Ciudad ideal

Leonardo, igual que muchos científicos y artistas de su época, estaba extremadamente interesado en la planificación y diseño urbano. El aspiraba a un modelo de ciudad ideal: confortable, espaciosa y racional en el uso de sus recursos, distante de la estrechas, claustrofóbicas ciudades de la Edad Media.

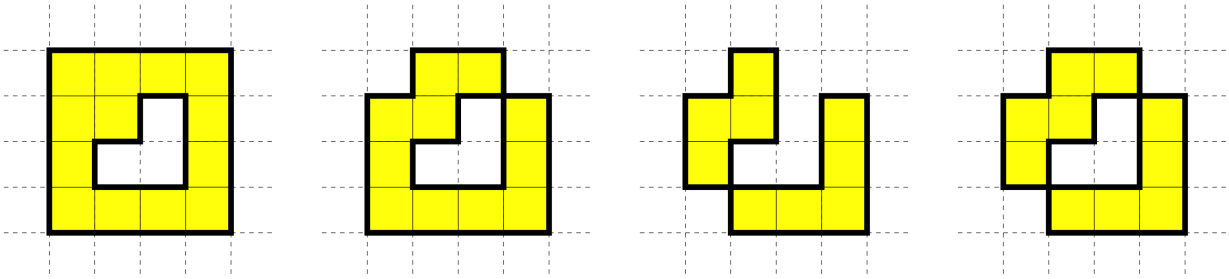
La ciudad ideal

La ciudad está hecha de N bloques colocados en una grilla infinita de celdas imaginarias. Cada celda está identificada por un par de coordenadas (fila, columna). Dada una celda (i, j), las celdas adyacentes son (si ellas existen): (i - 1, j), (i + 1, j), (i, j - 1), y (i, j + 1). Cada bloque, cuando se coloca en la grilla, cubre exactamente una de las celdas. Un bloque puede colocarse en la celda (i, j) sí y solo sí $1 \leq i, j \leq 2^{31} - 2$. Nosotros usamos las coordenadas de las celdas para también hacer referencia a los bloques encima de ella. Dos bloques son adyacentes si ellos están colocados en celdas adyacentes. En una ciudad ideal, todos sus bloques están conectados de tal forma que no hay "huecos" dentro de sus bordes, esto es, las celdas tienen que satisfacer ambas condiciones de abajo.

- Para dos celdas cualquiera *vacías*, existe al menos una secuencia de celdas adycentes *vacías* que la conectan a ella.
- Para dos celdas cualquiera *no vacías*, existe al menos una secuencia de celdas adycentes *no vacías* que la conectan a ella.

Ejemplo 1

Ninguna de las configuraciones de bloques de abajo representan a una ciudad ideal: las primeras dos de la izquierda no satisfacen la primera condición, la tercera no satisface la segunda condición, y la cuarta no satisface cualquiera de las condiciones.



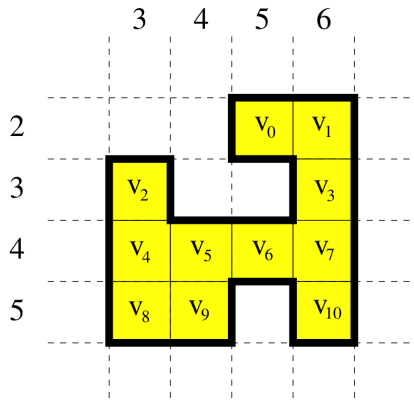
Distancia

Cuando recorremos la ciudad, un *brinco* indica ir desde un bloque hasta uno adyacente. Las celdas

vacías no pueden recorrerse. Sea v_0, v_1, …, v_{N-1} las coordenadas de los N bloques de la grilla. Para las coordenadas de dos bloques distintos cualquiera v_i y vj, su distancia d(v_i, vj) es el número más pequeño de brincos que se requieren para ir desde uno de estos bloques a otro.

Ejemplo 2

La configuración de abajo representa una ciudad ideal hecha de N = 11 bloques de 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.



Problema

Su tarea es, dada una ciudad ideal, escribir un programa para calcular la suma de todas las distancias de pares de bloques entre los bloques v_i y v_j para las cuales i < j. Formalmente, su programa debe calcular el valor de la siguiente suma:

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

Específicamente, usted tiene que implementar una rutina `DistanceSum`(N, X, Y) que, dado N y dos arreglos X y Y que describen la ciudad, calcule la fórmula de arriba. X y Y son ambos de tamaño N; el bloque i es de 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 muy grande para representarse usando 32 bits, tu debes reportar este módulo 1 000 000 000 (mil millones).

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

Subtarea 1 [11 puntos]

Usted puede asumir que $N \le 200$.

Subtarea 2 [21 puntos]

Usted puede asumir que $N \le 2\,000$.

Subtarea 3 [23 puntos]

Usted puede asumir que $N \leq 100\,000$.

Adicionalmente, las dos condiciones siguientes cumplen: dada dos celdas cualquiera no vacías i y j tal que X[i] = X[j], cada celda entre ellas también está no vacía; dada dos celdas cualquiera no vacías i y j tal que Y[i] = Y[j], cada celda entre ellas también está no vacía.

Subtarea 4 [45 puntos]

Usted puede asumir que $N \leq 100\,000$.

Detalles de la implementacción

Usted tiene que enviar exactament un fichero, llamado `city.c`, `city.cpp` o `city.pas`. Este fichero tiene que implementar el subprograma descrito arriba usando los siguientes encabezados.

Programas en C/C++

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

Programas en Pascal

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

Estos subprogramas tienen que comportarse como se describió arriba. Usted es libre para implementar otros subprogramas para su uso interno. Sus envíos no pueden interactuar con entrada y salida estándar, ni con otro fichero.

Ejemplo de evaluador (grader)

El ejemplo de evaluador suministrado con el ambiente esperará la entrada en el siguiente formato:

- línea 1: N;
- líneas 2, …, N + 1: X[i], Y[i].

Tiempo y Memoria límites

- Tiempo límite: 1 second.
- Memoria límite: 256 MiB.