Ideale Stadt

Leonardo, so wie viele andere italienische Wissenschaftler und Künstler aus seiner Zeit, war sehr interessiert an Städteplanung. Sein Ziel war es, eine ideale Stadt zu entwerfen: komfortabel, geräumig, rational in der Verwendung von Ressourcen, weit weg von den engen und klaustrophobischen Städten des Mittelalters.

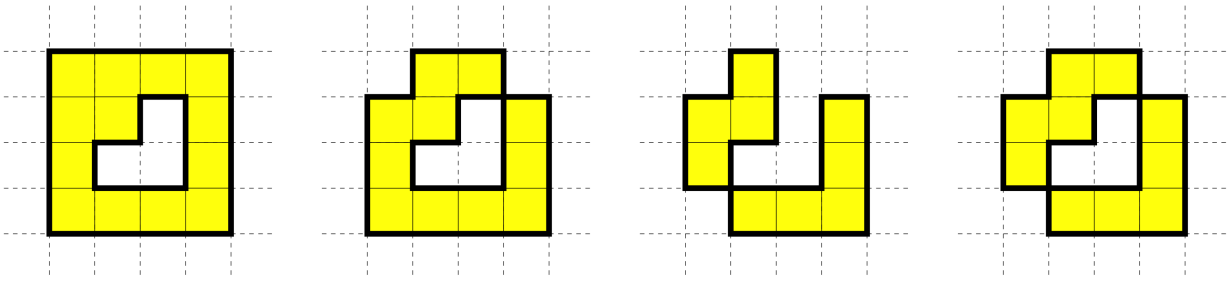
Die ideale Stadt

Die Stadt besteht aus N Blöcken, die auf einem unbegrenzten Raster aus Feldern platziert sind. Jedes Feld ist durch seine Koordinaten (Zeile, Spalte) identifizierbar. Die benachbarten Felder des Feldes (i, j) sind: (i - 1, j), (i + 1, j), (i, j - 1), und (i, j + 1). Jeder Block füllt exakt ein Feld, wenn er auf das Raster platziert wird. Ein Block kann auf das Feld (i, j) genau dann platziert werden, wenn $1 \leq i, j \leq 2^{31} - 2$. Wir benutzen die Koordinaten der Felder auch, um auf die Blöcke darauf zu verweisen. Zwei Blöcke sind benachbart, wenn sie sich auf benachbarten Feldern befinden. In einer idealen Stadt sind alle Blöcke so verbunden, dass sich innerhalb des Randes keine “Löcher” befinden. Dies ist der Fall, wenn die folgenden zwei Bedingungen erfüllt sind:

- Zwei *leere* Felder sind immer durch mindestens eine Sequenz aus benachbarten *leeren* Feldern miteinander verbunden.
- Zwei *nicht-leere* Felder sind immer durch mindestens eine Sequenz aus benachbarten *nicht-leeren* Feldern miteinander verbunden.

Beispiel 1

Keine der unten aufgeführten Konfigurationen an Blöcken entspricht einer idealen Stadt: Die zwei ersten Konfigurationen erfüllen nicht die erste Bedingung (die inneren leeren Felder sind nicht mit den äusseren leeren Feldern durch eine Sequenz aus leeren Feldern miteinander verbunden), die dritte erfüllt nicht die zweite Bedingung und die vierte erfüllt keine der Bedingungen.

