One hundred prisoners and a lightbulb*

Paul-Olivier Dehaye, Daniel Ford, Henry Segerman

November 14, 2003

A single lightbulb flickers into life in the center of the room. 100 prisoners shade their eyes from the glare, then focus on the prison warden standing by the lightswitch, with a standard evil-puzzler's glint in his eye. He begins to speak:

> "In one hour, you will all be taken to your cells to be kept in solitary confinement, with no possibility of communicating with any of your fellow inmates.
>
> Well, almost no possibility... every night from now on, I will choose one of you at random, retrieve you from your cell, and take you to this room, where you may see if the lightbulb is **on** or **off**, and you may turn it **on** or **off** as you wish."

A murmur ripples around the room as the prisoners consider the prospect of having such an effect on their hitherto impotent and externally controlled existences.

> "If at some point, I take you to this room and you believe that all 100 prisoners have been chosen and taken here at some time, then you may tell me this.
>
> If you are correct, I will free you all. If of course you are incorrect... well let's say none of you will live to flip any more lightbulb switches in this world."

He exits with a flourish of his cloak, thoughtfully leaving the lightbulb on.

The prisoners are in the dark as to how to get free, but they are perfectly clear about wanting to be able to at least flip light switches into old age (and it looks like they might need to!). So they must come up with a strategy that will announce that all 100 prisoners have been chosen only if they actually have, with 100% certainty, hopefully before they all die of old age.

At first it seems impossible that any one prisoner could know about what the other 99 have been up to. Coming into the room and seeing the lightbulb is **on** doesn't seem to give you much information. You don't know who set it, and if you flip the switch you have no idea who will see that you flipped it. There

*This is the accepted version of a paper that was published in the *Mathematical Intelligencer* **24** (2003) no. 4, pp 53–61. The publishing staff made some minor corrections. The editors appended a very short biography of each of the authors.

seems to be no way to send a message to anyone in particular. It seems hopeless that they will get out at all. But in fact:

Amazing fact 1. *They can get out.*

Here is how. (You may wish to ponder on your own before reading on.)

If at first you don't succeed...

Strategy 1. Cut the sequence of days into blocks of length 100. The first prisoner to enter the room in a given block turns the lightbulb **off**. If a prisoner enters the room a second time in the same 100 day block, then he turns the lightbulb **on**. If a prisoner enters the room on the last day of a 100 day block, and it is his first time, and the lightbulb is still **off**, then that prisoner knows that every prisoner has been chosen exactly once in this 100 day block. He then correctly declares that all prisoners have been in the central room at least once. If the lightbulb is **on** on the last day in a block then we have failed this time so we try again in the next block of 100 days, and keep trying until someone announces.

Expected results for strategy 1. The probability of succeeding in any given block is the number of orderings of the 100 prisoners divided by the number of possible ways the prisoners could have entered the room. With n prisoners, that is $(n!/n^n)$.

The expected number of blocks which must be used before succeeding is equal to $1/p$ where p is the probability of succeeding with one block. To see this, suppose p is the chance of succeeding in any given block. Then the expected number of blocks until we succeed is equal to $\sum_k kpq^{k-1}$ where $q = 1 - p$. This is equal to

$$p\frac{\mathrm{d}}{\mathrm{d}q}(1/(1-q)) = p/p^2 = 1/p$$

Thus the expected number of blocks is $n^n/n!$. Each block has length n so the expected number of days until freedom is $n^{n+1}/n!$, which is $O(n^{1/2}e^n)$ (using Stirling's formula). For 100 prisoners, the expected value is $100^{101}/100!$ or approximately 10^{44}.

They can get out, although this is a disappointingly large number for the prisoners: about 10^{41} years. Sadly the universe may have ended long before they are free [4].

Amazing fact 2. *They can get out before the universe ends.*

Soul-collecting

Strategy 2. One prisoner, who will be known as *The Countess*, will be responsible for announcing that every other prisoner has entered the room at some time. The other $(n-1)$ prisoners will be *ordinary*.

Each ordinary prisoner starts with a single token, called a *soul*, which they will try to leave in the central room. The Countess will collect souls from the central room until she has all of them. She may then declare success.

We may assume that the lightbulb starts in the **off** position (as the prisoner who enters on the first day may turn it **off** before doing anything else).

> When an ordinary person enters the room and finds the lightbulb **off**, they may drop their soul in the room, if they have not already done so, by turning the lightbulb **on**. If the lightbulb is already **on** then they leave it alone.
>
> When the Countess enters the room, if she finds the lightbulb **on** then she turns it **off** and adds one to her soul count. If her count is now $n-1$ then she knows that everyone must have entered the room so she can declare. If the lightbulb is **off** when she enters she leaves it **off**.

Expected results for strategy 2. For the strategy to complete we need to have a sequence of events happen. We first need a soul dropped in the room, then for the Countess to pick it up, then another soul dropped, etc. As the number of uncounted souls goes down, the probability of a new one turning up on the next day goes down from $(n-1)/n$ for the first soul to $1/n$ for the last soul. Meanwhile, the probability for the Countess to show up on the next day is constantly $1/n$.

