874. Walking Robot Simulation

A robot on an infinite XY-plane starts at point `(0, 0)` facing north. The robot can receive a sequence of these three possible types of `commands`:

- `-2`: Turn left `90` degrees.
- `-1`: Turn right `90` degrees.
- `1 <= k <= 9`: Move forward `k` units, one unit at a time.

Some of the grid squares are `obstacles`. The `i`th obstacle is at grid point `obstacles[i] = (xi, yi)`. If the robot runs into an obstacle, then it will instead stay in its current location and move on to the next command.

Return *the* ***maximum Euclidean distance*** *that the robot ever gets from the origin* ***squared*** *(i.e. if the distance is* `5`*, return* `25`*).*

Note:

- North means +Y direction.
- East means +X direction.
- South means -Y direction.
- West means -X direction.
- There can be obstacle in [0,0].

Example 1:

```
Input: commands = [4,-1,3], obstacles = []
Output: 25
Explanation: The robot starts at (0, 0):
1. Move north 4 units to (0, 4).
2. Turn right.
3. Move east 3 units to (3, 4).
The furthest point the robot ever gets from the origin is (3, 4), which squared is
32 + 42 = 25 units away.
```

Example 2:

```
Input: commands = [4,-1,4,-2,4], obstacles = [[2,4]]
Output: 65
Explanation: The robot starts at (0, 0):
1. Move north 4 units to (0, 4).
2. Turn right.
3. Move east 1 unit and get blocked by the obstacle at (2, 4), robot is at (1, 4).
4. Turn left.
5. Move north 4 units to (1, 8).
The furthest point the robot ever gets from the origin is (1, 8), which squared is
12 + 82 = 65 units away.
```

Example 3:

```
Input: commands = [6,-1,-1,6], obstacles = []
Output: 36
Explanation: The robot starts at (0, 0):
1. Move north 6 units to (0, 6).
2. Turn right.
3. Turn right.
4. Move south 6 units to (0, 0).
The furthest point the robot ever gets from the origin is (0, 6), which squared is
62 = 36 units away.
```

Constraints:

- `1 <= commands.length <= 10`4
- `commands[i]` is either `-2`, `-1`, or an integer in the range `[1, 9]`.
- `0 <= obstacles.length <= 10`4
- `-3 * 10`4` <= x`i`, y`i` <= 3 * 10`4
- The answer is guaranteed to be less than `2`31.

874. Walking Robot Simulation

```
/*
   Straight forward Simulation
   Time complexity: O(mlogm+n*ci*logm),  n*(c0+c1+c2+...+)=nD
   O(mlogm+n*D*logm)
   Space complexity: O(m+8)=O(m)
   where:
      n: size of commands table
      ci: commands[i], where 0<=i<n
      m: size of obstacles tables
*/
typedef std::vector<int> vi;
typedef std::vector<vi> vvi;
typedef std::pair<int,int> ii;

class Solution {
   public:
      int modulo(int a,int b){
         return ((a%b)+b)%b;

      }
      int robotSim(vi& commands, vvi& obstacles){
         std::set<ii> obstacles_set;
         for(auto& ob: obstacles) obstacles_set.insert({ob[0],ob[1]});

         vvi directions={{0,1},{1,0},{0,-1},{-1,0}};

         int x=0,y=0,dir=0,ans=0;
         for(auto& c: commands){
            if(c==-1) dir=modulo(dir+1,4);
            else if(c==-2) dir=modulo(dir-1,4);
            else{
               int dx=directions[dir][0];
               int dy=directions[dir][1];
               int i=1;
               while(i<=c && obstacles_set.find({x+dx,y+dy})==obstacles_set.end()){
                     x+=dx;
                     y+=dy;
                     i++;
               }
            }
            ans=std::max(ans,x*x+y*y);
         }
         return ans;
      }
   };
```