Problem Analysis and Solution Explanation

The problem revolves around **minimizing the number of infected houses** in a circular arrangement of houses, given that some houses are initially infected. Cirno can protect one uninfected house each day before the virus spreads further. The goal is to determine the **minimum number of infected houses** at the end if Cirno optimally chooses which houses to protect.

Key Observations

1. **Circular Arrangement**:
    a. Houses are arranged in a circle, so house n is adjacent to house 1. This means infections can wrap around the circle.
2. **Virus Spread Rule**:
    a. An uninfected, unprotected house becomes infected if **at least one of its neighbors is infected**.
3. **Protection Mechanism**:
    a. Each day, Cirno can protect **one uninfected house permanently**. Once protected, the house cannot be infected, even if its neighbors are infected later.
4. **Optimal Strategy**:
    a. To minimize the number of infected houses, Cirno must strategically choose which houses to protect. The protection should aim to "break" chains of infection by blocking the spread between infected houses.

Logic Behind the Solution

Step 1: Analyzing the Gaps Between Infected Houses

- Initially, there are m infected houses. These houses divide the circle into m gaps of uninfected houses.
- For example, if n = 10, m = 3, and the infected houses are [2, 5, 8], the gaps between infected houses are:

    - Gap 1: Between house 2 and 5: 3 houses (3, 4).
    - Gap 2: Between house 5 and 8: 2 houses (6, 7).
    - Gap 3: Between house 8 and 2 (wrapping around): 3 houses (9, 10, 1).
- These gaps represent regions where the virus can potentially spread. By protecting houses in these gaps, we can block the spread of the virus.

Step 2: Sorting the Gaps in Descending Order

- To minimize the spread, we should prioritize protecting houses in the **largest gaps first**, as they have the most potential for infection.
- Sorting the gaps in descending order ensures that we focus on the most critical regions first.

Step 3: Calculating the Number of Protected Houses

- For each gap, we calculate how many houses Cirno can protect before the virus spreads through the gap. This depends on the size of the gap and the number of days available.

Key Insight: Protecting Houses in a Gap

- If a gap has k uninfected houses, Cirno can protect up to floor(k / 2) houses in the gap without allowing the virus to spread through it. This is because:
    - Protecting every alternate house in the gap effectively blocks the spread of the virus.
    - For example, if a gap has 5 houses, Cirno can protect 2 houses (e.g., houses 1 and 3) to stop the virus from spreading.

Adjusting for Days Elapsed

- As time progresses, the virus spreads faster, so the effective size of each gap decreases. Specifically:
    - On the i-th iteration (starting from 0), the effective size of the gap is reduced by i * 4. This accounts for the fact that the virus spreads outward over time, making it harder to protect houses in larger gaps.

Step 4: Counting the Total Number of Protected Houses

- For each gap, we calculate the number of houses Cirno can protect:
  - If the adjusted gap size is 1, Cirno can protect exactly 1 house.
  - If the adjusted gap size is greater than 1, Cirno can protect (gap_size - 1) houses.
  - If the adjusted gap size is 0 or negative, no protection is possible in that gap.
- Summing up the number of protected houses across all gaps gives the total number of houses Cirno can protect.

Step 5: Calculating the Minimum Number of Infected Houses

- Finally, the minimum number of infected houses is:n - protected_count
   where protected_count is the total number of houses Cirno successfully protects.

Implementation Details

1. **Input Parsing**:
   a. Read the number of houses n and the number of initially infected houses m.
   b. Read the positions of the infected houses and sort them in ascending order.
2. **Gap Calculation**:
   a. Compute the gaps between consecutive infected houses, including the circular gap between the last and first infected houses.
3. **Sorting Gaps**:
   a. Sort the gaps in descending order to prioritize larger gaps.
4. **Protection Calculation**:
   a. Iterate through the sorted gaps, adjusting their sizes based on the iteration index and calculating the number of houses Cirno can protect.
5. **Output**:
   a. Print the minimum number of infected houses (n - protected_count) for each test case.

Example Walkthrough

Input:

n = 10, m = 3
infected houses = [2, 5, 8]

Step-by-Step Execution:

1. **Calculate Gaps**:
    a. Gap 1: Between 2 and 5: 3 houses (3, 4).
    b. Gap 2: Between 5 and 8: 2 houses (6, 7).
    c. Gap 3: Between 8 and 2 (wrapping around): 3 houses (9, 10, 1).
    d. Gaps: [3, 2, 3].
2. **Sort Gaps**:
    a. Sorted gaps: [3, 3, 2].
3. **Protect Houses**:
    a. Iteration 0:
        i. Adjusted gap size: 3 - 0*4 = 3.
        ii. Protect 3 - 1 = 2 houses.
    b. Iteration 1:
        i. Adjusted gap size: 3 - 1*4 = -1 (no protection possible).
    c. Iteration 2:
        i. Adjusted gap size: 2 - 2*4 = -6 (no protection possible).
    d. Total protected houses: 2.
4. **Minimum Infected Houses**:
    a. Infected houses: 10 - 2 = 8.

Output:

8

Conclusion

The solution leverages the observation that the virus spreads through gaps between infected houses. By prioritizing the largest gaps and protecting houses strategically, Cirno minimizes the spread of the virus. The implementation efficiently calculates the optimal number of houses to protect, ensuring the minimum number of infected houses at the end.