono: (i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1). Ciascun blocco, una volta posizionato sulla griglia, ricopre esattamente una delle celle. Si può posizionare un blocco sulla cella (i, j) se e solo se $1 \leq i, j \leq 2^{31} - 2$. Useremo le coordinate delle celle anche per far riferimento ai blocchi posizionati su di esse. Due blocchi sono adiacenti se sono posizionati in celle adiacenti. In una città ideale, tutti i suoi blocchi sono connessi in modo tale che non vi siano "buchi" all'interno dei suoi confini. Più precisamente, le celle devono soddisfare entrambe le seguenti condizioni.

- Per ogni coppia di celle *vuote*, esiste almeno una sequenza di celle adiacenti *vuote* che le connette.
- Per ogni coppia di celle *non-vuote*, esiste almeno una sequenza di celle adiacenti *non-vuote* che le connette.

Esempio 1

Nessuna delle seguenti configurazioni di blocchi rappresenta una città ideale: le prime due a sinistra non soddisfano la prima condizione, la terza non soddisfa la seconda condizione, e la quarta non soddisfa nessuna delle due condizioni.



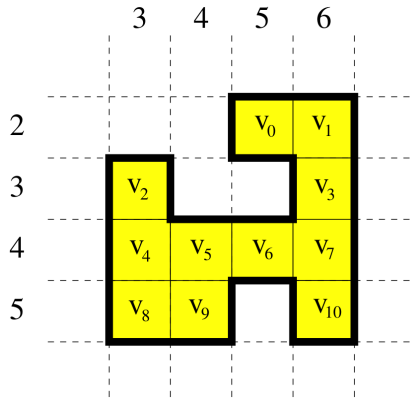
Distanza

Attraversando la città, un *salto* indica l'atto di spostarsi da un blocco a uno adiacente. Le celle vuote

non possono essere attraversate. Siano v_0, v_1, …, v_{N-1} le coordinate degli N blocchi posizionati sulla griglia. Per un'arbitraria coppia di blocchi posizionati alle coordinate v_i e v_j, la loro distanza $d(v_i, v_j)$ è il più piccolo numero di salti necessari per andare da uno di questi blocchi all'altro.

Esempio 2

La seguente configurazione rappresenta una città ideale formata da N = 11 blocchi, posizionati alle coordinate $v_0 = (2, 5)$, $v_1 = (2, 6)$, $v_2 = (3, 3)$, $v_3 = (3, 6)$, $v_4 = (4, 3)$, $v_5 = (4, 4)$, $v_6 = (4, 5)$, $v_7 = (4, 6)$, $v_8 = (5, 3)$, $v_9 = (5, 4)$, e $v_{10} = (5, 6)$. Ad esempio, $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, e $d(v_9, v_{10}) = 4$.



Descrizione del problema

Data una città ideale, il tuo compito è di scrivere un programma che calcoli la somma di tutte le distanze tra coppie di blocchi v_i e v_j per cui $i < j$. Formalmente, il tuo programma deve calcolare il valore della seguente sommatoria:

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

Nello specifico, devi implementare una funzione `DistanceSum`(N, X, Y) che, dati N e due array X ed Y che descrivono la città, calcoli la formula sopra indicata. Sia X che Y hanno dimensione N; il blocco i è posizionato alle coordinate (X[i], Y[i]) per $0 \le i \le N - 1$, e $1 \le X[i], Y[i] \le 2^{31} - 2$. Dal momento che il risultato potrebbe essere troppo grande per essere rappresentato in 32 bit, devi riportarlo modulo 1 000 000 000 (un miliardo).

Nell'Esempio 2, ci sono $11 \times 10 / 2 = 55$ coppie di blocchi. La somma delle distanze fra tutte le coppie di blocchi è 174.

Subtask 1 [11 punti]

Puoi assumere che $N \le 200$.

Subtask 2 [21 punti]

Puoi assumere che $N \le 2\,000$.

Subtask 3 [23 punti]

Puoi assumere che $N \leq 100\,000$.

Inoltre, valgono le seguenti due condizioni: date due qualunque celle non-vuote i e j tali che X[i] = X[j], anche tutte le celle tra esse comprese sono non-vuote; date due qualunque celle non-vuote i e j tali che Y[i] = Y[j], anche tutte le celle tra esse comprese sono non-vuote.

Subtask 4 [45 punti]

Puoi assumere che $N \leq 100\,000$.

Dettagli implementativi

Devi inviare esattamente un file, chiamato `city.c`, `city.cpp` o `city.pas`. Questo file deve implementare la funzione descritta sopra con i seguenti prototipi.

Programmi C/C++

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

Programmi Pascal

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

Questa funzione deve comportarsi come descritto sopra. Ovviamente sei libero di implementare altre funzioni per uso interno. Le tue sottoposizioni non devono interagire in alcun modo con l'input/output standard, né con nessun altro file.

Grader di esempio

Il grader di esempio fornito con l'ambiente del task si aspetta che l'input abbia il seguente formato:

- linea 1: N;
- linee 2, …, N + 1: X[i], Y[i].

Limiti di tempo e memoria

- Limite di tempo: 1 secondo.
- Limite di memoria: 256 MiB.