Since the expected time needed for an event occuring with probability p on the next day is $1/p$, we immediately get that the time is

$$n\left(\sum_{k=1}^{n-1}\frac{1}{k}\right)+(n-1)(n),$$

which is between n^2 and $2n^2$. Therefore the expected number of days until the prisoners escape is $O(n^2)$.

This is much better than our previous exponential solution. The 100 prisoners should get out in around 10,400 days, or about 29 years. They will be past their best, but they will live to see the outside world. However, they can do much better than that:

Amazing fact 3. *They can get out before they are ineligible for the Fields medal.*

Pyramid Scheme

This is again a method for collecting souls. This time there is no single counter. Rather everyone is involved in a process of collecting souls together. The lightbulb will be worth different numbers of souls on different nights.

Strategy 3. A sequence is given which describes how many souls the lightbulb is worth on each night, which is always a power of two. Let $V(n)$ denote the

number of souls that the lightbulb is worth if it is left in the **on** position on night n, or discovered **on** on night $n+1$.

Assume that the number of prisoners is a power of two. This will turn out not to matter in the end.

A prisoner enters the room on a night and collects however many souls have been left there from last night (so if it is night n and the lightbulb is **on** he picks up $V(n-1)$ souls) and turns the lightbulb **off**. He now looks at the number of souls M that he has collected, but represented in base 2. If the coming night is worth $V(n) = 2^k$ souls then he looks at the binary bit of M worth 2^k souls. If this bit is 1 then he drops 2^k souls by turning the lightbulb **on** and subtracting 2^k from M, his own total of souls collected. If the 2^k-bit is zero then he leaves the lightbulb **off**.

Notice that this has the effect that souls are 'glued together' into lumps of size 2^k which can be transferred on nights which are worth 2^k. Whenever a prisoner has two lumps of size 2^k he glues them together into a lump of size 2^{k+1}. This may occur if he has just picked up a 2^k lump or has just glued two smaller 2^{k-1} lumps together. When all the souls have been glued together into one lump of size $n = 2^{\log_2 n}$ then the prisoner who holds this lump declares success.

We have yet to say what would be an efficient choice of values for the $V(n)$. Starting with a block of nights worth 1 is a good idea, to hopefully glue all the single souls into pairs. Then follow with a block of nights worth 2 to glue into blocks of 4 souls, and so on. We want the lengths of the blocks to be long enough to give a good chance of gluing all the lumps of size 2^{k-1} into lumps of size 2^k, but not too long as we don't want to waste time once they have all been glued.

Expected results for strategy 3. In order to achieve a good asymptotics, we start with a block of $(n\log n + n\log\log n)$ 1-nights, then $(n\log n + n\log\log n)$ 2-nights, then $(n\log n + n\log\log n)$ 4-nights, all the way up to $(n\log n + n\log\log n)$ $(\log_2 n)$-nights. If we have failed after this number of days then we can simply throw up our hands and start over again. In other words, the sequence $V(n)$ repeats.

The probability of gluing all lumps of 2^{k-1} into lumps of size 2^k within $n\log(n) + cn$ nights (where c is some constant) is bounded by $e^{-e^{-c}}$ asymptotically. This is known as the *coupon collector's problem* [2]. With some careful estimation this result can be extended by changing c to a function $c(n) \leq \log(n)$. This gives us a probability of successfully completing each stage of at least $e^{-\frac{1}{\log(n)}}\epsilon(n)$, where $\epsilon(n)$ is an error factor such that $\epsilon(n)^{\log_2(n)}$ tends to 1.

The chance of successfully completing all $\log_2(n)$ stages is at least $\frac{e^{\log_2(e)}}{\epsilon(n)^{\log_2(n)}}$. Thus the expected number of times we need to go through the whole cycle is less than $e^{\log_2(e)}\frac{1}{\epsilon(n)^{\log_2(n)}}$. This gives that the expected number of days is of order $e^{\log_2(e)}\,[n\log(n)(\log(n) + \log(\log(n)))]$, which is $O(n\log(n)^2)$.

It can be proved that no changes to the lengths of blocks of nights $V(n)$ can

improve the asymptotics, but what if we want the best sequence for precisely 100 prisoners?

Our assumption that the number of prisoners is a power of two can be relaxed. To apply our strategy, we just need that everyone starts with at least one soul. So, with 100 prisoners for example, one prisoner could be given 29 souls and the other 99 prisoners given 1 soul to start with. The prisoner who first collects 128 souls declares.

To try to get a good upper bound on the expected number of days to freedom we used a computer simulation to search through the choices for $V(n)$. Our best give around 4400 days, or 12 years. One sequence of block lengths which has about this average is [730, 630, 610, 560, 520, 470, 560, 720, 490, 560, 570, 560, 590, 590], that is to say: 730 1-days, then 630 2-days,... then 560 64-days, then 720 1-days,... then 590 64-days, then (back to the start) 730 1-days, ... The optimisation algorithm works by trying to optimise the $V(n)$ for two passes through the types of days (1-days then 2-days, then 4-days etc.) then just repeats that sequence for the unlikely cases in which the prisoners have still not finished after 2 passes. It is not entirely clear why the above sequence is good, though it makes sense that the first six terms are decreasing since less people have to 'meet' in later stages. It also makes sense that the seventh term is larger, since one would want to give a lot of time for the last two blocks of 64 souls to meet. Giving up at this stage means having to continue through the next seven stages to finish up.

