DIU Unlock The Algorithm Programming Contest
Main Round, Spring 2025

Hosted by

Department of Computer Science and Engineering
Daffodil International University

In Association with

Date: Apr 27, 2025 9:00 AM

Table of Contents:

Problem Set: Link
Contest: Link
Problem solution: Link

A. Robot

Setter: Piyash Basak
Tester: Kh Sadman Sakib
Category: Graph (BFS)
Total Solve: 0
First to solve: N/A

Problem Summary:

- You are given a $N \times M$ grid with land (.) and sea (#) cells.
- K robots start at different land positions.
- Each robot can move to adjacent land cells (up, down, left, right).
- The goal is to determine, for each query, the minimum time for any remaining functional robot to reach (N, M) after disabling a specified robot.

Key Observations:

1. **BFS from Destination**:
   - Since all robots need to reach (N, M), we compute the shortest path from (N, M) to every other land cell using **BFS**.
   - This is efficient $O(N, M)$ and optimal for unweighted grids.
2. **Handling Robots**:
   - For each robot at $\left(x_i, y_i\right)$, its travel time is $dist[x_i][y_i]$ (precomputed via BFS).
   - If a robot is in a sea cell or unreachable, its distance remains **INF**.
3. **Efficient Query Processing**:
   - For each query, we need the **minimum distance among all active robots**.
   - A **multiset** is used to maintain these distances, allowing O(logK) insertions, deletions, and minimum queries.

B. AC Installation Plan

Setter: Sourov Biswas
Tester: Wahidul Islam Payel, Moniruzzaman Nahid
Category: Greedy, Math, BS
Total Solve: 40
First to Solve: IMRANUL ISLAM SHIHAB

We need to find out how many ACs DIU can install without crossing the electricity budget.

Each AC uses a fixed number of units.
Electricity cost increases as more units are used:

- 1–50 units → 10 taka/unit
- 51–80 units → 15 taka/unit
- 81–100 units → 20 taka/unit
- 101+ units → 25 taka/unit

Solution 1 — Binary Search

Idea: If k ACs work, k-1 ACs also work.

- Check if mid number of ACs fits in the budget.
- If yes, try more ACs.
- If no, try fewer ACs.
- **Time Complexity:** O(log k), k being the answer

Solution 2 — Math Trick (O(1)) [Super fast, just a few calculations.]

- The first 100 units cost 1350 taka.
- After that, each extra unit costs 25 taka.
- So after 1350 taka, every 25 taka = 1 unit.

So:

- Subtract 1350 from the budget.
- Calculate how many extra units we can afford.
- Find the number of ACs directly.

Solution 3 — Greedy [Simple and easy to think]

- Keep adding one AC at a time.
- Deduct cost based on which range of unit usage we're in.
- After 100 units, just divide the remaining budget by 25 taka/unit.

Summary:

- **Binary search** — Standard method (safe, fast).
- **Math** — Smart shortcut (fastest, O(1)).
- **Greedy** — Easy to understand.

C. Know your Siblings

Setter: Saimum Islam Hamza
Tester: Saimur Rahman Robin
Category: Graph/Trees
Total Solve: 2
First to solve: A.K.M. Shohan

- Use DFS or BFS from node 1 to determine each node's parent.
- Count how many children each node has.
- For node x, the number of siblings = the number of children of parent[x] - 1.

D. Narcissistic Number

Setter: Mr. Md. Ferdouse Ahmed Foysal
Tester: Kh Sadman Sakib
Category: Giveaway
Total Solve: 136
First to solve: Md. Mostafizur Rahman Antu

Suppose you think of $1^1 = 1$, which is ok.
But it says ***non-negative***, so $0^1 = 0$ it is the smallest answer.

E. Area Recovery

Idea: Presh Talwalkar (Mind Your Decision)
Setter: Sharif Hussain
Tester: Anup Barman, Kh Sadman Sakib
Category: Geometry
Total Solve: 22
First to solve: MD JAHID HASAN

Let, $AB = CD = M$ and $AD = BC = N$
Given that $BE = A$ and $DF = B$ and Area of $AEF = X$

So, $CF = M - B$ and $CE = N - A$
Now, Area of $ABCD$ = Area of ABE + Area of ADF + Area of CEF + Area of AEF
=> $M \times N = (0.5 \times BE \times AB) + (0.5 \times AD \times DF) + (0.5 \times CE \times CF) + X$
=> $2 \times M \times N = A \times M + B \times N + (M - B) \times (N - A) + 2 \times X$
=> $M \times N = A \times B + 2 \times X$
So, Area of $ABCD = A \times B + 2 \times X$

F. Minimize Difference

Setter: Moniruzzaman Nahid
Tester: Sourov Biswas, Kazi Amir Hamza
Category: Sorting/Searching
Total Solve: 9
First to Solve: Md Abdul Quym Shanto

When you pick three numbers x, y, z, the maximum difference is $max(x,y,z)-min(x,y,z)$. To solve this problem, we have to sort the array and compute $arr[i+2]-arr[i]$ for every i from 0 to $n-3$. The answer is the minimum among all values.

