2025 South Conference Division 2 Solutions

The Judges

November 8, 2025

Snakey String

Problem

- A "snakey string" is an n by m grid of tiles constructed from a normal string such that every column contains exactly one character and the ith column contains the ith character of the string.
- Given a snakey string, output the original string.

Solution

- A column-major traversal of the 2D grid will yield the original string.
- Iterate over columns from left to right. For each column, search every tile for a character. Once the character is found, print it out.

Blind Bottles

Problem

- There is a list of n unique bottles arranged in an unknown order. You make a guess of their order and receive the number of bottles that are correctly placed. Your task is to win this *Blind Bottles* game in 10^4 guesses.

Observation

- The limit of 10^4 guesses is large enough to make $O(n^2)$ guesses. If we can determine one bottle's correct position using $O(n)$ guesses, then we can solve the full problem by repeating this process n times.
- We can establish a baseline first by guessing an arbitrary ordering. Suppose the baseline has c correct positions. Then we can try determine which bottle is at the first position by swapping it with each of the other bottles, and checking if we get c, $c \pm 1$ or $c \pm 2$. Let's call this a *round*.

Blind Bottles

Solution

- We will name the positions $p_1, \ldots, p_n$, and the bottles $b_1, \ldots, b_n$. If all bottles are placed correctly, then b_i is at p_i for all i.
- In the current round, we are trying to find b_1 and move it to p_1. We swap the bottle currently at p_1 with every other bottle and record their number of correctly placed bottles.
- We have these cases :
  - b_1 is already at p_1: we will only get $c-1$ or $c-2$.
  - b_1 is not at p_1. Instead, b_1 is at p_k:
    - b_k happens to be at p_1: we will get $c+2$.
    - Some other bottle b_t is at p_1: we must get exactly two $c+1$ at two positions p_x and p_y, because one swap will get b_t into p_t, and another will get b_1 into p_1. Now what we do not know is which $c+1$ swap is the one that corrects b_1 (i.e. is $p_k = p_x$ or $p_k = p_y$?). We can resolve this with one extra guess – after exchanging the bottles at (p_1, p_x) and then at (p_x, p_y).

Blind Bottles

Solution (cont'd)

- After the first round, b_1 will be at p_1. We then proceed to the next round to place b_2 at p_2.
- Repeat the rounds until all bottles are placed correctly.
- It can be seen that this process uses one initial baseline guess, plus $(n-1)+(n-2)+\cdots+1 = n(n-1)/2$ guesses in the rounds. The extra guesses for the two $c+1$ case can at most double the guesses to $n(n-1)$. The total number of guesses is thus at most $1 + n(n-1) < n^2$.

One Way Only

Problem

- You are given an r by c grid and a path starting at the top left, ending on the bottom right, and only consisting of downwards and rightwards steps.
- You wish to determine the minimum number of tiles you must mark such that your given path is the only path consisting of downwards and rightwards steps that goes through zero marked tiles.

Observation

- This path splits the grid into two regions (above and below), with each region being the transpose of the other. So it suffices to write a solution to block out all the paths that go above our given path, and repeat the same steps for paths that go below.

One Way Only

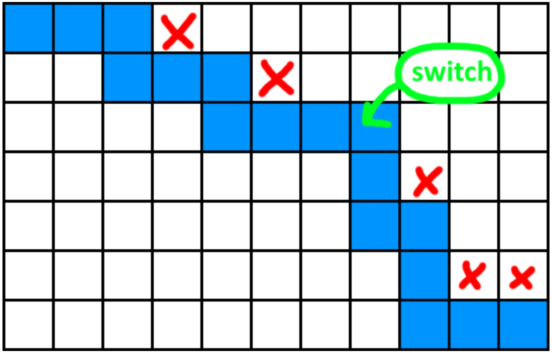
Observation

- We will focus on blocking off all paths that go above the given path.
- First, observe that it's never optimal to X out a tile that is not directly above or to the right of a tile on the input path.
- Furthermore, if and only if there exists two tiles a and b on our path such that a comes before b on the path, a has an empty tile to the right, and b has an empty tile above it, then there exists another path on our grid which does not go through a marked tile.
- So, our goal is to minimize the number of marked tiles while preserving the above condition.

One Way Only

Observation

- To do this, observe that we can iterate through the path, marking all the tiles immediately to the right of the path. At exactly one point, we can switch to marking out all the tiles immediately above the path instead.



One Way Only

Solution

- But where does this switch happen? To calculate this, store two arrays A and B.
- For A, for each tile on the input path, store the number of tiles that would get marked out if we were to mark out the tile immediately to the right of every tile that comes before this tile.
- For B, for each tile on the input path, store the number of tiles that would get marked out if we were to mark out the tile above every tile that comes after this tile.
- Then, our solution is the minimum of $A[i] + B[i]$, where i is a tile.