Can they do better with some other strategy? For 100 prisoners we suspect that some sort of hybrid algorithm is probably the best, to use good points of more than one strategy. Collecting together souls as in the pyramid scheme is certainly a good idea to start with, but something else may be better in the endgame. A hybrid given by B. Felgenhauer [1] uses the pyramid scheme to start with, but has a Countess start collecting midway through. His sequence of block lengths (chosen by hand) has expected days of around 3949, and running our optimisation program on on the variables for his strategy gives around 3890.

Asymptotically, they cannot do better than $O(n \log n)$ expected number of days to freedom, because that is the expected number of days for everyone to have visited the room, ignoring that all prisoners actually have to communicate!

Variations

Here are some variations you might like to think about. Each variation assumes all the conditions in the original problem, but with some aspects altered. In each case, you might like to ask yourself whether the prisoners can escape, and if so what is an efficient way to do this. We assume that the lightbulb always starts **off**.

1. **Multiple bulbs** — The central room contains two (or more) lightbulbs (the communication channel is wider).

2. **Multiple rooms** — There are two (or more) identical rooms. The prisoners are taken to one at random but don't know which they are in.

3. **Separate transmitter/receiver** — The warden turns the lightbulb **off** at 12AM, chooses one prisoner to visit at 1AM, and chooses again for someone to visit at 2AM. The visitors only transmit or receive, not both.

4. **Malicious Warden** — The warden is malicious and knows the strategy that the prisoners will use (he listens to them agreeing on what to do). Each day he will choose which prisoner to allow into the room. His conscience demands that he allows every prisoner to visit the room infinitely often.

5. **All prisoners have to announce** — The condition for everyone to be freed is that every prisoner must correctly announce (at some time). In other words: every prisoner must be sure that all prisoners have been to the room.

6. **Simultaneous announcing** — Anyone can announce on any day, not just the prisoner who was selected that morning. The condition for everyone to be freed is that everyone must correctly announce on the *same* day. If they are incorrect or if some announce while some do not then they all die.

7. **Prisoners are freed when they announce** — Everyone must correctly announce at some time. When someone announces, they are freed (never visit the room again), but the others stay until they too announce. Visitors are chosen uniformly among the remaining prisoners. Note that the prisoners are still most interested in everyone escaping, rather than minimizing their own time to escape.

8. **Red/Blue cells (one announcer)** — The prisoners are allocated red or blue prison cells. The announcing prisoner must correctly say how many red cells there are in order for them all to be released.

9. **Numbers in cells (one announcer)** — Each cell has a natural number written on the wall. The announcer must give all the numbers.

10. **All prisoners send messages to all prisoners** — (this is a combination of 5 and 9).

11. **Random visiting times** — The prison is subterranean, with no clocks, calendars, or any other information as to what the time is. The prisoners lose all track of time, and the warden chooses prisoners at random times. In other words, the prisoners have no idea how many people have visited the room since they were last there, and cannot use strategies which count days. They only know the order in which events occur.

12. **Random times, all prisoners must announce** — Combine variations 5 and 11. Every prisoner must announce that everyone has visited at some time, and they cannot use day counting strategies.

13. **Random times, message from one to one** — There are only two prisoners and the transmitter has to send a message (a natural number) to the receiver, but as in 11 they cannot count days.

14. **Random times, messages from many to one** — Combine variations 9 and 11. One announcer must give all numbers written on the walls, and the prisoners lose track of time.

15. **Random times, messages from many to many, 2 lightbulbs**

16. **Random times, messages from many to many, 1 lightbulb**

We now give a spoiler for most of these problems. It turns out that the strategies listed above (or slight modifications of them) are suitable for most of these variations.

1. **Multiple bulbs** — Counting souls (strategy 2) will still work, and can be made even faster as 2^k souls can be left in a room which has k distinct lightbulbs. $\log_2 n$ lightbulbs allow for the best possible time to escape — as soon as everyone has actually been in the room then the prisoner in the room can declare. Strategy 3 can also be improved by allowing gluings of souls into larger lumps, such as lumps of size $(2^k)^l$ if there are k distinct lightbulbs.

2. **Multiple rooms** — Counting souls (strategy 2) will still work. It will be slower, although the expected time until escape (for number of rooms independent of n) is still $O(n^2)$ days.

3. **Seperate transmitter/receiver** — A strategy similar to soul-collecting (strategy 2) works. The Countess always picks up and never drops souls. Everyone else drops souls at every opportunity (though they are forced to pick them up if they find them). This strategy has expected time between $n^2 \log_2 n$ and n^3. (If there are k souls outstanding then the chance of the countess picking one up the coming night is between $\frac{k}{n} \times \frac{1}{n}$ and $\frac{1}{n^2}$, depending on how those k souls are distributed. This gives a total expected time of between $n^2 \log_2 n$ and n^3.)

4. **Malicious Warden** — Strategy 2 will work, although there is clearly no bound on the time until escape — it depends on how mean the Warden wants to be.

5. **All prisoners have to announce** — 'Try-try-again' (strategy 1) works. Interleaving cycles of strategy 3 will also work: Each prisoner has one type of soul for each prisoner who will have to announce. One cycle is given to each prisoner's attempts to collect the souls destined for her, then after n

of these a second cycle is devoted to each prisoner and so on. This gives an expected time of $n^2 \log^2(n)$.

