100 PRISONERS AND A LIGHT BULB
(work in progress)

William Wu
wwu@ocf.berkeley.edu
http://www.ocf.berkeley.edu/˜wwu

2002 December 5

1 The Puzzle

There are 100 prisoners in solitary cells. There's a central living room with one light bulb; this bulb is initially off. No prisoner can see the light bulb from his or her own cell. Everyday, the warden picks a prisoner equally at random, and that prisoner visits the living room. While there, the prisoner can toggle the bulb if he or she wishes. Also, the prisoner has the option of asserting that all 100 prisoners have been to the living room by now. If this assertion is false, all 100 prisoners are shot. However, if it is indeed true, all prisoners are set free and inducted into MENSA, since the world could always use more smart people. Thus, the assertion should only be made if the prisoner is 100% certain of its validity.

The prisoners are allowed to get together one night in the courtyard, to discuss a plan. What plan should they agree on, so that eventually, someone will make a correct assertion?[1]

Initially, this puzzle may appear to have no solution. How can 100 people count to 100 using only one bit of communication? Surprisingly, there are algorithms which can solve this problem within the span of a prisoner's lifetime. This work in progress will present and analyze several such algorithms.

Ultimately, the author would like to develop a theory for constructing optimal algorithmic solutions, parameterized by the number of prisoners, number of light bulbs, and a risk tolerance on the validity of an assertion. So far he has only studied in depth the case of one light bulb with zero risk (success requires 100% probability of making a valid assertion). He suspects there exist applications of such a theory to distributed networks.

[1]Puzzle Origin: The author has been unable to track the exact origin of this puzzle, which he heard from a friend during the fall of 2001. According to IBM Research, the puzzle has been making the rounds of Hungarian mathematicians' parties (”Ponder This”, July 2002 Challenge, http://domino.watson.ibm.com/Comm/wwwr_ponder.nsf/challenges/July2002.html).

2 One Counter Method

2.1 Standard Solution

Many struggle with this puzzle because they believe every prisoner must act the same way. If we consider assigning different roles to prisoners, a straight-forward solution presents itself. We can assign one prisoner to lead the group. Denote him as The Counter.

- **Job of The Counter:** Maintains an integer inside his head, initialized to 1. If he enters the living room *and* sees an on light bulb, he increments the integer and turns off the bulb. Otherwise, do nothing. When the integer reaches 100, assert that everyone has been in the living room.
- **Job of Everyone Else:** If the bulb is off *and* he has never turned the bulb on before, turn the bulb on. Otherwise, do nothing.

It can be shown that the probability of success for this algorithm converges to 1. (Perhaps the author will prove this after finals.)

The expected number of days till this algorithm succeeds is 10417.74 days, or 28.54 years. Contrast this to the expected number of days till all prisoners are chosen, which is roughly 100 log(100) = 460.52 days = 1.262 years using a coupon collector analysis. (Note for future work: include variance as well.)

Proof: Let the random variable X return the running time of the algorithm in days. X can be written as the sum of random variables $\{X_i \mid i \in \{1, 2, ..., 99\}\}$, where X_i is the number of days after the $(i-1)^{th}$ unrecorded prisoner was tallied by The Counter until the i^{th} unrecorded prisoner is tallied. Note that there is no X_{100}, because The Counter already counts himself as having been in the living room. By linearity of expectation,

$$E[X] = \sum_{i=1}^{99} E[X_i]$$

Each X_i can be expressed as the sum of two geometric random variables: $X_i = Y_i + Z_i$, where Y_i is the number of days after the $(i-1)^{th}$ unrecorded prisoner was tallied until the i^{th} unrecorded prisoner is chosen by the warden, and Z_i is the number of days until The Counter realizes that the i^{th} unrecorded prisoner was chosen. Note that $Y_i \sim geom(\frac{100-i}{100})$ and $Z_i \sim geom(\frac{1}{100})$. Since the expectation of a geometric random variable is the reciprocal of its probability, we have:

$$\begin{aligned} E[X] &= \underbrace{\frac{100}{99} + 100}_{E[X_1]} + \underbrace{\frac{100}{98} + 100}_{E[X_2]} + \ldots + \underbrace{\frac{100}{1} + 100}_{E[X_{99}]} \\ &= (100 \cdot \sum_{i=1}^{99} \frac{1}{i}) + (99 \cdot 100) \\ &\approx 10417.74 \text{ days} = 28.542 \text{ years} \end{aligned}$$

In general, for n prisoners,

$$\begin{aligned} E[X] &= (n \cdot \sum_{i=1}^{n-1} \frac{1}{i}) + (n-1)n \\ &= n(H_{n-1} + n - 1) = O(n^2) \end{aligned}$$

where H_{n-1} is the (n-1)th harmonic number. This analysis could have been shortened by recognizing that the scenario is analogous to the coupon collector problem, except for an additional factor of n(n-1) contributed by The Counter.

