International Collegiate Programming Contest (ICPC)
Asia Regional Dhaka -2024
Online Preliminary Contest

Hosted By

Department of Computer Science and Engineering,
Daffodil University

Supported by

Daffodil International University

Problem A

Adhoc King

Problem setter: Shahjalal Shohag

Alternate writer: Rumman Mahmud

Analysis:

The product is hard to control, so let's try to fix the product. But sadly 1 is not allowed, so we can't fix the product. But for now, let's imagine a_i can also be 1.

Let $k = ceil(log_2(n))$. Then a possible solution is: $2^k - 1,\ 1,\ 1$,..., 1.

It works because the prefix product is constant $2^k - 1$ and it has all on bits and the prefix sum every time increases by 1 (that is, the prefix sum sequence is $2^k - 1,\ 2^k,\ 2^k + 1,\ 2^k + 2,\ 2^k + 3$,....), so the bitwise ANDs of the prefix products and prefix sums are $2^k - 1,\ 0,\ 1,\ 2,\ 3$,..... Hence the bitwise ANDs are distinct.

But wait, 1 is not allowed, so what can we do? Actually, it's not hard to see that if we just change 1 to $2^k + 1$ the solution still works! Why? Think for a minute, it's left as an exercise for you.

So the solution is $2^k - 1,\ 2^k + 1,\ 2^k + 1$,..., $2^k + 1$

Problem B

Network Disentanglement

Problem setter: Muhiminul Islam Osim

Alternate writer: Hasinur Rahman, Shahjalal Shohag

Analysis:

We can convert the input data into a new graph where the wire links are the nodes of the new graph. There exists an edge between two nodes if the wire links represented by the nodes cross or touch each other and the wire links are of different ISPs. Now, our task is to find the minimum vertex cover set of the graph. We can observe that the graph is actually a bipartite graph, which means that we can find the minimum vertex cover set by finding the maximum matching from the graph. There are multiple algorithms to find the maximum matching. After that, we can use Konig's theory to find the minimum vertex cover set from the maximum matching.

Overall Time Complexity: $O(T \cdot M \cdot M \cdot \sqrt{M})$ (using the Hopcroft-Karp algorithm to find the maximum matching). But the total operations will be at most $T \cdot (\frac{M \cdot M}{4}) \cdot \sqrt{M}$ as the total number of edges of the new graph doesn't exceed $(\frac{M \cdot M}{4})$.

Problem C

Watering the Flowerbeds

Problem setter: Ashraful Islam Shovon

Alternate writer: Shahjalal Shohag, Nafis Sadique

Analysis:
First of all, the flowerbeds are independent. For each flowerbed, we should only water it when we need to or on the last day otherwise the flowerbed would die. As such, simply dividing K by the number of days needed is enough. Some simple checks are required for impossible cases.

Problem D

Red Apple, Green Apple

Problem setter: Nafis Sadique

Alternate writer: Pritom Kundu, Shahjalal Shohag

Analysis:

What happens if each box has only a Red apple? The expected weight of picking K apples would be $AVERAGE\ WEIGHT \times K$. Where $AVERAGE\ WEIGHT = \frac{SUM\ OF\ ALL\ APPLE\ WEIGHTS}{N}$.

With red and green apples, we need to take the average weight of the apples in the box as the weight of the box (since either apples can be chosen in the same probability). So, we basically take the average of the average.

Assume $S = \frac{\sum_{1}^{N} RedApple_i + GreenApple_i}{2}$ or the sum of the average weight of all boxes. Their average is $\frac{S}{N}$.

So, the answer for choosing K apples would be $\frac{S \times K}{N}$.

Problem E

Master Evenius and Oddius

Problem setter: Ashraful Islam Shovon

Alternate writer: Shahjalal Shohag, Rumman Mahmud, Nafis Sadique

Analysis:
Oddious always wins if they start the game unless there is only one stone at the beginning. Evenious wins when they start only if the remainder of N divided by 4 is 1. This can be proven mathematically. However, even without a formal mathematical proof, you can solve this problem by using dynamic programming for smaller ranges and identifying the pattern.

Problem F

Three Quick Brown Foxes Jump Over a Lazy Chicken

Problem setter: Md Hasinur Rahman

Alternate writer: Shahjalal Shohag, Rumman Mahmud

Analysis:
The solution to this problem involves finding the radius of the incircle of the triangle. This is because the shortest distance from the trap to a side of the triangle is the perpendicular distance. Since all three foxes need the same jump distance, each of these three distances must be equal and perpendicular to the sides.

