2 Prisoner Challenge

Written by: **Hirotaka Yoneda** & **Masataka Yoneda** (Japan), The University of Tokyo
Prepared by: Wiwit Rifa'i
Solutions, review, and other problem preparations by: Abdul Malik Nurrokhman, Jonathan Irvin Gunawan, Prabowo Djonatan
Analysis author: Wiwit Rifa'i

This problem is about generating a finite state machine to compare two different integers, A and B, with the minimum number of states.

2.1 Subtask 1

In this subtask, we can choose $x = N$. We can use the following strategy:

- The first prisoner will always read 0 on the whiteboard, and then overwrites it with the number of coins in bag A.
- The second prisoner will read a positive integer which represents the number of coins in bag A. So, the second prisoner has to check bag B and then answer which bag has fewer coins.

2.2 Subtask 2

Let $S = \lceil \sqrt{N} \rceil$. In this subtask, we can do the similar thing in subtask 1 but with 2 phases instead of 1 phase. First, we can compare $\lfloor \frac{A}{S} \rfloor$ and $\lfloor \frac{B}{S} \rfloor$. If the values are the same, then we can continue to compare ($A \bmod S$) and ($B \bmod S$). This solution only needs $x = 2 \times S$.

There's also another solution that needs $x = 3 \times S$ using the fact that every non-negative number can be represented uniquely as $p^2 + q$ for non-negative numbers p, q such that $q < 2 \times p + 1$. First, we can compare $\lfloor \sqrt{A} \rfloor$ and $\lfloor \sqrt{B} \rfloor$. Then, if the values are the same, we can compare $A - \lfloor \sqrt{A} \rfloor^2$ and $B - \lfloor \sqrt{B} \rfloor^2$.

2.3 Subtask 3

In this subtask, we can get some partial scores depending on how small the chosen x is. The following are some possible solutions that could get some points from this subtask.

2.3.1 Solution with $x = 3 \times \lceil \log_2 N \rceil - 1 = 38$

We can simulate how to compare two binary numbers. By noting that it requires up to 13 bits to represent a number up to 5000, the value on the whiteboard represents the current state, i.e.

- If the value is $3 \times d$, then we need to check the $(12 - d)$-th bit of A, and then overwrite it with $3 \times d + 1$ or $3 \times d + 2$ depending on the bit value.
- If the value is $3 \times d + 1$ or $3 \times d + 2$, then we know the $(12 - d)$-th bit of A is either 0 or 1 from the previous prisoner. Then, the current prisoner has to check the $(12 - d)$-th of B and compare the

current bit value between A and B. If the values are still the same, the prisoner can overwrite it with $3 \times (d+1)$ to continue to the next bit. Otherwise, the prisoner can answers which bag has fewer coins.

Therefore, this solution needs $x = 3 \times \lceil \log_2 N \rceil - 1$ or $x = 38$ for $N = 5000$.

2.3.2 Solution with $x = 2 \times \lceil \log_2 N \rceil = 26$

Instead of storing the bit only from A on the whiteboard, we can store the bit of either A or B alternately. We can use the parity of the current bit position i to determine whether we are storing the i-th bit of A or B. Therefore, we don't need an additional state for when we have not stored any value of i-th bit.

2.3.3 Solution with $x = 3 \times \lceil \log_3 N \rceil = 24$

We can compare them using base 3 instead of base 2 using a similar idea as the previous solution.

2.3.4 Solution with $x = 3 \times \lceil \log_3 N \rceil - 2 = 22$

We can prune two states when processing the last digit (in base 3 representation). If the last digit is 0 or 2, then we can be sure whether $A < B$ or $A > B$ without needing to check the last digit from the other bag since we know that $A \neq B$.

2.3.5 Solution with $x = 3 \times \lceil \log_3 (N+1) \rceil - 3 = 21$

In the previous solution, we use the pruning idea on the last digit only. If we use the ternary number system like the previous solution, we can visualize it as a perfect ternary tree where we divide range $[0, 3^k - 1]$ into 3 parts recursively. And, the pruning in the previous solution is only applied for some leaf nodes (last digits).

Instead of pruning only the last digit, we can apply it to any possible range. Let's say we know that both A and B are currently inside the range $[L, R]$, and we are checking bag A. If we know that $A = L$, then we immediately know that $A < B$. If we know that $A = R$, then we know $A > B$. It means that we can "prune away" the leftmost and the rightmost integer from the range. After that, we can divide the range $[L+1, R-1]$ into 3 parts and determine which sub-range A belongs to. Then, the next prisoner can check bag B and determine which sub-range it belongs to. If both bags still belong to the same sub-range, then we can continue the process with this smaller range. Otherwise, we can already determine which bag has fewer coins. We continue this process until we can determine the answer.

When we divide a range into 3 sub-ranges, we need 3 additional states. If the base range is a range with 2 integers, then $3 \times K$ states can cover $3 \times (3 \times (\ldots 3 \times (2) + 2 \ldots) + 2) + 2 = 3^{K+1} - 1$ numbers. So, we need approximately $3 \times K = 3 \times (\lceil \log_3 (N+1) \rceil - 1) = 21$ states for $N = 5000$.

2.3.6 Solution with $x = 20$

We can optimize the previous solution. Dividing a range by 3 is not always optimal. We can use dynamic programming to determine what is the best divider for each level such that it can cover the largest range.

$$\texttt{dp}[x] = \max_{1 \leq i \leq x} (i \times \texttt{dp}[x-i] + 2)$$

where $\texttt{dp}[x]$ denotes the maximum range that can be covered using x states and $\texttt{dp}[0] = 2$. For $x = 20$, we can cover until $N = 5588$. Such x is the minimum such that $N \geq 5000$. It can be achieved by dividing the range $[1, 5588]$ by 3, 3, 3, 3, 3, 2, 2, and 1 for each level recursively using the same process as the previous solution.