2.2 Improvement: Dynamic Counter Assignment

We can improve this algorithm slightly by dynamically assigning the role of Counter. Let The Counter be the first prisoner to enter the living room twice. Then, if The Counter enters the room on day k, he knows that k-1 people have been in the room (including himself). Thus The Counter can initialize his integer to k-1 rather than 1, and save (k-1) - 1 = k-2 iterations. The algorithm will now have to be split into two stages: an initial 100 day stage during which the prisoners determine who The Counter is, and the following stage, which runs just like the standard one counter solution.

1. **Stage 1: Days 1-100**
   - Days 1-99: The light is initially off. The first prisoner to enter the room twice assigns himself to be The Counter, and turns on the light. He also initializes an integer into his head to the day of reentry minus 1. Aside from these actions on The Counter's behalf, nothing else takes place.
   - Day 100: Note that the first 99 days partitioned the prisoners into four groups:
     (a) Prisoners who entered the room and found an off bulb
     (b) Prisoners who entered the room and found an on bulb
     (c) Prisoners who never entered the room

(d) The Counter who entered the room at least twice, and turned on the light.

The prisoner chosen on day 100 has a special job:

– If bulb is off (probability 100^{-100}): Assert that all have been in living room.
– If bulb is on: If he belongs to groups (a) or (d), turn off bulb.

2. **Stage 2: Days 101- . . .**

   The second stage runs identically to the aforementioned algorithm without dynamic Counter assignment.

   - **Job of The Counter:** Maintains an integer inside his head, initialized to 1. If he enters the living room *and* sees an on light bulb, he increments the integer and turns off the bulb. Otherwise, do nothing. When the integer reaches 100, assert that everyone has been in the living room.
   - **Job of Everyone Else:** If the bulb is off *and* he has never turned the bulb on before, turn the bulb on. Otherwise, do nothing.

The expected running time of this algorithm is 24.42 years. (Note for future work: include variance as well.)

Proof: To determine the number of iterations saved, we compute the expectation of a random variable K, which returns the number of days until the first time a prisoner re-enters the living room. (In a balls and bins context, E[K] is the expected time till the first collision.)

$$E[K] = \sum_{i=k}^{101} kP(K = k)$$

Note that the summation ends at k = 101 because P(K = k) = 0 for k > 101. By the pigeonhole principle, the day of first reentry can be 101 at the latest. Computing the probability mass function of K:

$$\begin{aligned}
P(K = k) &= P(\text{collision on day k} \cap \text{no collision on day k-1} \cap \ldots \cap \text{no collision on day 1}) \\
&= P(\text{no col. day 1}) \cdot P(\text{no col. day 2} \mid \text{no col. day 1}) \\
&\quad \cdot \ldots \cdot P(\text{col. day k} \mid \text{no col. days 1 thru k-1}) \\
&= 1 \cdot \frac{99}{100} \cdot \frac{98}{100} \cdot \ldots \cdot \frac{100-k+2}{100} \cdot \frac{k-1}{100} \\
&= \frac{100!(k-1)}{100^k \cdot (100-k+1)!} \quad \text{for k } \in \{1, 2, \ldots, 101\}
\end{aligned}$$

Thus,

$$E[K] = \sum_{k=1}^{101} k \frac{100!(k-1)}{100^k \cdot (100-k+1)!} = 13.21 \approx 13 \text{ days}$$

and the total runtime is

$$\begin{aligned} E[X] &= (100 \cdot \sum_{i=(13-1)}^{99} \frac{1}{i}) + (99 - (13-1)) \cdot 100 \\ &\approx 8915.75 \text{ days} = \ 24.42 \text{ years.} \end{aligned}$$

On average, over four years are saved with this improvement. In general,

$$\begin{aligned} E[X] &= (n \cdot \sum_{i=\lfloor E[K] \rfloor -1}^{n-1} \frac{1}{i}) + ((n-1) - (\lfloor E[K] \rfloor - 1))n \\ &= (n \cdot \sum_{i=\lfloor E[K] \rfloor -1}^{n-1} \frac{1}{i}) + (n - \lfloor E[K] \rfloor)n \\ &= n(H_{n-1} - H_{\lfloor E[K] \rfloor -2}) + (n - \lfloor E[K] \rfloor)n \\ &= n(H_{n-1} - H_{\lfloor E[K] \rfloor -2} + n - \lfloor E[K] \rfloor) = O(n^2) \\ \text{where } E[K] &= \sum_{k=1}^{n+1} k \frac{n!(k-1)}{n^k \cdot (n-k+1)!} \end{aligned}$$

The one counter approach seems to be generally accepted as "the answer", according to word of mouth and the solution pages of some puzzle web sites. The dynamic counter assignment modification is not as well known.

