# 이상적인 도시

당대의 많은 이탈리아 과학자들과 예술가들처럼 다빈치는 도시 계획과 디자인에 대단한 관심이 있었다. 다빈치는 편안하고, 공간을 넓고 합리적으로 사용하며, 중세 시대 도시의 좁고 답답함과는 거리가 먼 이상적인 도시를 디자인할 계획을 가지고 있었다

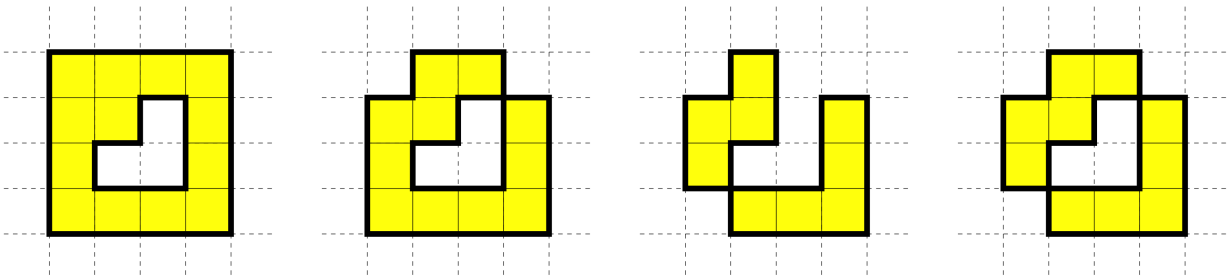
## 이상적인 도시

무한 개 네모 셀들의 격자 위에 N 개의 블럭들을 놓아서 도시를 만든다. 각 셀은 좌표들의 쌍 (행, 열)로 나타낸다. (i, j) 셀이 주어지면, 인접한 셀들은 (i-1, j), (i+1, j), (i, j-1) 그리고 (i, j+1) 이다. 그리드 위에 놓여질 때, 각 블럭은 정확히 하나의 셀을 덮는다. 블럭은 $1 \le i, j \le 2^{31} - 2$ 인 셀 (i, j) 에만 놓을 수 있다. 셀들 위에 놓여진 블럭들을 나타낼 때, 셀들의 좌표를 사용할 것이다. 두 블럭이 서로 인접한 셀들에 놓여지면 두 블럭은 인접했다고 말한다. 이상적인 도시에서 모든 블럭들은 도시안에 *구멍*이 없도록 연결된다. 다시 말해서, 셀들은 아래의 조건들을 만족해야만 한다.

- 임의의 두 *비어 있는* 셀들에 대해서, 인접한 *비어 있는* 셀들로만 이동하는 방법으로 한 셀에서 다른 셀에 도달할 수 있는 경로가 적어도 하나 이상 존재한다.
- 임의의 두 *비어있지 않은* 셀들에 대해서, 인접한 *비어있지 않은* 셀들로만 이동하는 방법으로 한 셀에서 다른 셀에 도달할 수 있는 경로가 적어도 하나 이상 존재한다.

### 예제 1

아래 그림 모두는 이상적인 도시가 아니다. 처음 두 개는 첫번째 조건을 만족하지 않고, 세번째 그림은 두번째 조건을 만족하지 않고, 네번째 그림은 두 조건 모두를 만족하지 않는다.



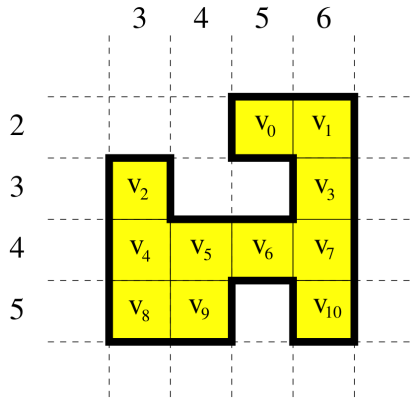
## 거리

도시 안을 이동할 때, *걸음*은 한 블럭에서 인접한 블럭으로 이동하는 것을 말한다. 비어있는 셀들로는 이동할 수 없다. $v_0, v_1, \ldots, v_{N-1}$ 을 그리드 위에 놓여 있는 N 개 블럭들의 좌표라고 하자. 좌표 $v_i$ 와 $v_j$ 를 가진 임의의 서로 다른 두 블럭들에 대해서, 그들간의 거리 $d(v_i,$

$v_j$)는 이 블럭들 중 하나에서 다른 곳으로 가는데 요구되는 걸음들의 최소 수로 정의된다.

