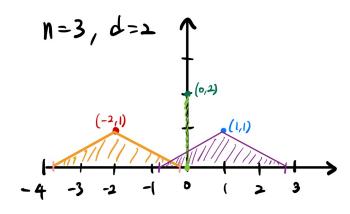
## **Problem Analyze**

2차원 좌표 평면 상에 village(x, y)가 있고, 가능한 well의 위치는 (k, 0)이라고 하자. 이 때, 두 점 사이의 최대 길이는 d라 할 수 있다. 위와 같은 상황에서, n개의 village들은 각각 자신에 대하여 가능한 well의 범위가 있다.

$$\sqrt{(x-k)^2+y^2} \le d \Leftrightarrow (x-k)^2 \le d^2-y^2 \Leftrightarrow (x-k) \le \pm \sqrt{d^2-y^2}$$
  
 $\Rightarrow x \pm \sqrt{d^2-y^2} \le k$ 이므로, k의 범위는  $x - \sqrt{d^2-y^2} \le k \le x + \sqrt{d^2-y^2}$ 이다.

이 때, 가장 왼쪽에 위치한 village의 well의 범위를 한 그룹의 범위라고 지정하고, 해당 범위와 다음 village의 well 범위를 비교하며 겹치는지 확인하는 과정으로 Well의 개수를 구한다. 이 때 겹친다면 같은 그룹이고, 겹치지 않는다면 다른 그룹이라고 생각하여 새로운 그룹을 생성할 수 있다.



## Solution.

- 1. n과 d를 입력받고, n만큼의 village들의 위치 (x, y)를 입력 받는다.
- 2. 주어진 village들의 위치와 d를 이용하여, 각 village들에 대한 well의 범위(start, end)를 구한다.
- 3. 그렇게 구해진 총 n개의 범위를 1차원 배열 wellRange[n]에 저장하고, start를 기준으로 오름차순 정렬한다.
- 4. 현재 well의 범위인 currWellRange를 wellRange[0]으로 초기화하고, 그룹의 수를 1로 초기화한다.
- 5. wellRange[i] (1 ≤ i < n)에 대하여 아래의 과정을 반복한다.
  - 1) wellRange[i]의 start가 currWellRange의 end보다 큰지 확인한다. (wellRange[i].start > currWellRange.end)
  - 2) **만약 크다면** (두 well은 겹칠 수 없는 것이므로) 그룹 수를 1만큼 늘리고, currWellRange를 wellRange[i]로 설정한다.
  - 3) **만약 그렇지 않다면** (두 well은 겹치는 것이므로) 그룹 수는 늘리지 않는다. 다만 **앞으로의 wellRange.start는 min(currWellRange.end, wellRange[i].end)보다 작거나 같아야만 같은 그룹인 것이므로, currWellRange.end는 min(currWellRange.end, wellRange[i].end)로 설정한다.**
- 6. 위의 과정을 통해 구한 그룹의 수를 출력한다.

## Time complexity.

1. N개의 village에 대한 wellRange[n]에 대하여 오름차순 정렬: **O(NlogN)**