3 Recursive Multiple Counter Method

Discussions in the forum of the author's puzzle web site[2] have produced multiple counter algorithms which beat the one counter algorithm substantially. Intuitively, one might perceive that a one counter approach is suboptimal, since only one prisoner is exercising his counting and memorization abilities, although all prisoners have these abilities. A better algorithm uses multiple counters who merge their readings over time. In computer simulations using a multiple counter approach with 100 prisoners, the average running time was reduced by over 60%.

[2] [wu :: riddles] at http://www.ocf.berkeley.edu/∼wwu/riddles. Perhaps responsible for popularizing the 100 Prisoners puzzle after the site was recognized by slashdot.org. Most online incarnations of the riddle use the author's phrasing.

3.1 Predefined Roles

This algorithm is parameterized by 4 variables, each to be explained below: C_p, C_s, N_s, and an finite array S[ ] of integers which correspond to stage lengths. Varying these parameters will result in different average performance.

0. **Meeting: Know Your Role**

   - Outline: Each prisoner sticks to one of three possible roles during the algorithm's execution: Primary Counter, Secondary Counter, and Drone. In Stage 1 of the algorithm, the Primary Counter will be responsible for counting at most C_p Drones, and the Secondary Counter will be responsible for counting at most C_s drones. In Stage 2, the Primary Counter will count how many Secondary Counters succeeded in Stage 1, and increment his own count accordingly. Stages 1 and 2 are then repeated until the Primary Counter's count reaches 100, upon which the assertion is made.
   - At the meeting, define who is who. There can be only one Primary Counter. Let the number of secondary counters be denoted by N_s. Everyone else is a Drone.
   - Primary Counter, and all Secondary Counters, are initialized to 1.

1. **Stage 1: Counting Drones; lasts for S[1] days**

   - Days 1 through S[1] - 1:
     (a) Bulb Is Turned On: If a Drone enters the room and has never turned the bulb on before, he turns it on. (The bulb has to be in its off position for the Drone to do this.)
     (b) Bulb Is Turned Off: If a Counter sees an on bulb *and* has not already exceeded his maximal Stage 1 count (C_s for Secondary Counter, C_p for Primary), then he increments his Stage 1 count and turns off the bulb. Else, do nothing.
   - Day S[1] (the last day of Stage 1): Whoever visits the living room should turn the bulb off before he leaves, unless he is a Secondary Counter who counted up to C_s.

2. **Stage 2: Merging Counts; lasts for S[2] days**

   To ease exposition, let us denote those Secondary Counters who managed to count up to C_s as being "completed", since they have completed their task. Recall that the goals of Stage 2 are to tell the Primary Counter how many Secondary Counters are completed, and possibly escape the prison.

   - Days 1 through S[1] - 1:
     (a) Bulb Is Turned On: If a completed Secondary Counter has never turned the bulb on before. However, if the visitor on the last day of Stage 1 was a completed Secondary Counter, he does not turn the bulb on again during Stage 2.

(b) Bulb Is Turned Off: If the Primary Counter sees an on bulb, he increments his master count by C_s, and turns off the bulb.* If the master count reaches 100, he asserts that all prisoners have been in the living room, and the game is over.

*This is key. A multiple counter approach allows the master count to be incremented in chunks larger than one unit. However, the larger the chunk, the less likely a Secondary Counter is to be completed in Stage 1. Thus there are tradeoffs in choosing parameters.

- Day S[2] (the last day of Stage 2): Whoever visits turns off the bulb. Repeat Stages 1 and 2 until everyone is freed. Anyone who has turned on the bulb in the past does nothing for the rest of his time in prison. Finally, the lengths of Stages 1 and 2 can be varied according to the array S[ ]. That is, while Stage 1 may have lasted for 2000 days in the algorithm's first iteration, it may be 1500 in the second iteration, and so on.

Finding closed-form expressions for the expectation and variance of this algorithm's running time may be difficult, and will be deferred by the author until winter break. This will be necessary to optimize parameter choices, probably using multivariable calculus.

In the meantime, computer simulations are easy to implement. Trial and error has produced expectations in the 3300-3800 day range.

