Guessing Game

EGOI 2023

Problem author: Edward Xiao.

Solution 1: $K = N - 1$ (10 points)

On the first $N - 1$ houses, Anna draws the numbers $1, 2, \ldots, N - 1$ in order of visiting. The number Emma draws must also be one of $1, 2, \ldots, N - 1$, and result in a duplicate value, and it will be the only duplicate value since all the other values are unique. Thus, we find the pair of houses that contains this duplicate value and guess those two houses.

Solution 2: $K = N/2$ (30 points)

Split the houses into two halves. For each half, we perform the same strategy as above. If either half contains a duplicate, we guess those. Otherwise, we guess the largest number in each half, since these were the last ones to be picked.

Solution 3: $K = \lceil N/3 \rceil + 1$ (36 points)

Note that we can actually improve on the strategy above by splitting the houses into thirds. For each third, repeat the strategy from $K = N - 1$ for all numbers except the last. For the last houses of the blocks, we pick two values x and y. Then, the strategy is simple again: if there are duplicates of x or y, guess those and return. Otherwise, there must be duplicates of the smaller values in one of the thirds, so we will guess those.

Solution 4: $K = \lceil \sqrt{N} - 1 \rceil + \lfloor \sqrt{N} - 1 \rfloor$ (60 points)

We can generalize the previous solution even further by splitting the houses into $\lfloor \sqrt{N} \rfloor$ blocks and picking a set X of $\lceil \sqrt{N} - 1 \rceil$ numbers for each number that is not the last in its block. For the last number in each of the $\lfloor \sqrt{N} \rfloor$ blocks, we assign them a different set Y of $\lfloor \sqrt{N} - 1 \rfloor$ numbers. If there is a duplicate of an element in Y, we guess those. Otherwise, there must be a duplicate of an element in X in one of the blocks, so we guess those instead.

Solution 5: $K = \lceil \log_2 N \rceil$ (90 points)

For 90 points, we will use divide and conquer to split the houses into multiple layers of blocks instead of just one layer. Consider a segment tree layout over the houses. For each index, we say it "completes" a segment in the segment tree if it is the last index in that segment to be assigned a value. In each round, Anna picks $v =$ the lowest depth of any segment that i completes. For Bertil, consider the values written on the doors assuming Anna picks the last value according to their strategy. In this case, Bertil can just find 1 and return its index. However, if there is no 1 in the array, then there are two cases:

1. The 1 was replaced by a 2. Then there are exactly two houses with 2s written on them, and we know that one of them must be Anna's, since there should only be one 2 if the strategy was followed for all indices. This is because the index that was picked later not only completes the segment on layer 2, but also necessarily completes the segment on layer 1 since the other half is already completed.
2. The 1 was replaced by a value greater than 2. Then, locate the half that contains the only 2 in the array. We know Anna's house must not be in this half, since it was completed first. Thus, we may recursively solve the problem on the other half, completing the solution.

Note that since 1 is never actually picked by Anna, we can subtract 1 from all values Anna picks to ensure we have exactly $K = \lceil \log_2 N \rceil$.

For illustrative purposes, consider an example where $N = 8$, Emma's house has index 2, and Emma and Anna visit the houses in the following order: $[0, 5, 4, 1, 6, 3, 7]$. Then, after Emma writes a number X on her own house, the array A will be $[3, 2, X, 3, 2, 3, 3, 1]$. We run through Bertil's strategy in the 3 possible scenarios:

1. $X = 1$: since there are duplicate 1s, it is clear that one of these must be for Emma's house. Guess $2, 7$ and return.
2. $X = 2$: since there is a unique 1, Emma's house must be in the half of the array that does not contain 1. Now we get to $A[0, 3] = [3, 2, 2, 3]$. Repeat our logic recursively. Since there are duplicate 2s, it is clear that one of these must be for Emma's house. Guess $1, 2$ and return.
3. $X = 3$: since there is a unique 1, Anna's house must be in the half that does not contain 1. Now we get to $A[0, 3] = [3, 2, 3, 3]$. Since there is a unique 2, Anna's house must be in the half of the array that does not contain 2. Now we get to $A[2, 3] = [3, 3]$. Since there are duplicate 3s, it is clear that one of these must be for Anna's house. Guess $2, 3$ and return.

