A

A. Answers

There is a functional graph (a directed graph where each vertex has only one outgoing edge). Each vertex has a number, which is either 1 or 2.

For several q, determine if there is a path with sum of numbers in its vertices equal to q.

A. Answers

If all numbers are 2, then no odd sum can be produced (for parity reasons). Any even sum can be produced: to get sum x, start at any vertex and make $x/2$ steps.

If there is at least one 1, any sum can be produced. Start at the vertex labeled with 1 and go forward until sum becomes $\geqslant x$. If it is x, we're done. If it is $x+1$, trim a single 1 from the front.

B

B. Business

There is a weighted tree. Some vertices contain a shop. For each vertex, the closest shop is chosen (compared by vertex number in case of a tie).

Find the location to build a new shop such that it would be the closest one for as many vertices as possible.

B. Business

Let's solve the problem if our shop has the highest priority (that is, it always wins against other shops with the same distance no matter what numbers are).

For each vertex, find the distance to the closest shop and the minimum number of such shop. To do it, run a BFS initializing the queue with all shops.

Let the vertex i have closest shop at distance r_i. If we build a shop anywhere closer than r_i then it will will. We use centroid decomposition to add 1 to all vertices u such that $dist(u, v) \leqslant r_i$.

Now for each vertex we know how many other vertices it will attract if we build a shop there.

B. Business

Now we deal with vertex numbers. Let us build a shop in vertex s. If the closest shop for v has number $c < s$ then our shop should be strictly closer. Thus in previous solution we should add 1 to all vertices with distance to v less than r_v.

We do it simultaneously for all vertices. Sort vertices by number. Initially for each vertex we add 1 to all vertices at a distance at most $r_v - 1$. Then we process vertices from highest numbers to lowest. Before trying to put a shop at vertex s, we take all non-processed vertices v with closest show number greater than s and add 1 to their neighbourhood of radius r_v.

This way we account exactly vertices that would prefer our shop: either their current closest shop is strictly farther than our shop or has the same distance but greater number.

C

C. Curves and Symmetries

You are given a curve in a plane. Find its symmetry line if one exists.

The curve $(x(t), y(t))$ is given as two polynomials $x(t)$ and $y(t)$.

C. Curves and Symmetries

Let d be the highest degree of polynomials. Extend shorter polynomial with zeroes such that both have degree d. If $d \leqslant 1$, solution can be found by hand. Now $d \geqslant 2$.

To get the direction of the symmetry line it suffices to consider only highest coefficients x_d and y_d. If d is even, the direction is (x_d, y_d), otherwise $(-y_d, x_d)$. It is easy to verify that it is the only option when t tends to infinity.

There are two problems left: find the shift of the line and validate that it is indeed the symmetry line. We do them at once.

C. Curves and Symmetries

If we reflect the curve against some line parallel to a symmetry line, we get the original curve but translated (that is, $x_{ref}(t) = x(t) + C_x$ and $y_{ref}(t) = y(t) + D$. To check if this is indeed the case, we explicitly calculate $x_{ref}(t)$ and $y_{ref}(t)$, reflecting the curve against the line with found direction and passing through origin.

After reflection functions would look like a sum of terms of kind $(x_i + y_j)^k$. Use binom expansion to expand the braces.

Free coefficient of the symmetry line is found from the difference between original and reflected curves.

D

D. Distance to Ellipsoid

There is an ellipsoid given by equation

$$ax^2 + by^2 + cz^2 + dyz + exz + fxy = 1$$

Find the minimum distance between the origin and some point on the boundary of the ellipsoid.

D. Distance to Ellipsoid

Note that the center of the given ellipsoid is at origin. So the distance to the ellipsoid equals the length of the shortest principal axis.

If we could translate its equation into canonical form $x^2/A^2 + y^2/B^2 + z^2/C^2 = 1$, the problem would be easy: lenghts of principal axis here are A^2, B^2 and C^2. To do the translation, we use the magic of eigenvectors.

D. Distance to Ellipsoid

Consider the quadratic form of the ellipsoid in general form:

$$\begin{pmatrix} a & f/2 & e/2 \\ f/2 & b & d/2 \\ e/2 & d/2 & c \end{pmatrix}$$

It is known that it can be transformed to a diagonal form with orthogonal transform which preserves eugenvalues. Diagonal matrix correspond to the canonical form, and eigenvalues there are exactly $1/A^2$, $1/B^2$, and $1/C^2$. So we have to find the minimum eigenvalue of our form.

D. Distance to Ellipsoid

The characterictic polynomial of the matrix in question is qubic. We will explicitly find its roots.

We can use the following trick to find its roots. Take the derivative, which is a quadratic equation. Solve it analytically. Roots of the derivative are the extrema of the original polynomial and divide the real line into three parts. Binary search in each part gives us all the roots of the original polynomial.

E

E. Elections

The following elections procedure happens. s candidates are to be selected, m parties propose several own candidates. Each party gets a certain number of votes. Canditates of the party with x votes get value $x/1$, $x/2$, $x/3$, ... After that s candidates with the largest value are picked. Ties are broken arbitrarily.

You know the outcome of the elections: the total number of votes cast and the number of picked candidates for each party. Determine the minimum and maximum possible number of votes each party could get.

E. Elections

Editorial will be posted soon.

F

F. Fruit Game

There are n fruits in a line, i-th fruit costing c_i. Two players play in turns, buying and removing some nonempty prefix of fruits. Given initial amounts of players' money, who wins if both play optimally?