6. **Simultaneous announcing** — The prisoners cannot be sure of escaping. Suppose they will announce on day A. There is a first day, D, on which they all know this. The prisoner who enters on day D knows that they have entered and the state of the lightbulb. Every other prisoner only knows that they did not enter the room on that day. If a different prisoner entered on day D then all of the other prisoners who did not enter would have the same information, and so would have to come to the same conclusion: that they should announce on day A (provided there are at least 3 prisoners). Therefore it cannot matter who enters on day D, so they must all know on day $D-1$. Which contradicts the assumption that D was the first day they all knew they would announce on day A.

7. **Prisoners are freed when they announce** — 'Sloshy' soul-collecting (as in the answer to variation 12 below) will work. When a prisoner has collected 100 souls and then given them all away again they may declare and be set free.

8, 9, 10. **Red/Blue cells, or Numbers in cells (one or all announcer(s))** — The prisoners can escape. See the Über-theorem (below) for a strategy and proof.

11. **Random visiting times** — Counting souls (strategy 2) will still work.

12. **Random times, all prisoners must announce** — 'Sloshy' soul counting will work. The lightbulb is always worth one soul. Any prisoner who has not announced does the following: If the lightbulb is **on** when they enter then they collect the soul and turn the lightbulb **off**. If the lightbulb is **off** when they enter and they have one or more souls then they drop one and turn the lightbulb **on**. Any prisoner who has already announced always drops any souls which they have, and leaves any that are in the room. This strategy has expected time order less than or equal to e^n. This can be shown by constructing an appropriate Markov chain and giving lower bounds for the chance that a given prisoner will announce in the next 200 days. Notice that when there is only one prisoner left to announce, this strategy reduces to strategy 2, soul-collecting with one Countess.

    Note that this strategy would also work (less efficiently) if prisoners who have already announced just continue to slosh souls around (give and take souls rather than just give). This is because a random walk in a finite space will eventually get everyplace. We will use this fact extensively later on.

    Another strategy is that each prisoner who is not a soul-collector has a (very small) chance each day they enter of becoming one. After a number of visits to the room as a soul-collector they give up and go back to being

an ordinary soul-giver. Any prisoner who has already announced always gives souls and never collects.

Can you think of a variation where the best strategy is worse than exponential in the number of prisoners?

13. **Random times, message from one to one** — We have two prisoners, one of whom is trying to send a message to the other. The transmitting prisoner encodes the message as a natural number, M. He tries to give the receiving prisoner M souls. The problem now is how the receiver knows when the message has been sent — how does she know when she has received all of the souls? To do this she occasionally puts a soul back into the room when she finds it empty. Hopefully the transmitter, having dropped all of his souls, will take the last soul back — thus indicating that he is finished. The receiver will then see that the soul has been taken and know that all of the souls have been sent, because the transmitter will only pick up a soul when he is done.

    To do this reliably the two prisoners behave as follows:

    The transmitter drops all of his souls until he has none left. When he has no souls left he will take one soul from the room if he can. When he has one soul he will drop it in the room if he can.

    The receiver takes every soul that she can, although she occasionally drops one back in the room ('pings'). If when she next enters the room she finds that the soul she has dropped has been taken, then she knows that the transmitter is finished and so knows the total number of souls sent.

14. **Random times, messages from many to one** — For n prisoners transmitting and one receiving, the transmitters all behave as in variation 13. First suppose that the receiver wants to know the sum of the numbers of the transmitters, M.

    This time the receiver occasionally tries to drop n souls back at the same time. The only way that all n pings will be taken is if all n of the transmitting prisoners are finished. When she succeeds then her maximum value was the sum of the numbers of the transmitters, M.

    What happens is that the receiver's total collected souls usually increases, but never falls back as much as n from the current all-time maximum unless that maximum is the total number of souls being transmitted. When a transmitter finishes, the receiver's total is allowed to slosh back by one more than before. When all transmitters are finished then the receiver's total will slosh between M and $M - n$, and when she sees both extremes in that order then she knows it is done.

    Knowing the total, M, is enough to allow all the n transmitters to send arbitrary messages. Choosing base 2, give the i-th transmitter digits $i, i + n, i + 2n, \ldots$ in which to encode his message.

15. **Random times, messages from many to many, 2 lightbulbs** — We can use the solution to variation 13 together with a way to pass around who is transmitter and who is receiver. To be precise, they use lightbulb one just as in 13. Some prisoner is chosen to be the first transmitter. We assume lightbulb two is **on** to start with. Whoever turns it **off** (picks up the 'listening stick') is the first receiver. The transmitter sees that the listening stick has been picked up, and starts transmitting on lightbulb one. When the receiver knows the message is done, he puts down the listening stick and becomes the new transmitter. The new receiver is whoever next picks up the stick. The prisoners keep sending messages around (without knowing who they are transmitting to) and eventually each prisoner collects all the messages.

16. **Random times, messages from many to many, 1 lightbulb** — See the Über-Über-theorem below.

Über-theorems

We will now give our method for variations 8, 9 and 10.

