Coding w/ Vihaan

A coding and computers channel.

USACO Feb 2025

Bronze

Problem 1 - Reflection

- Farmer John has a square canvas represented by an N by N grid of cells ($2 \leq N \leq 2000$, N is even). He paints the canvas according to the following steps:
  - First, he divides the canvas into four equal quadrants, separated by horizontal and vertical lines through the center of the canvas.
  - Next, he paints a lovely painting in the top-right quadrant of the canvas. Each cell of the top-right quadrant will either be painted (represented by '#') or unpainted (represented by '.').
  - Finally, since he is so proud of his painting, he reflects it across the previously mentioned horizontal and vertical lines into the other quadrants of the canvas.

Problem 1 - Reflection

- For example, suppose $N = 8$ and FJ painted the following painting in the top-right quadrant in step 2:

```
.#..
.#..
.##.
....
```

- Then after reflecting across the horizontal and vertical lines into the other quadrants in step 3, the canvas would look as follows:

```
..#..#..
..#..#..
.##..##.
........
........
.##..##.
..#..#..
..#..#..
```

Problem 1 - Reflection

- However, while FJ was sleeping, Bessie broke into his barn and stole his precious canvas. She totally vandalized the canvas—removing some painted cells and adding more painted cells! Before FJ woke up, she returned the canvas to FJ.

- FJ would like to modify his canvas so that it once again satisfies the reflective condition: that is, it is the result of reflecting the top-right quadrant into each of the other quadrants. Since he only has a limited number of resources, he would like to achieve this in as few operations as possible, where a single operation consists of either painting a cell or removing paint from a cell.

- You are given the canvas after Bessie's vandalism, as well as a sequence of U ($0 \leq U \leq 10^5$) updates to the canvas, each toggling a single cell to '.' if it is '#', or vice versa. Before any updates, and after each update, output the minimum number of operations x FJ needs to perform so that the reflective condition is satisfied.

- **More details: https://usaco.org/index.php?page=viewproblem2&cpid=1491**

Reflection - Solution

- We need to take advantage of the fact that updates only touch one square. Because exactly one square is touched per update, only four squares can possibly be impacted, namely the four squares that map to the updated one. We can temporarily ignore the optimal painted state for those four squares, perform the update, and recompute the answer.

Reflection - Implementation

- Can be found at https://github.com/VihaanAnand/Coding-Vihaan/blob/main/usaco-2025-02/bronze/1.cpp

Problem 2 - Making Mexes

- You are given an array a of N non-negative integers $a_1, a_2, \cdots, a_N$ ($1 \leq N \leq 2 \cdot 10^5$, $0 \leq a_i \leq N$). In one operation, you can change any element of a to any non-negative integer.
- The *mex* of an array is the minimum non-negative integer that it does not contain. For each i in the range 0 to N inclusive, compute the minimum number of operations you need in order to make the mex of a equal i.
- **More details: https://usaco.org/index.php?page=viewproblem2&cpid=1492**

Making Mexes - Solution

- For the mex of a to equal an integer i,
  - No element of a must equal i.
  - All of $0,\ 1, \cdots,\ i-1$ must appear in a.
- So the number of changes we need to make is at least the maximum of the following two quantities:
  - the number of elements of a equal to i (since we must change each element to a different value)
  - the number of non-negative integers less than i that do not appear in a (for each such integer, we must change an element to equal it).
- Conversely, it can be seen that as long as $0 \leq i \leq N$, the answer is exactly equal to the maximum of these two quantities. We can show this via the following strategy.
- While there are changes left to make, prioritize selecting an element equal to i to change if such an element exists, or else any element other than those less than i that appear exactly once in a. Then prioritize changing it to a non-negative integer less than i that doesn't already appear in a if such an integer exists, or else any integer greater than i. This strategy is guaranteed to decrease both quantities by one if they are positive or leave them at zero otherwise.
- To compute the first quantity, we can maintain an array of length $N+1$ storing the count of each value from 0 to N. To compute the second quantity, we can loop i from 0 up to N and increment it as necessary.

Making Mexes - Implementation

- Can be found at https://github.com/VihaanAnand/Coding-Vihaan/blob/main/usaco-2025-02/bronze/2.cpp

Problem 3 - Printing Sequences

- Bessie is learning to code using a simple programming language. She first defines a valid program, then executes it to produce some output sequence.
- Defining:
  - A program is a nonempty sequence of statements.
  - A statement is either of the form "PRINT c" where c is an integer, or "REP o", followed by a program, followed by "END," where o is an integer that is at least 1.
- Executing:
  - Executing a program executes its statements in sequence.
  - Executing the statement "PRINT c" appends c to the output sequence.
  - Executing a statement starting with "REP o" executes the inner program a total of o times in sequence.

Problem 3 - Printing Sequences

- An example of a program that Bessie knows how to write is as follows.

```
REP 3
    PRINT 1
    REP 2
        PRINT 2
    END
END
```

- The program outputs the sequence $[1, 2, 2, 1, 2, 2, 1, 2, 2]$.

- Bessie wants to output a sequence of N $(1 \leq N \leq 100)$ positive integers. Elsie challenges her to use no more than K $(1 \leq K \leq 3)$ "PRINT" statements. Note that Bessie can use as many "REP" statements as she wants. Also note that each positive integer in the sequence is no greater than K.

- For each of T $(1 \leq T \leq 100)$ independent test cases, determine whether Bessie can write a program that outputs some given sequence using at most K "PRINT" statements.

Printing Sequences - Solution

- **Subtask 1:** $K = 1$
  - When $K = 1$, the answer will always be "YES" since the sequence will contain all 1s due to the constraint that all elements of the sequence have to be at most K.

Printing Sequences - Solution

- **Subtask 2:** $K = 2$
  - When $K = 2$, all sequences can be outputted with code that is described by the following format.

```
REP outer
    REP o1
        PRINT x
    END
    REP o2
        PRINT y
    END
END
```

  - One way is to approach this is to store information about blocks of same numbers. For example, $[1, 1, 1, 2, 2, 1, 1, 1, 2, 2]$ can be broken down into the following.
    - Three 1s
    - Two 2s
    - Three 1s
    - Two 2s
  - From here we can see that a sequence can be outputted by the code if both the following is true.
    - The number of blocks is even or ≤ 2.
    - Block i is the same as block $i + 2$ for all i.

Printing Sequences - Solution

- **Subtask 3:** $K = 3$
  - We can extend the reasoning for $K = 3$. Let a d-**degree** sequence be some sequence that can be outputted with at most d "REP"s. We can reduce the $K = 3$ problem into the following cases.

    ```
    Case 1
    REP outer
        <Program that outputs a degree 1 sequence>
        <Program that outputs a degree 2 sequence>
    END

    Case 2
    REP outer
        <Program that outputs a degree 2 sequence>
        <Program that outputs a degree 1 sequence>
    END
    ```

  - The inside of the outer "REP" will make up some repeating block of the sequence. We can iterate over each prefix and check if it is a repeating block. If so, we check whether it can be the output of the body of the outer "REP". This means that we need to check if it is the concatenation of a degree 1 and degree 2 sequence or vice versa.
- Checking can be done by iterating over the dividing point of the degree 1 and 2 sequences and seeing if the sequences on the left and right of the dividing point have the desired degrees. We can use our solution from the previous subtask to help us do this.

Printing Sequences - Implementation

- Can be found at https://github.com/VihaanAnand/Coding-Vihaan/blob/main/usaco-2025-02/bronze/3.cpp

Coding w/ Vihaan

4.3B subscribers

Subscribe

65.6K

Share

16.8M views 1 second ago