GUID Generator

Problem

- You are given a tree of n nodes in which each node has a hexadecimal character. Count the number of unique strings you can get by walking along any simple path on the tree and concatenating all the hexadecimal characters along the path.

Observation

- There are $O(n^2)$ different paths, and each path produces a string of length n. The total length of all these strings is $O(n^3)$. It would be too slow and take too much space to generate all these strings.
- Consider all the paths that start from the same node s, walking along those paths naturally corresponds to inserting those strings, starting from s, into a Trie (prefix tree).

GUID Generator

Observation (cont'd)

- The Trie also helps us deduplicate the strings. The size of the Trie gives us the count of unique strings.
- For a different starting node s', we can reuse the Trie that was built by starting from s earlier.

Solution

- For each node s, traverse the tree starting from s, while inserting every hexadecimal character encountered into a Trie. Each node in the Trie can have at most 16 children. The size of the Trie minus 1 is the final answer.
- Time and Space: $O(n^2)$.

GUID Generator

Alternative Solution

- We can also compute a hash value for the string we have while walking along the tree paths.
- A standard polynomial hash will work:
  $Hash(Sc) = (Hash(S) \cdot a + Hash(c)) \bmod p$.
  - S is a string, c is a character, and Sc is the result of concatenating S and c.
  - a and p are two constant integers that are coprime, with p being a prime.
- Counting the unique hash values at the end gives us the final answer.

Photo Encoding

Problem

- Given a list of integers, output the minimum n such that every integer x_i can be matched to a tile in an n by n image of which the Manhattan distance to the top-left tile is equal to x_i.

Observation

- For an n by n image, we notice that, for some distance d, there are exactly $(n - 1 - |n - 1 - d|)$ tiles in said image that have a distance of d to the top-left tile.

Solution

- We count the number of occurrences of each Manhattan distance, and use that to bound our n. If we have x occurrences of some distance d, then $(n - 1 - |n - 1 - d|) \geq x$.

Photo Encoding

Solution

- To solve this problem, first count the number of occurrences of each distance, and use the above equation to bound n. Once all inequalities are satisfied, we have our solution!
- The bounds are small enough to simply check every possible value of n against every inequality (we can show that n is always less than 2000). Alternatively, you can manipulate the inequality above to solve for n to solve the inequality instantly.

Breakout

Problem

- You are given a circular hallway with n rooms on its circumference. You wish to move from room 1 to a specified exit room by walking along the edge of the circle.
- Some pairs of adjacent rooms have boxes between them. You wish to break the minimum number of boxes to get to the exit.

Solution

- We pick one direction (clockwise or counterclockwise) to traverse the rooms until we reach the exit. There is no reason to turn around while traversing.
- Our answer is therefore the minimum number of boxes destroyed when picking one of the two directions described above.

Breakout

Solution

- Say our target room is r. When traveling clockwise, we will destroy all boxes between rooms 1 and r. This corresponds to the integers between 1 and $r - 1$ in the input.
- When traveling counterclockwise, we will destroy all of the remaining boxes.
- So our solution is the minimum of the number of boxes in the hallways labeled 1 to $r - 1$, and the number of boxes in the hallways labeled r to n.

Shinjuku Station

Problem

- You are given $n \leq 50$ train arrival times and $m \leq 50$ train departure times. Find the minimum transfer time that is at least s seconds between any pair of arriving and departing trains.

Solution

- Since n and m are both small, we can enumerate all pairs of arriving and departing trains and check the time difference between them.
- It is recommended to parse every time into three integers and convert them into a total number of seconds, e.g. in Python you can do

```
h, m, s = map(int, line.split(':'))
seconds = h * 3600 + m * 60 + s
```

- Time: $O(nm)$

Memories of Passport Control

Problem

- There are s passport stamps that you received. You know that each trip gives you either k stamps or 1 stamp. Determine the minimum number of trips possible.

Observation

- To minimize the number of trips, we want to assume that we have as many k-stamp trips as possible.

Solution

- The answer is simply $\left\lfloor \frac{s}{k} \right\rfloor + (s \bmod k)$.

Inazuma Cube Device

Problem

- You are given a DAG where edges represent chain reactions between the cubes.
- Since for any chain reaction from cube i to cube j we have $i < j$, it means that you can't have any circular relations.

Observation

- Since we are dealing with a DAG, we know it doesn't hurt us to work on the cubes with 0 indegree first. We have solve those cubes first so that they have k petals lit.