It turns out that each prisoner can transmit an arbitrary message to all of the other prisoners, using only the one light.

We will start with one prisoner transmitting one bit to every other prisoner. If the transmitter wants to send a 0-bit then on any even-numbered day he leaves the lightbulb **on** and on any odd-numbered day leaves the lightbulb **off**. If he wants to send a 1-bit then on any even day he leaves the lightbulb **off** and on any odd day he leaves the lightbulb **on**. Every other prisoner leaves the lightbulb **off**. Now any prisoner who finds the lightbulb **on** when they enter the room will know for sure that the transmitted bit is a 0 or a 1, depending on whether the previous day was even or odd. Every prisoner will find the lightbulb **on** at some time (with probability 1), and so will receive the message. Of course, there is nothing special about even and odd days. Any bijection between $\mathbb{N}$ and $\{0,1\} \times \mathbb{N}$ would do just as well. For example, $j \leftrightarrow (0,k)$ would mean that day j is the k^{th} 0-bit day. Those days whose number correspond to $(0,n)$ are 'even days' and those which correspond to $(1,m)$ are 'odd days'.

To send two bits, divide the days into four sets. In other words, provide a bijection between $\mathbb{N}$ and $\{0,1\} \times \{0,1\} \times \mathbb{N}$. The first bit is represented by the first two types of day, 0 and 1 mod 4 say, and the second bit by the other two types of days, 2 and 3 mod 4 say. Any prisoner who finds the lightbulb **on** will know for sure one of the bits being transmitted.

To transmit a message of arbitrary length, provide a bijection between $\mathbb{N}$ and $\{0,1\} \times \mathbb{N} \times \mathbb{N}$.

To allow every prisoner to transmit a message to every other prisoner, first divide the days among the prisoners (so that each is allocated an infinite number) and then run the above algorithm with prisoner k transmitting on days which are allocated to her. For M prisoners, this can be thought of as given by a

bijection between $\mathbb{N}$ and $\{1, \cdots, M\} \times \{0, 1\} \times \mathbb{N} \times \mathbb{N}$.

To speed up transmission, if another prisoner knows a given bit in one of the messages being transmitted then they can retransmit this bit by acting as the transmitter would — 'echoing' the message.

Über-Über-theorem

We will now discuss a method that allows each of the prisoners to send a set of arbitrarily long messages, one to each other prisoner. We assume further that we are in the setting of variation 11 ('Random visiting times'), and hence that the prisoners have no time reference other than the order in which events occur. Unlike all the variations we discussed until now, this one could not be solved using direct modifications of strategies 1 or 2. One of the authors (D.F.) came up with what we think is an original strategy.

The idea of the method

- The n prisoners will have agreed upon an ordering among them ahead of time.
- Prisoner 1 will be the *observer*, looking at the *system* formed by all the other prisoners (and the lightbulb). Those non-observers will be called *robots* because they will follow a simple rule.
- Before starting his rule, the first transmitter, say prisoner n, introduces 0 or 1 souls into the system.
- The observer will try to deduce how many souls were originally introduced from the behavior of the robots. For this, prisoner 1 has different procedures at his disposal:
  - Two *testing* procedures P_0, P_1 that allow prisoner 1 to conduct experiments. He is trying to answer positively to one of the two questions Q_0, Q_1: *"Did prisoner n introduce i souls ($i = 0$ or 1) in the system?"*. However, both P_0 and P_1 can only produce positive results, or be inconclusive. Hence prisoner 1 will only answer negatively to Q_1 when P_0 is conclusive.
  - A *resetting* procedure that allows prisoner 1 to set the system back to its original position (the number of souls in the system is as the transmitter left it). This allows him to proceed with additional experiments.

  The two testing procedures will eventually give an answer to the observer.
- Now prisoner 1 triggers prisoner 2 into an observing phase. That is, they (more or less) exchange their roles, and prisoner 2 becomes an observer, while prisoner 1 starts following a simple rule and so becomes a robot. Eventually, from the experiments he will conduct on the system formed

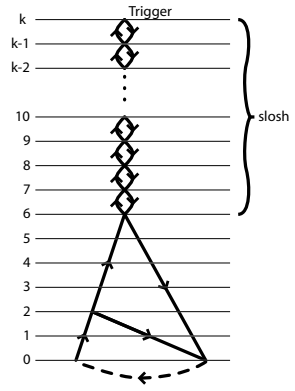
by the other prisoners, prisoner 2 will find out which bit prisoner 1 left in the system and then becomes a robot.

- This continues, cycling through all the prisoners. We have each prisoner i sending a first bit to prisoner $i+1 \mod n$, then all of them sending a second bit, etc...
- Using intermediates, any prisoner can send a message to any other, and not only to his follower in the ordering.

The simpler case $n = 3$

We now describe each step in full for the case $n = 3$.

Simple rules. The behavior of the prisoners who are not currently observing will be given by the directed graphs φ_k, with k a positive integer (see diagram 1). These graphs describe the number of souls each prisoner is eager to have at any time, and hence determine whether he wants to drop or grab a soul each time he enters in the room. The graphs are to be read left to right, and considered to repeat (the dashed line). At any time where more than one option is offered, the prisoner chooses which option to try with equal probability.



