Ciudad Ideal

Leonardo, como muchos otros científicos y artistas italianos de su época, estuvo extremadamente interesado en la planeación de ciudades y diseño urbano. El buscó diseñar una ciudad: confortable, espaciosa y racional en el uso de recursos, más allá de las estrechas y claustrofóbicas ciudades de la edad media.

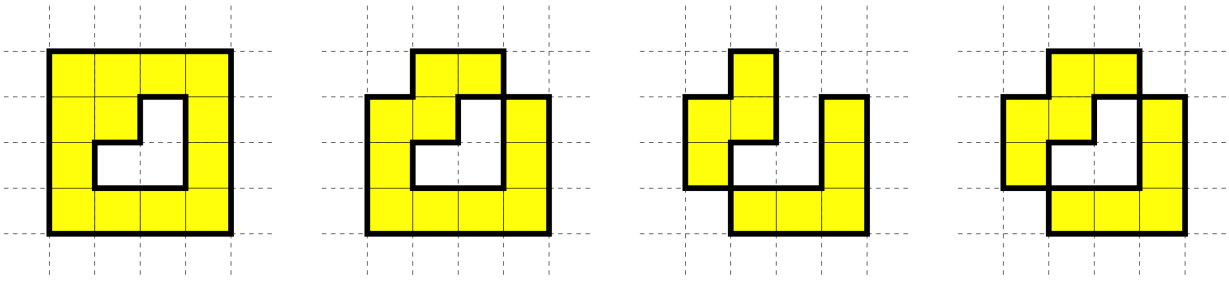
La Ciudad Ideal

La ciudad está hecha de N bloques colocados sobre una cuadricula cuadrada compuesta por varias celdas. Cada celda es identificada por un par de coordenadas (fila, columna). Dada 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 sobre la cuadricula, cubre exactamente una de las celdas. Un bloque puede ser puesto sobre una celda (i, j) si y solo si $1 \leq i, j \leq 2^{31} - 2$. Se usarán las coordenadas para referirse a los bloques que se encuentran sobre ellas. Dos bloques son adyacentes si son colocados en celdas adyacentes. En una ciudad ideal, todos sus bloques están conectados de tal modo que no existen “huecos” dentro de sus fronteras, esto es, las celdas deben satisfacer las dos condiciones mostradas debajo.

- Para cualesquiera dos celdas “vacías”, existe al menos una secuencia de celdas adyacentes “vacías” que las conecta.
- Para cualesquiera dos celdas “no-vacías”, existe al menos una secuencia de celdas adyacentes “no-vacías” que las conecta.

Ejemplo 1

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

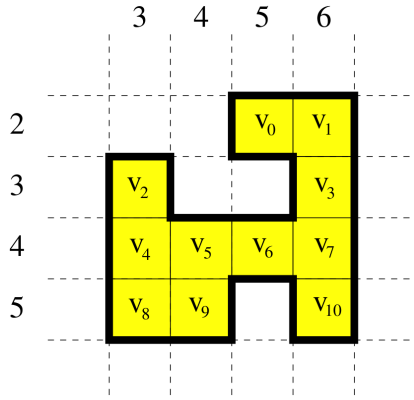


Distancia

Cuando atraviesas la ciudad, un “salto” indica ir desde un bloque hasta otro adyacente. Celdas vacías no pueden ser atravesadas. Supón que $v_0, v_1, \ldots, v_{N-1}$ son las coordenadas de los N bloques puestos en una cuadricula. Para cada dos bloques distintos en coordenadas v_i y v_j, su distancia $d(v_i, v_j)$ es la cantidad más pequeña de saltos que son requeridos para ir desde uno de los bloques hasta el otro.

Ejemplo 2

La configuración mostrada debajo representa una ciudad ideas hecha de 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 $d(v_9, v_{10}) = 4$.



Problema

Tu tarea es escribir un programa que calcule la suma de todos las distancias entre pares de bloques v_i y v_j para los cuales $i < j$. Formalmente tu programa debe calcular el valor de la siguiente sumatoria:

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

Específicamente, tu debes implementar una función `DistanceSum`(N,X,Y) la cual, dados N y dos arreglos X y Y que describan la ciudad, calcule el valor de la formula de arriba. Ambos arreglos X y Y tienen tamaño N; el bloque i está localizado en las coordenadas (X[i], Y[i]) para $0 \leq i \leq N - 1$, y $1 \leq X[i], Y[i] \leq 2^{31} - 2$. Dado que los resultados pueden ser demasiado grandes para ser representados usando 32 bits, debes reportar el resultado modulo 1,000,000,000 (mil millones).

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

Sub-tarea 1 [11 puntos]

Puedes asumir que $N \leq 200$.

Sub-tarea 2 [21 puntos]

Puedes asumir que $N \leq 2{,}000$.

Sub-tarea 3 [23 puntos]

Puedes asumir que $N \leq 100{,}000$.

Adicionalmente, las siguientes dos condiciones son validas: dadas cualesquiera dos celdas no-vacías i y j, tales que X[i] = X[j], cada celda entre ellas es también una celda no-vacía; dadas cualesquiera dos celdas no-vacías i y j, tales que Y[i] = Y[j] cada celda entre ellas es también una celda no-vacía.

Sub-tarea 4 [45 puntos]

Puedes asumir que $N \leq 100{,}000$.

Detalles de implementación

Debes enviar un solo archivo llamado `city.c`, `city.cpp` o `city.pas`. Este archivo debe implementar la función descrita arriba usando las siguientes firmas.

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;
```

Dichas funciones deberán comportarse de acuerdo a la descripción anterior. Está claro que eres libre de implementar funciones adicionales para uso interno. Tus envíos no deberán interactuar de ninguna manera con la entrada/salida estándar ni con algún archivo.

Evaluador de ejemplo (Grader)

El evaluador de ejemplo provisto con el entorno de la tarea, esperará el siguiente formato de entrada:

- 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.