Inazuma Cube Device

Solution

- Strike any cube with zero indegree until they all have k petals lit. Propagating these hits to the cubes they point to as per the chain reactions.
- After getting each cube with zero indegree to have k petals lit, remove it from the graph and all its edges.
- If your solution is failing, keep in mind that the petals range from 1 to k but if you mod by k you can get the petals to range from 0 to $k - 1$. One way of handling this is to just subtract 1 from every cube's petal count at the start.

Game of Nines

Problem

- You have a list of n digits. You can add one digit to the other, and replace it with their sum mod 10. Any sum that is 9 is eliminated. Determine the maximum number of digits that can be eliminated.

Observation

- If we can obtain at least one digit 1, then we can use 1 to eliminate all the other digits by repeatedly adding 1 to the other digits until they become 9.
- It can be shown either by case work or number theory that you can obtain a digit 1 except for three cases:
  1. All digits are even.
  2. All digits are multiples of 5 (mod 10): i.e., 0 or 5.
  3. All digits are multiples of 3 (there exists at least one 3 or 6).

Game of Nines

Solution

- In cases 1 and 2, we cannot eliminate any digits. The answer is n.
- In case 3, we can add one 3 to all the other digits repeatedly until we eliminate all of them. The answer is 1.
- In all other scenarios, we can obtain a digit 1 and eliminate everything else. The answer is 1.

Swap for Palindrome

Problem

- You are given a string of $n \leq 5\,000$ characters. You are to make exactly one swap of two characters so that the resulting string has the longest palindromic substring.

Swap for Palindrome

Observation

- Since $n \leq 5\,000$, we can afford enumerating all palindromic substring centers and greedily fixing any mismatches going outward. This will take $O(n^2)$ time.
- Let `c` (or `cc`, when the substring length is even) be the center of the palindromic substring.
- (Case 1) The best mismatch we can resolve with one swap looks like this: `..[a..b..c..a..b]...` In this case we can swap `a` and `b`.
- (Case 2) If we have `..[a..c..b]..`, we can also try to find an `a` or `b` outside `[..]`, or from the center (don't forget `c` when `c` $=$ `a` or `c` $=$ `b`).
- After resolving the mismatch with one swap, we shall continue extending the palindrome until another mismatch, which we won't be able to fix.

Swap for Palindrome

Solution

- Enumerate the palindrome centers (separate odd/even lengths).
- Find the first two pairs of mismatches `..[s..p..c..q..t]..`, i.e. everything matches in `[..]` except that $\mathtt{p} \neq \mathtt{q}$ and $\mathtt{s} \neq \mathtt{t}$.
- Check if we can fix the mismatches using the cases discussed previously. After any fix, continue extending the palindrome until another mismatch.
- Two common pitfalls:
  - Must pay special attention to the center character (`c`) which can also participate in the swap.
  - The swapped letter in Case 2 to fix the (`p`, `q`) match could be between `s..p` or `q..t`, which changes where the palindrome can extend to.

Moonlit Time Machine

Problem

- You have a time machine that projects you d days into the future, where d is unknown. To find d, you use the time machine n times, and record the phase of the moon after each use.
- Each phase of the moon is present on specific known days. Therefore, you wish to find such a d that the n uses of the time machine match up with your recordings.

Observation

- The size of n and d is very small, so we can get away with directly simulating each possible value of d.

Moonlit Time Machine

Solution

- Simulate the recordings for each possible value of d ($1 \leq d \leq 28$), starting at a new moon.
- Once a value of d whose recordings match up with the input is found, that is the solution.

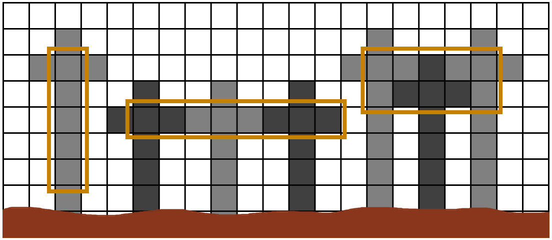
Much Room for Mushrooms

Problem

- You wish to determine if it is possible to completely fill an r by c region with specially defined "mushrooms".

Observation

- We can observe three types of regions that we can fill with mushrooms, as shown in the diagram below.



Much Room for Mushrooms

Solution

- Check which of the above cases we have.
- For X by 1 regions, we can use a single tall mushroom.
- For 1 by X regions, we need $\lceil \frac{X}{3} \rceil$ mushrooms placed side by side at the same height.
- For 2 by X regions, we need $\lfloor \frac{X}{2} \rfloor + 1$ mushrooms placed at an alternating heights.
- All other cases are impossible.