To start with, one of the robots will follow φ_{k_0} and one will follow φ_{k_1} Assume that k_0 is big, and k_1 is bigger. This will be made precise later. We play the role of prisoner 1, and (for now) only observe prisoner 2 (and 3) running their instructions φ_{k_0} (resp. φ_{k_1}). More precisely, when we get a chance to go in the room, we note whether the state of the lightbulb has changed from the last time we were there (what we call a *flickering*).

If the total number of souls in the system is 0 (remember we include the lightbulb in the system!), nothing can happen because both prisoners are both eager to get more souls, but none are available. If the total is 1, the lightbulb might be switched **on** and **off** some small number of times (if the prisoner who starts with the soul is initially eager to get rid of it), but eventually one of the

2 prisoners will have 1 soul and be eager to have 2, and the other will have 0 and be eager to have 1. So the situation will stall there after a finite number of flickerings. Similar stops will occur if there are 4 or 5 souls in the system.

On the other hand, if 2 souls are available in the system, the system might stop in a situation where each have 1 soul and are eager to have 2, but more importantly, the lightbulb *might* be turned **on** and **off** an arbitrarily large amount of times, if they both keep going through a sequence $2, 1, 0, 1, 2, 1, 0, ...$ (with a delay in their phases). The lightbulb is then said to *flicker indefinitely*. The same thing could happen if there are 6,7,... souls in the system. Finally, in the case of 3 souls, the system might produce indefinite flickering in the lamp in a more complicated fashion.

This behavior is summarized in the following chart.

Number of souls in the system	Indefinite flickering
$6, 7, ..., k_0 + k_1$	possible
$4, 5$	impossible
$2, 3$	possible
$0, 1$	impossible

It is also worth noting that there exists an integer M such that if there are 0, 1, 4 or 5 souls in the system, the system will stall in less than M flickerings. Hence, observing $M + 1$ flickerings will guarantee that we are not in any of the cases 0, 1, 4 or 5, what we call a *positive* result.

Experimentation. Assume the system contains either 0 or 1 soul, and conduct one of the following procedures:

P_1 Add 1 soul to the system. Wait for a positive result for some time. If this positive result arrives, return **Yes**, otherwise return **unknown**.

P_0 Add 3 souls to the system. Wait for a positive result. If this positive result arrives, return **Yes**, otherwise return **unknown**.

The waiting times should be taken so that we can potentially observe at least $M + 1$ flickerings and hopefully get a positive result.

We have the following chart of outcomes:

# of souls originally	0	1
# of souls after adding-step in P_0	3	4
Possible outputs for P_0	**unknown**, **Yes**	**unknown**
# of souls after adding-step in P_1	1	2
Possible outputs for P_1	**unknown**	**unknown**, **Yes**

Hence, a positive result to P_i guarantees a positive answer to Q_i.

Assuming we did not get a conclusive result we would certainly like to run further experiments, but the system has probably stalled. What should we do now?

Resetting. If we could return the system to its original state with 0 or 1 soul (as set up by prisoner 3), we could experiment further. To do this, we would like to take souls out of the system. It seems hopeless, if for instance one robot has no souls, the other has 5 and they are both eager to have more. But if we are ready to give them some, they will eventually have 6 and be willing to drop the souls again. We can then grab those leftovers, until we are back to the initial number (0 or 1). This allows us to conduct other experiments, and hence to determine eventually whether prisoner 3 left behind a soul or not. Note that we never have to raise the number of souls added to the system to more than 12 to get it moving again, since with 12, at least one robot prisoner is at the start or into his 'slosh' region, and is willing to either take or give souls.

Triggering. Now that we know what the bit sent by prisoner 3 was, we prepare our message for prisoner 2 by setting the total number of souls to 0 or 1. After that, we would like to signal prisoner 2 to start his role of observer. This is where the numbers k_i come into play.

Prisoner 2 has agreed beforehand that he will be 'triggered' when he has exactly 18 souls ($k_0 = 18$). Note first that we never needed to go that high during our experimentation phase (we needed to go at most up to 12). So we can be sure that we have not triggered prisoner 2 before now. We drop those 18 souls in the system, and then start to apply the rule φ_{k_2} for some k_2 agreed on ahead of time, bigger than k_1.

We now have 18 or 19 souls in the system, and each prisoner is running a rule $\varphi_\star$. We only have robots running the place! So the whole system evolves according to a random walk. Since there are only 18 or 19 souls there are finitely many possible states. Moreover, we know that one of the prisoners has at least 6 souls, and hence the option of increasing or decreasing his amount of souls. This guarantees that our random walk never stops, and there is a non-zero probability of getting from any state to any other state. Hence prisoner 2 will eventually end up with 18 souls.

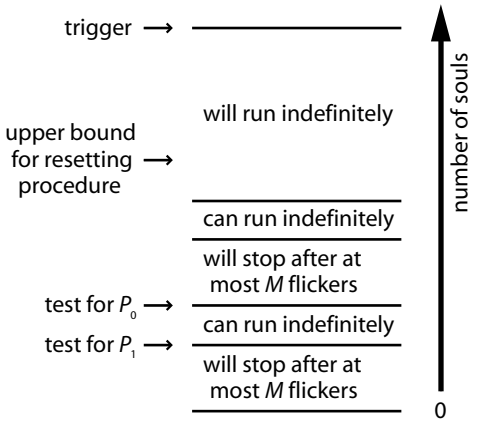
Now that he has his 18 triggering souls, prisoner 2 just needs to erase them in his mental count of souls. He is back to 0 souls, and there might be 1 soul left somewhere else in the system. He becomes an observer and his situation is similar to the one enjoyed by prisoner 1 at the start.

