Moscow IPT Contest

Problem analysis

Artem Vasilev   Pavel Krotkov

Peking University Camp, April 2016

A. Automaton

Problem statement

- You are given an non-deterministic automaton for single-character alphabet.
- Two states are *partially equivalent* if they can be both be reached with the same string.
- After that, apply transitive closure to relation above.
- Can given two strings can end up in partially equivalent states?

A. Automaton

Preprocessing

- First, let's remove all unreachable vertices.
- Transitive closure means we can merge two partially equivalent states together until there are none left.
- When there are at least two edges from a vertex, these two states are equivalent.
- So after merging, every vertex has at most one outcoming edge.

A. Automaton

Preprocessing

- Maintain a set of all vertices with outdegree ≥ 2
- Get the vertex from this set and merge all the vertices using outcoming edges.
- Need to merge the sets of incoming and outcoming edges.
- Merging "from small to large" results in $O(m \log^2 m)$ or $O(m \log m)$ time.

- Graph, where each vertex has at most one outcoming edge looks like a chain, maybe with an edge from the last vertex.
- After extracting the length of preperiod and the length of period, it's easy to check whether two numbers are equivalent.

B. Bijections

Problem statement

- Calculate the number of bijections (permutations) of size n such that numbers $2, \ldots, k$ are reachable from 1.
- n and k are huge.
- Output the first non-zero coefficient modulo p.

B. Bijections

Number of permutations

- The answer is $\frac{n!}{k}$.
- Proof by induction on n: if $n = k$, then all the elements lie on the same cycle, number of such cycles is $(k-1)! = \frac{k!}{k}$
- When we add number $n+1$ in the permutation, it can be added as a fixed point (1) or inserted after any element in each cycle (n possibilities). So the number of these permutations if $\frac{n!}{k}(n+1) = \frac{(n+1)!}{k}$.

B. Bijections

How to output

- Represent k as $k'p^t$, where k' isn't divisible by p. In the end, multiply answer by $k'^{-1} \mod p$
- How to calculate $n! \mod p$?
- $n! = 1 \cdot 2 \cdot \ldots \cdot (p-1) \cdot p \cdot (p+1) \cdot \ldots \cdot 2p \cdot \ldots \cdot n$
- Every p^{th} number is divisible by p.
- $n! = \ldots \cdot 1p \cdot \ldots \cdot 2p \cdot \ldots \cdot \left\lfloor \frac{n}{p} \right\rfloor p \cdot \ldots$
- $n! \mod p = (-1)^{\left\lfloor \frac{n}{p} \right\rfloor} \left\lfloor \frac{n}{p} \right\rfloor ! \cdot 1 \cdot 2 \cdot \ldots \cdot (n \mod p)$
- $\left\lfloor \frac{n}{p} \right\rfloor !$ can be calculated recursively, and all partial factorials $a! \mod p$ can be precalculated.
- Division can be performed in $O(\log n)$, so the whole solution becomes $O(\log^2 n)$

C. Egyptian Tale

Problem statement

- Calculate the "pyramidal" Fibonacci number F_N
- $N = a_1^{a_2^{\cdot^{\cdot^{\cdot^{a_n}}}}}$

C. Egyptian Tale

Solution

- Solution can be divided in 3 large "steps":
  1. Determining the period of Fibonacci sequence modulo m $\pi(m)$
  2. Calculating $N = a_1^{a_2^{\cdot^{\cdot^{\cdot^{a_n}}}}}$ modulo $\pi(m)$
  3. Calculaing Fibonacci number F_N

C. Egyptian Tale

Computing the period

- $\pi(m)$ is called a *Pisano period*.
- $\pi(m) \leq 6m$
- Let $m = p_1^{k_1} p_2^{k_2} \dots p_n^{k_n}$. Then, by CRT, $\pi(m) = LCM(\pi(p_1^{k_1}), \pi(p_2^{k_2}), \dots, \pi(p_n^{k_n}))$
- We are OK with finding any multiple of $\pi(m)$.
- Some facts about $\pi(m)$:
  1. $\pi(p^k) \mid p^{k-1}\pi(p)$
  2. $$\pi(p) = \begin{cases} 3 & \text{p} = 2 \\ 20 & \text{p} = 5 \\ \text{p - 1} & \text{5 is a quadratic residue modulo} p \\ 2(\text{p} + 1) & \text{5 is not a quadratic residue modulo} p \end{cases}$$
- Allows us to calculate $\pi(m)$

- Let's calculate $a_1^X \mod k$ where $X = a_2^{\cdot^{\cdot^{\cdot^{a_n}}}}$
- Suppose we know $X \mod \phi(k)$ or, if X is small enough, the value of X itself.
- If X is small enough, just calculate $a_1^X \mod k$ and return.
- Otherwise, you can calculate $a_1^{X+\phi(k)} \mod k$.