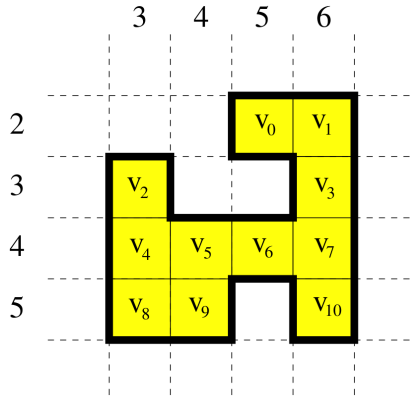


Distanzen

Beim Durchqueren der Stadt bedeutet ein *Hop* (de.: Sprung), dass man von einem Block zu einem benachbarten Block geht. Leere Felder können nicht durchquert werden. Die platzierten N Blöcke auf dem Raster befinden sich an den Koordinaten v_0, v_1, …, v_{N-1}. Die Distanz $d(v_i, v_j)$ zwischen zwei Blöcken an den Koordinaten v_i, v_j ist die kleinste Anzahl an Hops, die man benötigt, um von einem Block zum anderen zu gelangen.

Beispiel 2

Die unten aufgeführte Konfiguration repräsentiert eine ideale Stadt bestehend aus N = 11 Blöcken an den Koordinaten $v_0 = (2, 5)$, $v_1 = (2, 6)$, $v_2 = (3, 3)$, $v_3 = (3, 6)$, $v_4 = (4, 3)$, $v_5 = (4, 4)$, $v_6 = (4, 5)$, $v_7 = (4, 6)$, $v_8 = (5, 3)$, $v_9 = (5, 4)$, und $v_{10} = (5, 6)$. Einige Distanzen sind: $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, und $d(v_9, v_{10}) = 4$.



Aufgabe

Schreibe ein Programm, das in einer gegebenen idealen Stadt die Summe aller Distanzen zwischen zwei Blöcken an den Koordinaten v_i und v_j berechnet, wobei $i < j$ ist. Formal muss dein Programm die folgende Summe ausrechnen:

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

Du musst die Funktion `DistanceSum`(N, X, Y) implementieren, die anhand des Wertes N und der zwei Arrays X und Y, welche die Stadt beschreiben, die oben aufgeführte Formel berechnet. Die beiden Arrays X und Y haben die Grösse N; ein Block i befindet sich an den Koordinaten (X[i], Y[i]) für $0 \le i \le N - 1$, und $1 \le X[i], Y[i] \le 2^{31} - 2$. Da das Resultat möglicherweise nicht mit 32 Bit dargestellt werden kann, muss das Resultat Modulo 1.000.000.000 (1 Milliarde) zurückgeliefert werden.

Im Beispiel 2 befinden sich $11 \times 10 / 2 = 55$ Blockpaare. Die Summe aller Distanzen zwischen diesen Blockpaaren ist 174.

Subtask 1 [11 Punkte]

Du kannst annehmen, dass $N \le 200$ ist.

Subtask 2 [21 Punkte]

Du kannst annehmen, dass $N \leq 2.000$ ist.

Subtask 3 [23 Punkte]

Du kannst annehmen, dass $N \leq 100.000$ ist.

Zusätzlich gelten die folgenden zwei Bedingungen: für zwei nicht-leere Felder i und j mit X[i] = X[j] ist jedes Feld dazwischen auch nicht-leer; für zwei nicht-leere Felder i und j mit Y[i] = Y[j] ist jedes Feld dazwischen auch nicht-leer.

Subtask 4 [45 Punkte]

Du kannst annehmen, dass $N \leq 100.000$ ist.

Implementierungsdetails

Du musst eine einzige Datei einreichen. Diese Datei muss `city.c`, `city.cpp` oder `city.pas` heissen. Diese Datei muss die oben beschriebene Funktion mit folgender Signatur implementieren.

C/C++ Programme

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

Pascal Programme

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

Die zu implementierende Funktion muss sich wie oben beschrieben verhalten. Es ist dir freigestellt andere Unterprogramme zu internen Zwecken anzulegen. Das eingereichte Programm darf in keinster Weise mit der Ein- und Ausgabe oder mit irgendeiner anderen Datei interagieren.

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format:

- Zeile 1: N;
- Zeilen 2, …, N + 1: X[i], Y[i].

Zeit- und Speicherlimits

- Zeitlimit: 1 Sekunde.
- Speicherlimit: 256 MiB.