Ciudad Ideal

Leonardo, al igual que muchos otros cientificos italianos y artistas de su epoca, estuvo muy interesado en la planificacion de ciudades y el diseño urbano. El dirigio el modelo de una ciudad ideal: confortable, espaciosa y con un uso racional de recursos, mucho mas alla de las estrechas y claustrofobicas ciudades de la Edad Media.

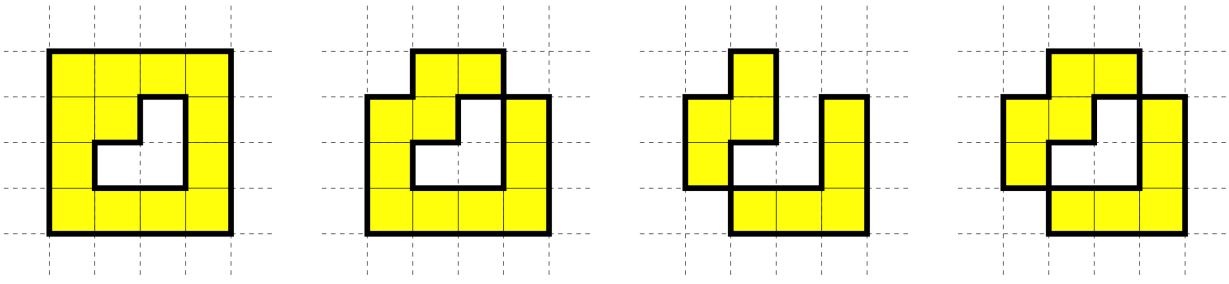
La ciudad ideal

La ciudad esta construida con N bloques localizados en una infinita grilla de celdas. Cada celda es identificada por un par de coordenadas (fila, columna). Tomando una celda (i,j) las celdas adyacentes son: (i - 1, j), (i + 1, j), (i, j - 1), y (i, j + 1). Cada bloque, cuando es puesto dentro de la grilla, cubre exactamente una de las celdas. Un bloque puede ser puesto en la celda (i, j) si y solo si $1 \leq i, j \leq 2^{31} - 2$. Usaremos las coordenadas de las celdas para referirnos tambien a los bloques sobre ellas. Dos bloques son adyacentes si estan localizados en celdas adyacentes. En una ciudad ideal, todos sus bloques estan conectados de forma que no existen "huecos" dentro de sus bordes, esto es, las celdas deben satisfacer las dos condiciones de abajo.

- Para cualquieras dos celdas *vacias*, debe existir al menos una secuencia de celdas adyacentes vacías conectándolos.
- Para dos celdas *no-vacias* cualquieras, debe existir al menos una secuencia de celdas adyacentes *no-vacias* conectándolos.

Ejemplo 1

Ninguna de las configuraciones de bloques abajo dibujadas representa una ciudad ideal: las primeras dos sobre la izquierda no satisfacen la primera condición, la tercera no satisface la segunda condición, y la cuarta no satisface ninguna de las condiciones.

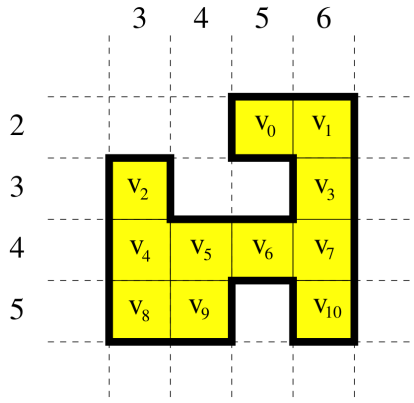


Distancia

Cuando se recorre la ciudad, un *hop* indica ir de un bloque a uno adyacente. Las celdas vacías no pueden ser atravesadas. Sea v_0, v_1, …, v_{N-1} las coordenadas de los N bloques colocados sobre la grilla. Para dos bloques distintos cualesquiera en coordenadas v_i and v_j, su distancia $d(v_i, v_j)$ es el menor número de hops que son requeridos para ir desde uno de estos bloques hacia el otro.

Ejemplo 2

La configuración más abajo representa un ciudad ideal hecha de N = 11 bloques en coordenadas 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), y v_{10} = (5, 6). Por ejemplo, $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, y $d(v_9, v_{10}) = 4$.



Enunciado

Tu tarea consiste en, dada una ciudad ideal, escribir un programa para computar la sum de todas las distancias de a pares entre bloques v_i y v_j para los cuales i < j. Formalmente, tu programa debería computar el valor de la siguiente suma:

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

Especificalmente, tienes que implementar una rutina `DistanceSum`(N, X, Y) que, dados N y dos arreglos X y Y que describe la ciudad, calcuar la formula dada arriba. Ambos X y Y son de tamaño N; el bloque i está en coordenadas (X[i], Y[i]) para $0 \le i \le N - 1$, y $1 \le X[i], Y[i] \le 2^{31} - 2$. Como el resultado puede ser demasiado grande para ser representado usando 32 bits, debes informarlo módulo 1 000 000 000 (mil milliones).

En el Ejemplo 2, hay 11 × 10 / 2 = 55 pares de bloques. La suma de todas las distanciass de a pares es 174.

Subtarea 1 [11 puntos]

Puedes suponer que $N \le 200$.

Subtarea 2 [21 points]

Puedes suponer que $N \leq 2\,000$.

Subtarea 3 [23 points]

Puedes suponer que $N \leq 100\,000$.

Adicionalmente, valen las dos condiciones siguiente: dadas dos celdas no-vacías i y j tal que X[i] = X[j], cada celda entre ellas es también no-vacía; dadas dos celdas no-vacías cualesquiera i y j tal que Y[i] = Y[j], cada celda entre ellas es también no-vacía.

Subtarea 4 [45 points]

Puedes suponer que $N \leq 100\,000$.

Detalles de implementación

Tienes que enviar exactamente un archivo, llamado `city.c`, `city.cpp` o `city.pas`. Este archivo debe implementar el subprograma descripto arriba usando los siguiente encabezamientos.

Programas C/C++

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

Programas Pascal

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

Estos subprograma debe comportarse tal como está descrito arriba. Por supuesto eres libre de implementar otros subprogramas para su uso interno. Tu envío no debe interactuar de ningún modo con la entrada/salida estándar, ni sobre ningún otro archivo.

Evaluador de muestra

El evaluador de muestra provisto en el ambiente de la tarea supondrá una entrada en el siguiente formato:

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

Límites de tiempo y memorias

- Tiempo limite: 1 segundo.
- Memoria limite: 256 MiB.