Idealne miasto

Podobnie jak wielu współczesnych mu naukowców i artystów, Leonardo da Vinci interesował się zagadnieniami urbanistyki i planowania przestrzennego. Postawił sobie za cel zaprojektowanie miasta idealnego: wygodnego i przestronnego, w efektywny sposób gospodarującego zasobami naturalnymi. Ta koncepcja zdecydowanie wykraczała poza średniowieczne standardy wąskich, wręcz klaustrofobicznych miast.

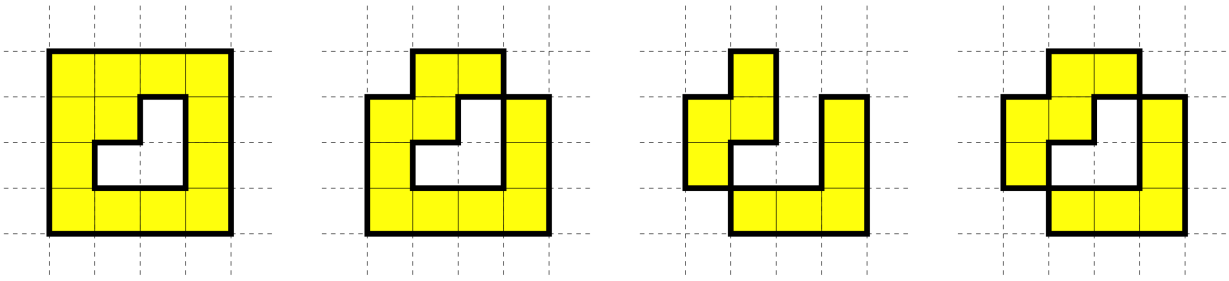
Czymże jest idealne miasto?

W tym zadaniu miasto będziemy wyobrażać sobie jako N kwartałów zabudowy, dla prostoty nazywanych blokami, rozmieszczonych na nieskończonej kratce. Położenie każdego pola kratki określamy za pomocą pary współrzędnych (wiersz, kolumna). Pola sąsiadujące z polem (i, j) to: (i - 1, j), (i + 1, j), (i, j - 1) oraz (i, j + 1). Każdy blok zajmuje dokładnie jedno pole kratki. Każde pole (i, j) zawierające blok miasta spełnia warunek $1 \leq i, j \leq 2^{31} - 2$. Jako współrzędne bloku przyjmujemy współrzędne pola kratki, które ten blok zajmuje. Dwa bloki nazywamy sąsiednimi, jeśli zajmują one sąsiednie pola. Idealne miasto składa się z pewnej liczby połączonych bloków, a jego brzeg nie zawiera "dziur". Dokładniej, muszą być spełnione dwa następujące warunki:

- Pomiędzy każdymi dwoma *pustymi* polami można przejść, poruszając się jedynie po *pustych* polach i w każdym kroku przechodząc na sąsiednie pole.
- Pomiędzy każdymi dwoma *niepustymi* polami można przejść, poruszając się jedynie po *niepustych* polach i w każdym kroku przechodząc na sąsiednie pole.

Przykład 1

Żaden z poniższych układów bloków nie przedstawia idealnego miasta. Dwa pierwsze z lewej nie spełniają pierwszego z powyższych warunków, trzeci nie spełnia drugiego z powyższych warunków, natomiast czwarty nie spełnia żadnego z powyższych warunków.

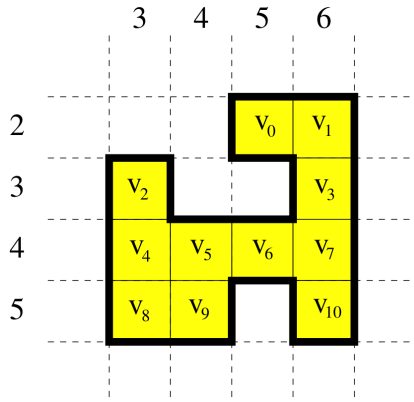


Odległość pól

Po mieście można przemieszczać się za pomocą *ruchów* polegających na przejściu z pewnego bloku do bloku, który z nim sąsiaduje. Ruchy nie mogą prowadzić przez puste pola kratki. Niech $v_0, v_1, \ldots, v_{N-1}$ oznaczają wszystkie bloki miasta. Odległością między blokami v_i oraz v_j, oznaczaną przez $d(v_i, v_j)$, nazywamy najmniejszą liczbę ruchów potrzebnych do przemieszczenia się z jednego z tych bloków do drugiego.

Przykład 2

Poniższy układ przedstawia idealne miasto złożone z N = 11 bloków: $v_0 = (2, 5)$, $v_1 = (2, 6)$, $v_2 = (3, 3)$, $v_3 = (3, 6)$, $v_4 = (4, 3)$, $v_5 = (4, 4)$, $v_6 = (4, 5)$, $v_7 = (4, 6)$, $v_8 = (5, 3)$, $v_9 = (5, 4)$ i $v_{10} = (5, 6)$. W tym przykładzie $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, a $d(v_9, v_{10}) = 4$.



Zadanie

Napisz program, który mając dany plan rozmieszczenia bloków w pewnym idealnym mieście, obliczy sumę odległości między wszystkimi parami bloków v_i, v_j dla $i < j$, tzn. sumę

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

Zaimplementuj funkcję `DistanceSum`(N, X, Y), która przyjmuje opis miasta w postaci liczby N oraz tablic X i Y i oblicza wartość powyższej sumy. Tablice X, Y mają rozmiar N i określają współrzędne bloków; i-ty blok znajduje się na polu o współrzędnych (X[i], Y[i]), dla $0 \le i \le N - 1$, przy czym $1 \le X[i], Y[i] \le 2^{31} - 2$. Ponieważ wynik funkcji mógłby przekroczyć rozmiar 32-bitowych typów całkowitych, należy obliczyć resztę z dzielenia wyniku przez 1 000 000 000 (miliard).

W Przykładzie 2 mamy $11 \times 10 / 2 = 55$ par bloków. Suma odległości wszystkich par bloków to 174.

Podzadanie 1 [11 punktów]

- $N \le 200$

Podzadanie 2 [21 punktów]

- $N \leq 2\,000$

Podzadanie 3 [23 punktów]

- $N \leq 100\,000$

W tym podzadaniu zachodzą dwa dodatkowe warunki: dla dowolnych dwóch niepustych pól i oraz j, takich że X[i] = X[j], każde pole znajdujące się pomiędzy nimi jest także niepuste; dla dowolnych dwóch niepustych pól i oraz j, takich że Y[i] = Y[j], każde pole znajdujące się pomiędzy nimi jest także niepuste.

Podzadanie 4 [45 punktów]

- $N \leq 100\,000$

Szczegóły implementacji

Należy zgłosić dokładnie jeden plik o nazwie `city.c`, `city.cpp` lub `city.pas`. Powinien on zawierać implementację opisanej powyżej funkcji.

Programy w C/C++

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

Programy w Pascalu

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

Funkcja powinna działać dokładnie tak, jak opisano powyżej. Twój program nie powinien korzystać ze standardowego wejścia, standardowego wyjścia lub jakichkolwiek plików.

Przykładowy moduł oceniający

Przykładowy moduł oceniający wczytuje dane w następującym formacie:

- wiersz 1: N;
- wiersze 2, …, N + 1: X[i], Y[i].

Ograniczenia

- Maksymalny czas działania: 1 sekunda.
- Dostępna pamięć: 256 MiB.