Time complexity: **O(T)**
Memory complexity: **O(1)**

Problem G

Travel on the grid

Problem setter: Raihat Zaman Neloy
Alternate writer: Pritom Kundu

Analysis:
As the grid size is only (7 x 7) and no single cell is attacked by more than one cell, so there can be at most 10 bombs in the grid. Considering this, we can keep a 3-based mask while keeping if a bomb is active, diffused and has a diffuser on its cell. Now, running a regular dijkstra with a complexity of O(NxMx3^(number of bombs)) should be enough for us to find the cheapest path.

Better solution:
Let's say each bomb only blocks its own cell. Then we can do a dijkstra with the states (x, y, b) where b =1 iff we are carrying a diffuser. However, in our problem each bomb blocks its four adjacent cells as well. How to handle that? If we have reached a cell with a bomb, we can assume that it is diffused and visit any of our neighbors without worry. It turns out only handling this case is enough to solve the problem!

Let (x, y) be the cell we are currently in and (x', y') be a neighboring cell. We simply run a dijkstra over all states (x, y, b) with the following transitions

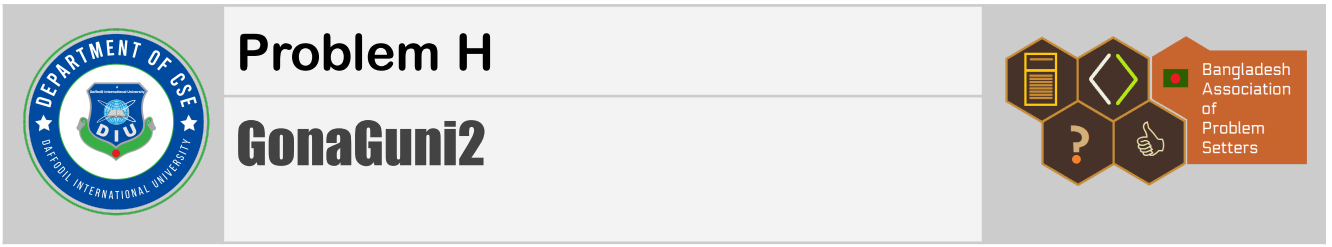
- If (x, y) and (x', y') are both safe
  - (x, y, 0) → (x', y', 0) (no cost)
  - (x, y, 1) → (x', y', 1) (cost a[x][y])
- If (x, y) is safe and (x', y') has a bomb
  - (x, y, 0) → (x', y', 0) (cost a)
  - (x, y, 1) → (x', y', 0) (cost b)
- If (x, y) has a bomb
  - (x, y, 0) → (x', y', 0) (no cost)
  - (x, y, 0) → (x', y', 1) (cost a[x][y])

Clearly, these transitions are valid. But it turns out these are sufficient. Why? We can make a few observations.

1. If we diffuse a bomb, we can assume that we move onto its cell next move.
2. We can assume that we only build a diffuser right before using it on a bomb.
3. If we visit a cell with a diffuser, we will never visit it again.
4. If we visit a cell without a diffuser, we can only visit it again with a diffuser.
5. If we ever visit a blocked cell neighboring a bomb, we can assume that we visit it straight after diffusing the said bomb.
6. We only ever visit at most one neighboring cell of any bomb.

The exact proof is left as an exercise! The time complexity is O(mn log (mn))

Fun fact: Neither the author or the alter saw this solution and panicked when contestants found this "unintended solution". They were initially afraid an wrong solution may have passed due to weakness of judgedata, but were able to prove the correctness of the solution eventually. Big props to all the teams finding the optimized solution!



Problem setter: Md Mahamudur Rahaman Sajib

Alternate writer: Nafis Sadique

Analysis:
Although the number of columns is small, if we try to do dp having the previous row as the state it will be too slow. Instead we can classify/normalize the row patterns. Multiple patterns can fall into the same group, if we do some kind of normalization and for each group we can calculate solutions separately.

for example row = [7, 7, 1, 1, 4, 1, 4], we can normalize it as [1, 1, 2, 2, 3, 2, 3]
More specifically, we will take distinct numbers order of their first occurence, for this case [7, 1, 4] and then map them to their order. For this case, 7 will be mapped to 1, 1 will be mapped to 2 and 4 will be mapped to 3. Benefit of this normalization is that more patterns can fall into the same class or normalized pattern. For example, [2, 2, 3, 3, 5, 3, 5] also falls under the same category. Interesting thing is that for M = 7, the number of normals is 203.

Now let's think about the previous dp, if we think about keeping the previous row as our state then patterns that fall under the same class will provide us the same answer.
But if we still do dp it will be slower as we need to iterate on different patterns for the current row. So we can think of this way that we are moving from one normalized pattern to another normalized pattern. For the previous row having normalized pattern i, how many patterns we can put in the current row which falls under normalized pattern j. This way we can create a coefficient matrix and do matrix exponentiation.