G. MindSync Neural System

Setter: Fahim Khandaker
Tester: Kh Sadman Sakib
Category: DP
Total Solve: 0
First to solve: N/A

1. Problem Restatement

We have an $m \times n$ grid of digits (0 to 9).

- **Row requirement:** In each row, all cells must have exactly the same digit.
- **Column requirement:** In each column, adjacent rows must have *different* digits.

We want to change as few digits as possible to achieve these two requirements.

2. Key Insight: Reduce to Row Choices

Since each row must be constant, we only need to decide *one* digit per row.

- If we choose digit d for row i, then we must change every cell in row i that is not already d.
- The cost to make row i all d is: $COST_{i,d} = n - freq_{i,d}$

The column constraint becomes: if row i picks digit a and row $i+1$ picks digit b, then $a \neq b$
Thus, we have turned the grid problem into: **choose one digit per row** (out of 10), with a cost for each choice, and forbid choosing the same digit in consecutive rows.

3. Dynamic Programming Formulation

Let $dp[i][\text{last}]$ be the minimum total cost to fix rows $i,\ i + 1,\ i + 2, \cdots, m - 1$, given that row $i-1$ (the previous row) used digit $last$.

- **State variables:**
  - i — the current row index (0-based).
  - $last$ — the digit chosen for the previous row (or a special value when $i = 0$).
- **State transition:** For row i, we try all 10 possible digits $d \neq last$
  The cost is:

$$dp[i][\text{last}] = \min_{\substack{0 \le d \le 9 \\ d \neq \text{last}}} \{\text{cost}[i][d] + dp[i+1][d]\}$$

- **Base case:** $dp[m][*] = 0$ since if we've processed all rows, there is no further cost.
- **Answer:** We call $dp[0][sentinel]$, where sentinel is any value not in 0, 1,..., 9, for example, 10 or 11.

4. Precomputing Row Costs

Before the DP, for each row i and each digit d, count how many times d it appears in that row.

- You can read the input row by row and maintain a frequency array $cnt[i][0 \cdots 9]$.
- Then compute $cost[i][d] = n - cnt[i][d]$ at $O(1)$per digit.

5. Time and Memory Complexity

- **Time:** Filling dp takes $O(m \times 10 \times 10)$ since both 11 and 10 are constants.
- Precomputation of costs takes $O(m \times n)$.
- Overall: $O(m \times n)$.

H. Frequency Showdown

Setter: Saimur Rahman Robin
Tester: Saimum Islam Hamza
Category: Implementation
Total Solve: 16
First to Solve: Md Abdul Quym Shanto

1. **Count Frequencies**:
   - Use a map to count the occurrences of each element in the array.
2. **Store in a Vector**:
   - Store the frequency and element pairs in a vector, i.e., {frequency, element}.
3. **Sort the Vector**:
   - Sort the vector primarily by frequency in descending order.
   - For ties, sort by the element's value in ascending order.
4. **Print Top K Elements**:
   - Print the first K elements from the sorted vector.
   - If fewer than K distinct elements are available, print all of them.

Time Complexity:

- Counting frequencies: O(N)
- Sorting: O(D log D), where D is the number of distinct elements
- Total: **O(N + D log D)**

I. Yaaay, Party!

Setter: Kazi Amir Hamza
Tester: Sourov Biswas
Category: Greedy
Total Solve: 5
First to solve: Esrat Anam Kamy

Observation:

- When a friend arrives at moment x, he occupies a chair until he leaves.
- When a friend departs at moment y, the chair will be available at $y+1$
- The Maximum number of friends present at any moment is the number of chairs needed.
- Monika already has 2 chairs

Solution idea:
Creating a ***timeline*** array will help keep track of the number of friends at any moment. When someone arrives at moment x, the number of friends from moment x to moment y increases by 1.
timeline[x]++;
timeline[y+1]--;

Now find the maximum number of friends present at any moment on the timeline. That's the number of chairs Monika needs(She already has 2, 1 is for herself)
So, the answer will be $max(0, max\_friend - 1)$