Poor Pigs Problem Documentation

Table of Contents

1. Problem Statement

You have buckets of liquid, and exactly one of them is poisonous. Your goal is to determine which bucket is poisoned within a given minutesToTest time limit. You can use pigs to test the buckets under the following conditions:

- A pig can be fed from multiple buckets at once.
- Once a pig drinks from a poisonous bucket, it will die after minutesToDie minutes.
- You cannot reuse a dead pig.
- You can repeat the testing process until minutesToTest minutes are up.

Given buckets, minutesToDie, and minutesToTest, the task is to find the minimum number of pigs required to determine the poisonous bucket.

2. Intuition

Instead of testing each bucket individually, we can use the pigs strategically to maximize information from each test.

- Each pig acts like a binary counter where it either dies or survives after a test.
- We take advantage of the fact that pigs can be tested multiple times.
- The idea is to create a testing strategy where pigs drink from a combination of buckets, allowing us to determine the exact bucket based on which pigs survive.

3. Key Observations

- Each pig can be tested multiple times, given by:

$$tests = \frac{minutesToTest}{minutesToDie} + 1$$

(The +1 accounts for the initial state before any testing starts.)

- With p pigs and tests opportunities, the total number of buckets that can be tested is:

$$(\text{tests})^{p}$$

- The problem reduces to finding the minimum number of pigs such that:

$$(\text{tests})\ p \geq \text{buckets}$$

- We can determine p using logarithms:

$$p = \lceil \log_{\text{tests}}(\text{buckets}) \rceil$$

where ceil ensures we get the minimum integer value of pigs required.

4. **Approach**

   i. Calculate tests:

      Compute how many times we can test before the time runs out:

   ii. tests = minutesToTest // minutesToDie + 1

   iii. Find the minimum pigs required:

      Start with pigs = 0 and keep increasing until:

   iv. tests ** pigs >= buckets

   v. Return the number of pigs.

5. **Edge Cases**

Scenario	Explanation
buckets = 1	No pigs are needed since there's only one bucket.
minutesToDie == minutesToTest	Only one test is possible, so we must maximize pig usage.
Large buckets	The logarithmic approach ensures efficiency for large inputs.
buckets is a power of tests	The number of pigs exactly fits the requirement.
buckets is slightly larger than a power of tests	An extra pig might be needed to cover the additional buckets.

6. **Complexity Analysis**

Time Complexity

- We incrementally increase pigs until the condition is met.
- Since the number of pigs grows logarithmically:

  O(logtests(buckets))

Space Complexity

- We only use a few integer variables (tests, pigs).
- The space complexity is:

  O(1)O(1)

7. Alternative Approaches

Brute Force Approach

- Assign one pig per bucket.
- Time Complexity: O(buckets)
- Drawback: Extremely inefficient for large buckets.

Binary Search Approach

- Instead of iterating over pigs, we can use binary search to find the minimum pigs required.
- Time Complexity: O(log(buckets)), slightly better than direct iteration.

8. Test Cases

Test Case	Input	Expected Output
Basic Case	buckets = 4, minutesToDie = 15, minutesToTest = 15	2
More Testing Time	buckets = 4, minutesToDie = 15, minutesToTest = 30	2
Single Bucket	buckets = 1, minutesToDie = 5, minutesToTest = 10	0
Large Buckets	buckets = 1000, minutesToDie = 15, minutesToTest = 60	5

9. Final Thoughts

- The problem is best approached with a mathematical and combinatorial perspective.
- Instead of brute force testing, using pigs efficiently by leveraging multiple tests saves resources.
- The logarithmic nature of the solution ensures it scales well with large inputs.
- Key takeaway: This problem demonstrates information encoding in problem-solving, which is useful in various fields such as networking and system design.