In the case of 3 prisoners, we can actually take $k_0 = 18, k_1 = 20, k_2 = 22, k_3 = 24, \cdots$, and in general the k's will increment by 2 each time. The only requirements are that they are big enough that with that many souls in the system (or one more if the message is a 1) the system never gets stuck (when all prisoners are robots), and that prisoners are not triggered too early when one is trying to trigger someone else. Incrementing by 2 gives just enough leeway so that the 1 bit message doesn't set someone else off too early.

Cycling. Now the prisoners just have to cycle through that algorithm, and give further bits to the prisoner following them in the ordering. This will eventually allow them to exchange arbitrarily long messages with the other prisoners too.

The case of more prisoners

We would like first to identify the important properties that the rules φ_k have that allow the algorithm to work. Really all we care about is the behaviour of the system as a whole. Specifically we want it to behave in different ways depending on the number of souls in the system, as shown in diagram 2. In the



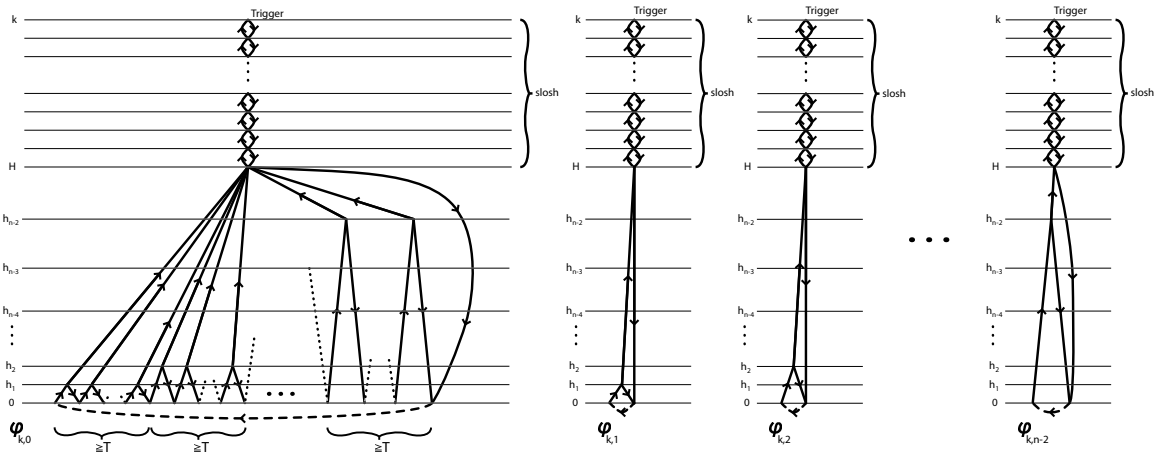
case of 3 prisoners, the test for P_1 is done at the boundary between 1 and 2 souls and the test for P_0 at the boundary between 3 and 4 souls. The maximum number of souls that the observer needs to add to the system to reset it is 12. Also, in sending the first bit, the trigger value k is 18.

Note that in the case of $n = 3$, the fact that all the rules used are of the same type is not really important. In the general case, we will have $n - 1$ types of rule, all with different trigger values k, and we require the triggering prisoner to adopt the same type of rule as the one the triggered prisoner is running. He knows which rule this is just by counting the number of cycles all the prisoners have gone through.

The rules. A set of rules that will work for general n are shown in diagram 3.

We set h_1 to 2, and the other h_i are defined recursively, so that $h_i \geq 2\sum_{j=1}^{i-1} h_j$. The value T refers to a number of soul exchanges required to cover that section of the graph, rather than the number of peaks. Take the number of peaks s_i to be such that the total 'length' $2s_ih_i \geq T$. The value of T will be specified later.

All experimentation happens for values of souls less than H, so once a prisoner starts up on the long journey towards H, they will never be able to come back down until the observer wants to reset the system. H has to be set larger than $h_{n-2} + \sum_{i=1}^{n-2} h_i$ (the sum of the highest peak in each $\varphi_{\star,i}$) so that



the normal running of the system, with other prisoners on their zig-zags down below will not bring an escapee to H and allow him to go back down. Again we define the exact value of H a little later, but assume for now that it is big.

The trigger values k are different for each prisoner. The algorithm will work with $k \geq (n-1)H$ so that with that many souls in the system, at least one prisoner is into his slosh range and therefore the system cannot get stuck when that many are added. They need only increment by 2 each time, as in the case for 3 prisoners. To trigger the prisoner running rule $\varphi_{k,\star}$ simply add k souls, and your message (0 or 1), and become a robot. As in the case of 3 prisoners, the random walk will eventually end up with the prisoner being triggered on k souls, and all other prisoners have triggers of at least 2 more than k so will not be triggered prematurely.

