

Ciudad ideal

Leonardo, como muchos otros científicos y artistas italianos de su época, estaba extremadamente interesado en planeamiento de ciudades y diseño urbano. Logró modelar una ciudad ideal: comfortable, espaciosa y racional en su uso de recursos, muy lejos de las angostas, claustrofóbicas ciudades de la Edad Media.

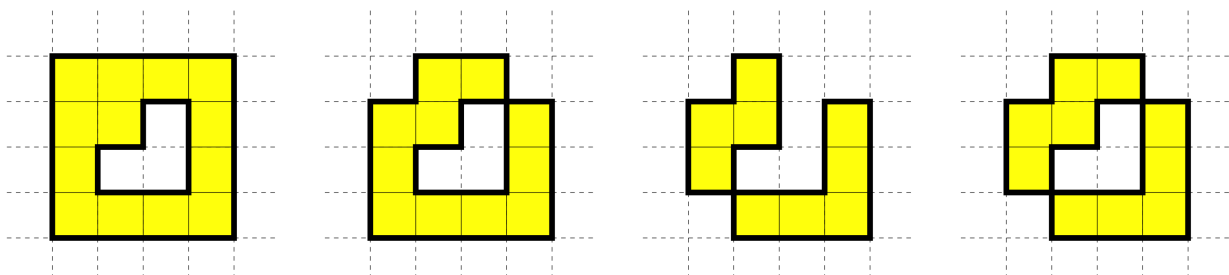
La ciudad ideal

La ciudad está hecha de N bloques colocados sobre una infinita grilla de 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, de ser colocada sobre la grilla, cubre exactamente una de la celdas. un bloque puede ser colocado sobre la celda (i, j) si y sólo si $1 \leq i, j \leq 2^{31} - 2$. Usaremos las coordenadas de las celdas para referirnos también a los bloques apoyados sobre ellas. Dos bloques son adyacentes si ellos están colocados en celdas adyacentes. En una ciudad ideal, Todos sus bloques están conectadas de modo tal que no haya “huecos” dentro de su borde, esto es, las celdas deben satisfacer ambas condiciones detalladas abajo.

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

Ejemplo 1

Ninguna de las configuraciones de bloques abajo dibujadas representa una ciudad ideal: las primeroas 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.



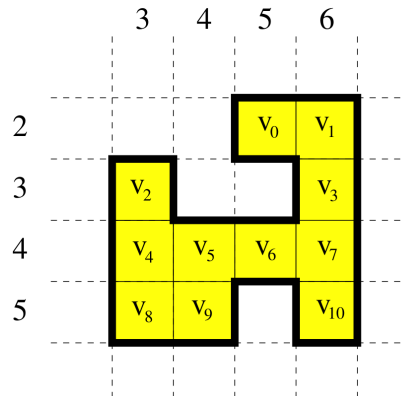
Distancia

Cuando se recorre la ciudad, un hop indica ir de un bloque a uno adyacente. Celdas vacías no pueden ser atravesadas. Sea v_0, v_1, \dots, v_{N-1} las coordenadas de los N bloques colocados sobre la

grilla. Para dos bloques distintos cualesquiera en coordenadas v_i y v_j , su distance $d(v_i, v_j)$ 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 $v_0 = (2, 5)$, $v_1 = (2, 6)$, $v_2 = (3, 3)$, $v_3 = (3, 6)$, $v_4 = (4, 3)$, $v_5 = (4, 4)$, $v_6 = (4, 5)$, $v_7 = (4, 6)$, $v_8 = (5, 3)$, $v_9 = (5, 4)$, and $v_{10} = (5, 6)$. For example, $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, and $d(v_9, v_{10}) = 4$.



Enunciado

Tu tarea consiste en, dada una ciudad ideal, escribir un programa para computar la suma 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:

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

Específicamente, tienes que implementar una rutina `DistanceSum(N, X, Y)` que, dados N y dos arreglos X y Y que describen la ciudad, calcule la formula dada arriba. Ambos X y Y son de tamaño N ; el bloque i está en coordenadas $(X[i], Y[i])$ para $0 \leq i \leq N - 1$, y $1 \leq X[i], Y[i] \leq 2^{31} - 2$. Como el resultado puede ser demasiado grande para ser representado usando 32 bits, debes informarlo 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 de a pares es 174.

Subtarea 1 [11 puntos]

Puedes suponer que $N \leq 200$.

Subtarea 2 [21 puntos]

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

Subtarea 3 [23 puntos]

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 puntos]

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 descrito 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 subprogramas deben 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:

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

Límites de tiempo y memorias

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