# Ciudad Ideal

Leonardo, como muchos otros científicos y artistas de su época, estaba muy interesado en la planeación de ciudades y en el diseño urbano. Él buscaba modelar la ciudad ideal: cómoda, espaciosa y moderada en el uso de sus recursos, lejos de las angostas y claustrofóbicas ciudades de la Edad Media.

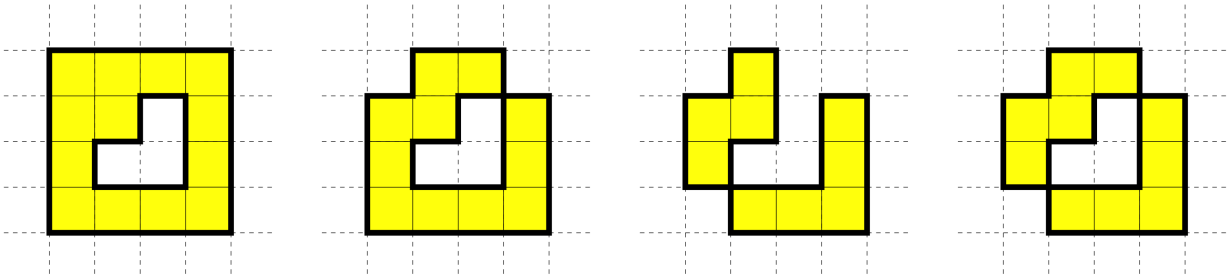
## La ciudad ideal

La ciudad está compuesta de N bloques puestos en una cuadrícula infinita de casillas. Cada casilla está identificada por un par de coordenadas (fila, columna). Dada una casilla (i, j), las casillas adyacentes (si existen) son: (i - 1, j), (i + 1, j), (i, j - 1), y (i, j + 1). Cada bloque, cuando es puesto en la cuadrícula, cubre exactamente una casilla. Un bloque se puede poner en la casilla (i,j) si y solo si $1 \leq i, j \leq 2^{31} - 2$. Usaremos las coordenadas de las casillas para referirnos también a los bloques encima de ellas. Dos bloques son adyacentes si son puestos en casillas adyacentes. En una ciudad ideal, todos los bloques están conectados de tal forma que no quedan "huecos" entre de ellos, es decir, las casillas deben satisfacer las siguientes dos condiciones.

- Para dos casillas *vacías*, debe existir por lo menos una secuencia de casillas *vacías* adyacentes conectándolas.
- Para dos casillas *no vacías*, debe existir por lo menos una secuencia de casillas *no vacías* adyacentes conectándolas.

## Ejemplo 1

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



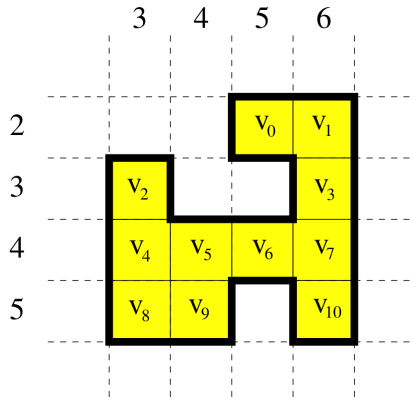
## Distancia

Recorriendo la ciudad, un *salto* representa ir de un bloque a uno adyacente. Las casillas vacías no

pueden ser recorridas. Sean $v_0$, $v_1$, ..., $v_{N-1}$ las coordenadas de los bloques puestos en la cuadrícula. Para cualquier par de bloques distintos de coordenadas $v_i$ y $v_j$, su distancia $d(v_i, v_j)$ es la menor cantidad de saltos que se requieren para ir de uno de los bloques al otro.

**Ejemplo 2**

La siguiente configuración representa una ciudad ideal hecha con N = 11 bloques en las 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

Su tarea es, dada una ciudad ideal, escribir un programa que calcule la suma de las distancias entre todos los pares de bloques $v_i$ y $v_j$ donde $i < j$. Formalmente, su programa deberá calcular el valor de la siguiente suma:

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

Especificamente, debe implementar la rutina `DistanceSum`(N, X, Y) que, dados N y dos arreglos X y Y que describen la ciudad, calcule la fórmula anterior. Tanto X como Y son de tamaño N; el bloque i está en las coordenadas (X[i], Y[i]) para $0 \le i \le N - 1$ y $1 \le X[i], Y[i] \le 2^{31} - 2$. Dado que el resultado puede ser muy grande para representarse con 32 bits, usted deberá reportar el resultado módulo 1 000 000 000 (mil millones).

En el Ejemplo 2, hay $11 \times 10 / 2 = 55$ pares de bloques. La suma de todas las distancias entre pares es 174.

# Subtarea 1 [11 puntos]

Puede asumir que $N \le 200$.

# Subtarea 2 [21 puntos]

Puede asumir que $N \le 2\,000$.

# Subtarea 3 [23 puntos]

Puede asumir que $N \leq 100\,000$.

Adicionalmente, las siguientes dos condiciones se cumplirán, dadas dos casillas no vacías i y j tal que X[i] = X[j], cada casilla entre las dos también será no vacía; dadas dos casillas no vacías i y j tal que Y[i] = Y[j], cada casilla entre las dos también será no vacía.

# Subtarea 4 [45 puntos]

Puede asumir que $N \leq 100\,000$.

# Detalles de implementación

Usted debe enviar exactamente un archivo llamado `city.c`, `city.cpp` o `city.pas`. Este archivo debe implementar la subrutina descrita anteriormente 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;
```

Estas subrutinas se deben comportar como se describe anteriormente. Por supuesto, usted es libre de implementar otras subrutinas para uso interno. Sus envíos no deben interactuar de ninguna forma con la entrada o salida estándar, ni con ningún otro archivo.

### Calificador ejemplo

El calificador ejemplo que se le provee con el ambiente tendrá una entrada esperada en el siguiente formato:

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

# Límites de tiempo y memoria

- Límite de tiempo: 1 segundo.
- Límite de memoria: 256 MiB.