Clearly this system will not run indefinitely with 0 souls. It is also clear that it *might* run indefinitely with $\sum_{i=1}^{n-2} h_i$ souls. Here is one sequence that, if followed, will run forever: Call the prisoner applying the rule $\varphi_{\star,i}$ robot i. To start, set robot 0 to be at the bottom of any valley on his cycle, just before a peak of altitude h_j, and give to each robot i exactly h_i souls (necessarily starting at the peak of his cycle). If robot j gives his h_j souls to robot 0 and then takes them back, we are in a similar position to the one we started with. We can continue doing this indefinitely.

We need to show that the system gets stuck for some higher value of souls. This will require us to prove that no proper subset of the robots can run indefinitely, if there are less than H souls in the system, which is proved later on. Given this, it only takes one robot on his way up to H to stall the system. We can ensure this will happen by putting in $h_{n-2} + \sum_{i=1}^{n-2} h_i + 1$ souls. We can now take H to be any number larger than this number, say $h_{n-2} + \sum_{i=1}^{n-2} h_i + 2$.

The power of the Collective. We now show that with less than H souls, the system cannot run indefinitely if not all robots are involved. Assume the system is running indefinitely with a minimal number of souls changing hands.

Once a robot starts up towards H, he can only take souls and never return them. By minimality, he never takes any new souls and might as well not be there. So we can assume that our subset of robots must be able to run indefinitely without anyone leaving towards H or giving souls to any robot going towards H.

First, we will assume that $\varphi_{\star,0}$ is missing. Assume a subset not including $\varphi_{\star,0}$ runs, then there is a minimal subset not including $\varphi_{\star,0}$ which runs.

Now, let m be the largest number such that $\varphi_{\star,m}$ is included in this subset. As the subset is minimal $\varphi_{\star,m}$ must complete a full cycle, as if it did not then we could simply leave robot m out. Thus, at some time robot m must have h_m souls. However, by the choice of the sizes of the peaks $h_m > \sum_{i=0}^{m-1} h_i$, it is clear that he can never get rid of them all without pushing one of the other robots onto its path towards H.

As there are a finite number of initial states (looking only at the robots below their peaks h_i), there is a global bound on the number of exchanges, L, which can occur before the system halts. T is chosen to be larger than L.

So we are left with the case when $\varphi_{\star,0}$ is included in the subset, but some $\varphi_{\star,m}$ is missing. As $\varphi_{\star,0}$ is included it must complete a full cycle (as otherwise it could be left out and we would have the previous case). Once it has reached the beginning of the series of peaks of height h_m it will, for at least the next T soul exchanges, behave exactly as $\varphi_{\star,m}$ did in the previous case. But this subsystem is guaranteed to stop before $\varphi_{\star,0}$ finishes its height h_m peaks, as for at least the next T transitions this subsystem behaves exactly as in the previous case. Thus $\varphi_{\star,0}$ will never complete its h_m peaks, and so never complete a full cycle.

Epilogue. We have now proved existence of a strategy. To actually apply this strategy, we would need to calculate precisely the value of several constants used in our algorithm. For instance, the constants T and M and the values at which we test for P_0, P_1 are hard to find, particularly within the hour that the warden has given us.

Acknowledgements and sources

The origins of this problem are not clear. According to legend [6, 7], similar problems have been the delight(bulb) of Hungarian mathematicians. The first written occurence we could find of the problem involving 100 prisoners and a unique lightbulb is on an online forum hosted at Berkeley [8]. Another variant involves 23 or 24 prisoners, two lightbulbs and the obligation for the prisoners to always change exactly one of the lightbulbs. This one appeared on the *Ponder This* website of IBM Research [7]. Those two online occurences, on the Berkeley forum in February 2002 and the IBM website in July 2002, were followed by several others, in either of the two versions. A thread was immediately started on `rec.puzzles` [5], it was published in the Fall issue of the Mathematical Sciences Research Institute newsletter *Emissary* [3], and the problem was finally asked

on the popular radio show *Cartalk* [6]!

We would like to thank Andrew Bennetts for introducing us to the original problem.

References

[1] Bertram Felgenhauer. 100 prisoners and a lightbulb. Newsgroup `rec.puzzles`, available through `http://groups.google.com`, July 28 2002.

[2] William Feller. *An introduction to probability theory and its applications. Vol. I*, pages 46,59. Second edition. John Wiley & Sons Inc., 1968.

[3] Mathematical Sciences Research Institute. Emissary newsletter, November 2002. Also available at `http://www.msri.org/publications/emissary/`.

[4] Renata Kallosh and Andrei Linde. Dark energy and the fate of the universe. 2003. `http://arxiv.org/abs/astro-ph/0301087`.

[5] "Oleg". 100 prisoners and a lightbulb. Newsgroup `rec.puzzles`, available through `http://groups.google.com`, July 24 2002.

[6] National Public Radio. Cartalk Radio Show. Transcription available at `http://cartalk.cars.com/Radio/Puzzler/Transcripts/200306/index.html`.

[7] IBM Research. Ponder This Challenge. `http://domino.watson.ibm.com/Comm/wwwr_ponder.nsf/challenges/July2002.html`, July 2002.

[8] William Wu. Hard riddles. `http://www.ocf.berkeley.edu/~wwu/riddles/hard.shtml#100prisonersLightBulb`, February 2002.