F. Fruit Game

We run dynamic programming from left to right. Consider some suffix. We know the total balance of players and want to know, what is the minimum amount of money first player should have in order to win starting from this suffix. We denote this value with l_i.

Let S be the sum of all costs and s_i be the sum of all costs from i to n. Assume we know l_j for some j. How much money do we need at position i to be able to move to j and put the second player in losing position?

- Second player should have at most $l_j - 1$ money.
- We both in total have $A + B - (s_0 - s_j)$ money.
- We should spend $s_i - s_j$ money to get from i to j.
- In total, we need $s_i + (A + B + s_0 - l_j + 1)$ money at i.

F. Fruit Game

- In total, we need $s_i + (A + B + s_0 - l_j + 1)$ money at i.

The lesser is the value in brackets, the better for the first player. We store the suffix minimum of the value in brackets and calculate $l_i = s_i + suffix\_minimum$.

G

G. Guards on the Wall

You are given a simple polygon with sides parallel to coordinate axes. Place a minimum possible number of guards to some vertices of the polygon such that each vertice could be observed by at least one guard.

G. Guards on the Wall

For each pair of vertices check if one can be observed from another. Now we have an undirected graph and have to find a minimum dominating set.

Checking all subsets works in $\Theta(2^n)$ and is too slow.

Any backtracking with reasonable pruning works.

Fix the desired answer and run backtracking. At each step we try two options: either forfeit the vertex or add it to the dominating set if it dominates some new vertex.

G. Guards on the Wall

Fibonacci-like reasoning can be used (it is not directly applicable in this editorial but is still useful). When the vertex is taken, at least two vertices should likely be covered: itself and some its neighbor. Thus for a subproblem of size n we get subproblems of size $n-1$ (not to take the vertex) and at most $n-2$ (to take the vertex and automatically satisfy its neighbor), so $T(n) = T(n-1) + T(n-2)$, yielding Fibonacci series.

For this certain problem there exists an $O(1.5...^n)$ algorithm. Consider Wikipedia page "Dominating set" for further links.

H

H. Happiness and Cities

m people travel through n cities in order. Every day person i goes to the next city with probability p_i and stops the journey with probability $1 - p_i$.

Visitor i gets h_{ij} happiness when visiting city j. More, if c_j people visit city j and c_{j-1} visit city $j-1$, then each visitor of the city j gets extra $\frac{c_j}{c_{j-1}} h_{ij}$ happiness.

Calculate the total expected happiness of all visitors at the end of the tour.

H. Happiness and Cities

We do a submask DP backwards. Starting with the last city, we store $d[i][mask]$ — the expected total happiness if people from mask $mask$ start at city i.

The hard part of the problem is the one with $\frac{c_j}{c_{j-1}} h_{ij}$. To calculate the DP for city i, we iterate through the mask of people which will make it to the city $i+1$. For each mask, we know its probability, the number of people in it, and, recursively, the total expected happiness if they start from the city $i+1$.

After that we do a subset convolution and calculate sum of submasks for each mask, taking personal probability into account. Then we iterate through masks of people who made it to the current city and use the convoluted value to update the answer. The size of the current mask is accounted directly at this step, the size of the next mask is accumulated in the convolution.

I

I. Ice Age

There is a grid with at most 32 empty cells. At most 5 cells are occupied by animals. They move through the grid according to specific rules (see original problem statement for details). You should find the optimal way to bring the squirrel to the nut safe and sound (again, see original problem statement).

I. Ice Age

There are at most 32 cells and 5 animals, of which either two pairs or three animals are the same. So in the worst case there are at most $32 \cdot 31 \cdot 30 \cdot 29 \cdot 28/4 \approx 6 \cdot 10^6$ configurations. There are extra configurations where some animals are dead, but there are less than a million of them. Many configurations are likely unreachable due to the nature of the problem.

Store configuration in a bitmask (5 bits for the position of each animal). Run a BFS from initial configuration. To do a transition, try moving each alive animal to each possible direction checking all stopping conditions naively.

In jury tests there were at most 10^5 states per each layer of BFS.

J

J. Jokémon

There are m creatures, each assigned with one or two of n types. Each type has a damage multiplier, one of 0, 0.5, 1, 2. When we hit a single-type creature, it gets damage equal to the multiplier of its type. When we hit a double-type creature, it gets damage equal to the product of multipliers of its types.

For each creature, we know if the damage it gets is 0, between 0 and 1, exactly 1, or greater than 1. Find the set of multipliers for all types which is consistent with the known damage.

J. Jokémon

- We denote type multipliers with m_i.
- Determine all multipliers that must be nonzero. Set all others to zero.
- For other types we use 2-SAT.
- Assign two variables to each multiplier, m_{i+} and m_{i-}, meaning that it is greater than 1 or less than 1, respectively. For each i, add a clause that m_{i+} and m_{i-} are not both true.

J. Jokémon

- If $m_i \cdot m_j = 0$, do nothing.
- If $0 < m_i \cdot m_j < 1$, request that one of m_{i-} and m_{j-} is true and m_{i+} and m_{j+} are false.
- If $m_i \cdot m_j = 1$, request that m_{i+} and m_{j-} have the same value. Same for m_{i-} and m_{j+}.
- If $1 < m_i \cdot m_j$, request that one of m_{i+} and m_{j+} is true and m_{i-} and m_{j-} are false.