**예제 2**

아래 그림은 좌표 $v_0 = (2, 5)$, $v_1 = (2, 6)$, $v_2 = (3, 3)$, $v_3 = (3, 6)$, $v_4 = (4, 3)$, $v_5 = (4, 4)$, $v_6 = (4, 5)$, $v_7 = (4, 6)$, $v_8 = (5, 3)$, $v_9 = (5, 4)$, 그리고 $v_{10} = (5, 6)$ 를 가지는 N = 11 개의 블럭들로 이루어진 이상적인 도시를 나타낸다. 그러면, $d(v_1, v_3) = 1$, $d(v_1, v_8) = 6$, $d(v_6, v_{10}) = 2$, 그리고 $d(v_9, v_{10}) = 4$ 이다.




# 해야 할 일

당신은 모든 가능한 $i < j$ 인 두 블럭 쌍 $v_i$ 와 $v_j$ 에 대한 거리들의 합을 계산하는 프로그램을 작성해야 한다. 정확히 말하면, 프로그램은 다음의 합을 계산해야 한다.

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

구체적으로, 도시를 나타내는 N 과 두 배열 X 와 Y 가 주어 질때, 위 공식을 계산하는 함수 `DistanceSum`(N, X, Y) 를 구현해야 한다. 배열 X 와 Y 는 크기 N 이고, $0 \le i \le N - 1$ 에 대해서, 블럭 i 는 좌표 (X[i], Y[i])를 가지고 $1 \le X[i], Y[i] \le 2^{31} - 2$ 이다. 결과가 32비트를 사용해서 표현하기에 너무 클 수 있기 때문에 결과를 1,000,000,000 으로 나눈 나머지로 계산한다.

예제 2에서 $11 \times 10 / 2 = 55$ 개의 블럭 쌍이 존재한다. 모든 쌍 간의 거리들의 합은 174 이다.

# 서브태스크 1 [11점]

$N \le 200$ 으로 가정할 수 있다.

# 서브태스크 2 [21점]

$N \le 2{,}000$ 으로 가정할 수 있다.

# 서브태스크 3 [23점]

$N \le 100{,}000$ 으로 가정할 수 있다.

더불어 다음 두 조건들이 만족된다.

- X[i] = X[j] 인 임의의 두 비어있지 않은 셀들이 주어질 때, 그들 사이의 모든 셀들 또한 비어있지 않다.
- Y[i] = Y[j] 인 임의의 두 비어있지 않은 셀들이 주어질 때, 그들 사이의 모든 셀들 또한 비어있지 않다.

# 서브태스크 4 [45점]

N ≤ 100,000 으로 가정할 수 있다.

# 구현 세부사항

`city.c`, `city.cpp` 혹은 `city.pas` 중 정확히 한 파일을 제출한다. 이 파일은 다음 선언을 사용해서 위에서 설명한 함수를 구현해야 한다.

**C/C++ 프로그램**

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

**Pascal 프로그램**

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

이 함수들은 위에서 설명한대로 동작해야 한다. 물론 함수 내부에 다른 함수들을 구현하는 것은 자유이다. 제출 프로그램은 기본 입/출력 또는 임의의 다른 파일과 어떤 식으로든지 상호 작용하면 안된다.

**grader 예시**

주어지는 grader의 입력 양식은 다음과 같다.

- line 1: N;
- lines 2, …, N + 1: X[i], Y[i].

# 시간 및 메모리 제한

- 시간 제한 : 1초.
- 메모리 제한 : 256 MB