A

A. A day in the woods

You are given a directed graph with a specified source s. Find a numbering of all vertices such that the source is numbered 1, and numbers along any simple path starting at s are increasing.

A. A day in the woods

Consider a directed edge (v, u), and suppose that there is a path from s to v evading u. Clearly, this means that in any suitable numbering the number of v should be smaller than the number of u.

If, on the other hand, every path from s to v contains u, then (v, u) does not lie on any simple path, and can be safely ignored.

We now have a working idea: erase all the edges (v, u) such that there is an $s \to v$ path evading u, and topologically order the resulting graph.

How to find all useless edges? Try all options of u, for each option erase it from the graph and determine which v become unreachable.

The total complexity $O(nm)$.

B

B. Bus Stops

There are s bus stations, and b bus route descriptions (for each station, `1` if it belongs to the route, and `0` otherwise). Find the number of stations directly reachable (via a bus route without changing) from the station m.

Consider only routes that contain the station m, unite the lists for all these routes.

C

C. Campus for SIT

There is an $n \times m$ grid, each cell containing a forest or a lake. Place buildings in some forest cells such that:

- no two buildings are in the same cell or adjacent to each other;
- each forest cell either contains a building, or is adjacent to another cell with a building.

C. Campus for SIT

Many possible approaches.

Approach 1: construct a graph with vertices being forest cells, and edges connecting adjacent cells. This graph is bipartite, thus we can place buildings in all vertices in any half.

…unless there is an uncovered vertex in another half! This can only happen if the vertex does not have any neighbours.

C. Campus for SIT

Approach 2: consider forest cells in any order. For the current cell, if none of its neighbours contain a building, place a building in it.

Why does this work correctly? Clearly, no two buildings will be adjacent. If there is a cell that does not have a building at the end of the process, then it must have had an adjacent building at the moment we were processing this cell. Thus, all conditions are satisfied.

D

D. Data Consistency

There is a stack of data drives of size s MB each, initially all empty. To save a file of size d MB, we fill empty space on all drives starting from the first, and replace all full drives with empty ones once the file is saved. How many drives in the stack we need to be able to save an indefinite number of files of size d MB?

D. Data Consistency

Consider two cases:

- $d \geqslant s$: each file takes up $\lfloor d/s \rfloor$ full drives (which are immediately emptied), plus $d \bmod s$ MB on the next free drive, hence we can add $\lfloor d/s \rfloor$ to the answer and change $d \to d \bmod s$.
- $d < s$: if $d = 0$, we are done. Otherwise, it will take $\lceil s/d \rceil$ files to fill up the first drive (which will be immediately emptied). This is equivalent to adding 1 to the answer, and changing $d \to d\lceil s/d \rceil - s = (-s) \bmod d$.

This is very similar to Euclid's algorithm, and similar analysis shows that the algorithm converges in $O(\log \max(s, d))$ steps.

E

E. Evenly Divided

There are n people, half tall and half short. Find a matching between the halves that avoids matching a person with their mentor in a hierarchical tree structure (probably with many roots).

E. Evenly Divided

There are several possible solutions.

Solution 1: the answer exists if and only if there is no vertex adjacent to the entire opposite half in the forest. Proof by induction: either there is vertex with a single neighbour and removing it leaves a solvable position, or matching any possible pair leaves a solvable position.

Solution 2: if all connected components of the hierarchy forest contain $< n/4$ vertices of each half, then greedy merging works (take the largest component for each half). To reduce to this situation as fast as possible, get rid of the centroid of the largest component.

Solution 3: there are very few vertices with large degree in the forest. Do many random matchings starting from the ones with fewer possible matches.

F

F. Formula-1

You are given a form of letters and a picture with several letters. Letters consist of pixels that are switched on, and letters don't touch each other. Find the number of occurences of each letter on the picture.

Letters are 4-connected, and can't touch by side. So each 4-connected component is a single letter.

Let's use dfs to get all 4-connected components. Each component can be checked against each rotation of each letter.

Bounding boxes of letters can intersect, so one should check only 'X' cells, that would be fast enough.

G

G. Generator

There is a circular wheel with buckets on it. A bucket gets filled when it's on the top, and gets emptied when it's on the bottom. You are given the weight for a filled and an empty bucket. The wheel is rotating. Find the maximum value of the sum of products of bucket weight and x-coordinate.

When the set of the filled buckets is fixed, the resulting vector is rotating with the wheel. So, the number of events when the set changes is $O(n)$. Let's sort them all and process one by one.

When the set is fixed there are 4 interesting points — the start and end points of a segment, and two points when vector is horizontal. Last two can be found from the direction of vector, and checked if they are inside the segment.

H

H. Hacker's Heist

There are n cafes, each opens at some moment, closes at some moment, and have wifi with some speed. File of a known size should be uploaded. Distance between each two cafes is given. What is the smallest time in which the upload can be finished?

There are only two reasons to move: either the cafe we are sitting in closes, or some other one will open when we come to it.

So we can create $O(n^2)$ edges, which means "we can move from cafe i to cafe j at time t".

H. Hacker's Heist

For each edge we want to find the maximum data that can be uploaded before transfering along it. Let's process edges in order of the increasing timestamp. There are $O(n)$ possible previous edges, we can choose the best one just by checking them all.

At the end we should check all edges as the last one, if it's enough time to upload all the data after transferring along this edge in the last cafe.

The total complexity is $O(n^3)$.

I

I. Interesting Game

There is a tree. Two players play a game. The first one chooses a vertex, then they move to the adjacent vertex, no returns are allowed. The player who can't move loses. How many starting positions the first player can choose so that he wins the game?

If we need to solve the problem for a single root, dynamic programming can be used.

Each subtree can be either winning (if a player starts in its root, then they win) or losing. Subtree is winning if at least one of its children is losing.

The total time is $O(n^2)$ — too slow.

I. Interesting Game

To make the solution faster one need to reuse results.

Each subtree can be determined by the edge with direction. There are $O(n)$ different subtrees for all roots. Doing just this optimizes solution to $O(\sum d_i^2)$, which is better but still not enough.

To get rid of the square, we need to calculate the number of losing children for the vertex once, and then check if there is a losing vertex by subtracting one child, instead of summing all others. Now the complexity is $O(\sum d_i) = O(n)$.

J

J. Jackpot

There are n doors. The player can ask to open d doors, and then can choose one. If they choose correctly, a prize is given. What is the maximum expected prize?

If d doors are opened the expected payout is $\frac{m-(d \cdot f)^2}{n-d}$.

One can show that this function is convex, so the minimum can be found by ternary search. Also, one can find the derivative roots analytically, and check for the closest integer values.

K

K. Kings

n kings stays on chess board of size $n \times n$. They need to be moved to diagonal. King can move one cell vertically or horisontally in one move. How many moves are required?

Consider resulting permutation of kings. Let king i finish at point (a_i, a_i). Total cost is no less than $\sum\limits_{i=0}^{n-1} |a_i - x_i| + |a_i - y_i|$.

This cost is reachable if we first move closest king. If there is other king on path, we can go upto this position, and exchange finish points, moving obstructive king instead, and our king will be moved to it's position.

Minimizing cost is well known assigment problem. It could be solved by Min-Cost-Flow or Hungarian algorithm.