Solution 6: $K = O(\log\log N)$ or $K = O(2^{\log^* n})$ (100 points)

To solve the problem fully, we consider the following two-phase approach:

In the first phase Anna just writes the number K on the first $N - \Theta(\log(n))$ houses which are visited. In the second phase (which we describe later), we promise that Anna will not write the number K on the remaining $\Theta(\log(n))$ houses.

Now we need to handle two cases:

- If Emma does not write the number K on her house, then we know it is one of the $\Theta(\log(n))$ houses not with a K.
- If Emma writes the number K, then we need to figure out which of the $N - \Theta(\log(n))$ houses with K written on it belongs to Emma.

In the former case, we have essentially reduced the problem to an instance with $N' = \Theta(\log(n))$, but now with numbers $1, \dots, K-1$.

For the latter case, we make the following observation: If we known the sum S of indices, modulo N, of the $\Theta(\log n)$ houses not part of the first phase, then we can figure out Emma's house in case she writes K. Indeed, we can identify that we are in the second case by counting the number of houses with K written on them. Then, we know that Emma's house index plus all indices of houses without a K, must equal S. Hence we can solve for Emma's house index.

The strategy for the second phase is now the following: we want to essentially solve an instance where $N' = \Theta(\log N)$ while simultaneously encoding the sum S (which is known at the start of the second phase).

One has to be a bit careful on how to encode the sum S, in the remaining $N' = \Theta(\log n)$ houses, since one of these houses we do not have control over and Emma can write K on it instead. One way of doing it is to write S (modulo N) in binary and then duplicate each bit, to make sure that we can recover the sum S even after one bit (the one corresponding to Emma's house) is dropped.

Solving the instance on $N' = \Theta(\log(N))$ in our strategy can be done in multiple ways:

- Using Solution 1 ($K = N' - 1$) for the recursive part, which will end up in ≈ 76 points with $K = \Theta(\log n)$.
- Recursively using our two-phase strategy here, which will result in ≈ 96 points and $K = \Theta(2^{\log^* N})$.
- Using Solution 5 ($K = \lceil \log_2 N' \rceil$), which happens to be quite good with small N. If implemented carefully this will obtain the full 100 points using $K = 7 = O(\log\log n)$, see below for details.

Detailed optimization tricks for 100 points:

- Use $K = 7$ and $N - 32$ numbers for the first phase (so $N' = 32$).
- In the second phase we run Solution 5 with $N' = 32$, which will use numbers $1, \ldots, 5$.
- Instead of encoding the sum S modulo N, we use the fact that we have two guesses and encode it modulo $N/2$. This requires 16 bits of information, since $2^{16} = 65536 > N/2$.
- Note that Anna will write exactly 16 "5"s in the second phase (see Solution 5 for more details). Hence we can change some of these "5"s to "6"s to encode the sum S.
- All in all, the solution becomes quite succinct: the jury's 100pt solution is just 40 lines of python code!

Harder Version: $K = O(1)$

It is possible to solve the problem with K being constant (that is not growing when N grows). The jury is aware of a solution using $K = 5$ no matter how large N is. We leave finding a solution with constant K (or even as low as $K = 5$) as a difficult bonus challenge.

The jury can also prove that using $K = 3$ is impossible (even for small N); so what about $K = 4$, is it possible or impossible?

How does the grader work?

The grader for guessinggame is a bit complex. It might run Anna's part twice to try to predict what would be a confusing number for Emma to write on her house.

In the grader, the order of houses is fixed per testcase, but the value Emma writes on her own house is chosen adaptively. The second line of each .in file explains how the value gets chosen for that test case:

- `random`: the last line of the input file contains a number X. The value is chosen as X
- `interact`: same as random, except that the judging is performed interactively, instead of non-interactively for performance. This is an implementation detail which should not matter. See includes/ for more details if curious.
- `copy`: the last line of the input file contains a number X. The value is copied from house with index X.

- `fork`: the last line of the input file contains a number X. Phase 1 of the submission is run as a trial run with Emma's house placed at (0-based) position X of the house visit order, and the house that was originally at position X removed. Let the number that Anna writes on that house be Y. Then in the real run of phase 1, Emma writes the number Y on her house.

  That is Emma, essentially asks Anna "if we had visited my house at time X, what number would you have written on it?".

The grading is performed by adding a file that gets compiled/run together with the submission and uses the fork() function to make it run multiple times.