3.2 Binary Stages and Dynamic Role Assignment

Alex Harris suggests an alternative approach which the author does not fully understand yet, given current schoolwork time constraints. Harris's approach does not improve the running time substantially according to simulations, but it is more elegant. Apparently if everything is done with binary numbers, roles do not have to be defined a priori. Everyone can begin as an "active" prisoner. If an active prisoner enters the room, they *toggle* the light switch. If the toggle turns the bulb off, the active prisoner calls himself a Counter, and stays active in the next stage. If the toggle turns the bulb on, he calls himself a Drone and becomes inactive from that point on. Of course, both active and inactive prisoners must turn off the bulb and account for it next time if the bulb is on at the end of the stage.

Harris's analysis is reprinted below, adapted from an online forum post. Some probability facts he uses:

- Variance (or mean) of the sum of independent random variables is the sum of the variances (or means).
- A geometric random variable with probability p has mean $\frac{1}{p}$ and variance $\frac{(1-p)}{p^2}$.
- Sum of harmonic series $H(n) \sim \ln$ n.

- $H_2(n) = \sum_{i=0}^{\infty} \frac{1}{i^2} \to \pi^2/6$.
- For p in [0,1]: $(1\text{-p})^n \geq 1$ - np
- Less than $1/k^2$ of a distribution can be > mean + k * (deviation) ln and log_2 are off by a constant factor, so Harris will not distinguish between them.

* * *

In stage i we start with a = n / 2^i active prisoners (prisoners with stageBit = 1) — I only really need that a $\leq$ n which is obvious. Every time an active prisoner is selected by the warden, he flips his bit and becomes inactive for the rest of the phase. Each reduction in the number of active prisoners is an independent geometric distribution with p=(a/n) where a is the current number of active prisoners.Observe that the sum of means is nH(a) and the sum of variances is $< H_2$(a) * n^2, so these are the mean/variance of success of the stage as a whole. Lets say we want at least a 1- $\frac{1}{2}$ log n chance of success for the stage. We can ensure this by waiting (mean) + $\sqrt{2logn}$ * deviation < n log n + $\sqrt{(2logn)}$ * n * $\frac{\pi}{\sqrt{6}}$. This is O(n log n). If we have log n stages, then the odds that all will succeed are at least (1-1/(2log n))^{logn} > 1 - (log n)/(2log n) = 1 - 1/2 = .5 and the total time needed is O(n (log n)^2). Since each pass through all the stages wins with p>.5, we average < 2 passes and the total expected time for success is O(n (log n)^2).

Here is the code snippet from my working test model. There is a little extra complexity on stage ending days, but overall its very simple. On 100 prisoners it takes about 4500 with my arbitrarily chosen stage length parameters. On 256 it takes 14800 or so on average and on 4 it takes about 20. Better parameters might improve those times by a bit (probably not more than 10 percent or so). In practice, the number of days is c(n) n (ln n)^2 where c(4)<3 and drops as you increase problem size with c(128)<2. I think c drops to something like 1.5 asymptotically with current parameters.

```
bool prisonerToRoom(int prisoner, bool lightOn){
      currentDay++;
      if (currentDay == stageCutoffs[currentStage]){


            if (lightOn) { // prisoner must absorb unreceived message
                  myCounter[prisoner] += stageBit;
            }
            stageCompleted(); // stageBit changes here

            // commence the next stage
            if (myCounter[prisoner] & stageBit){
                  myCounter[prisoner] -= stageBit;
```

```
                return true;
            }
            else
                return false;
        }
        else if (myCounter[prisoner] & stageBit){
            if (lightOn){
                myCounter[prisoner] += stageBit;
                return false;
            }
            else {
                myCounter[prisoner] -= stageBit;
                return true;
            }
        }
        else
            return lightOn;
    }
```

4 Next Steps

The author hopes to optimize algebraic expressions for the expectations and variances of the parameterized multiple counter algorithms. Then he will explore a few completely different approaches to solving the puzzle.

An example of a different approach is based on token exchange. Each person starts with one token (conceptually speaking), and we establish

- rules, by which people can transfer some or all of their tokens to each other, and
- a policy, such that the tokens tend to accumulate in fewer hands.

Inherent in the rules will be the fact that a person can only transfer tokens by being in the room, so as soon as anyone gets 100 tokens, the prisoners are free.

The author is also interested in scenarios with multiple light bulbs and a parameterized risk tolerance. Such conditions should make the puzzle more applicable to real world situations, where there are multiple agents working toward a common goal, each randomly taking turns sharing more than one bit of memory, and all willing to accept less than a 100% probability of success.

5 Credits

Several people contributed good ideas for solving this puzzle via the author's web forum. Only their online avatars are known to the author, but their real names will certainly be collected if this semblance of a paper is to be published someday. Important players include Alex Harris and Paul Hammond, all of whom the author would forward this paper to for review. David Lau is also thanked for first introducing the author to this puzzle.