C. Egyptian Tale

Calculating F_N

- In steps 1 and 2 we learned Pisano period $\pi(m)$ and $N \mod \pi(m)$
- Just calculate $N \mod \pi(m)$ Fibonacci number in $\log(m)$

D. Cut the String

Problem statement

- String S
- Find k non-overlapping substring of maximum length
- $|S| \leq 10^5$, large TL

D. Cut the String

General idea

- Binary search for an answer
- Answer check
  - Calculate hash function for all substrings of length t
  - For every value of hash find maximum set of occurences separated by at least t positions
  - Can be done with two pointers
- $O(n \times log_2^2 n)$, large constant, but large TL

E. Graph Game

Problem statement

- n vertices
- A move — adding one new edge
- The player who connect all vertices loses

E. Graph Game

Perfect solution

- Answer depends on the $n \mod 4$
- Proof for every remainder is different
- Figure out all proofs

Easy solution

- Bruteforce
- $2^{\frac{n\times(n-1)}{2}}$ states
- Works for maybe 8 vertices
- Spot the answer

E. Graph Game

Another solution

- Bruteforce
- State contains of
  - Partition of n
  - Current amount of edges we can add without connecting two components (X)
- Transitions:
  - Add the edge inside one of components ($X = X - 1$)
  - Merge the components of size a and b ($X = X + a \times b - 1$)
- Will work for larger n, easier to spot the answer

F. Politicians

Problem statement

- n Thores, m Whigs
- Round table
- Calculate arrangements without $a+1$ Thories and $b+1$ Whigs sitting consequently
- $n, m, a, b \leq 1000$

F. Politicians

Simplification

- Fix the color of person on the 1^{st} position
- Fix the size of subsequent persons around 1^{st} position k
- Solve the problem for the remaining seats
- Multiply the answer on k (all possible shifts)

F. Politicians

Solution idea

- Dynamic programming
- $f_{i,j,last}$ — we used i Thores, j Whigs and the last person was *last*
- Transition:

```
for t = j + 1 ..  j + a
  f[i][t][!last] += f[i][j][last];
```

- $O(n^2)$ states, linear transition, total complexity $O(n^3)$

- Transition:

```
for t = j + 1 ..  j + a
 f[i][t][!last] += f[i][j][last];
```

- We can make this transition in $O(1)$ if we use prefix sums
- Total complexity $O(n^2)$

G. Polyhedron

Problem statement

- Set of n points in 3D space
- Find total length of all edges of convex hull
- $n \leq 300$

G. Polyhedron

General idea

- For every three points on the same line remove the middle one
- Check every pair of points if it's an edge of a convex hull
- $O(n^2)$ checks, every check should take linear time, total complexity — $O(n^3)$

G. Polyhedron

Check of one segment

- Build a plane orthogonal to this segment
- For every point get a projection of it on this plane
- If the point corresponging to our segment is a vertex of convex hull of this projections — the segment is an edge of a 3D convex hull

G. Polyhedron

Check whether a point is a vertex of CH

- We need to do it in linear time
- Take the point with maximum polar angle
- Build a line
- Check whether all points are at the same half-plane

H. Traffic Jams

Problem statement

- Graph
- Length of edges is changed through time
- Find paths from a vertex to other vertices

H. Traffic Jams

Problem solution

- Read the statement carefully
- Dijkstra's algorithm
- Why does it work here?
  - We never need to wait in some vertex
  - We never want to arrive to some vertex later

I. Robots

Problem statement

- We have a grid $n \times m$ $(n, m \leq 50)$
- Some cells are closed
- We need to cover it with cyclic routes of length ≥ 4

I. Robots

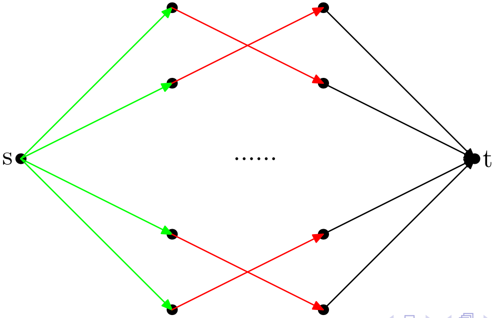
Useful observations

- Let's color our grid like a chess board
- In any cyclic route any cell has two neighbours of opposite color

I. Robots

Problem solution

- Let's build a network
  - First layer — black cells
  - Second layer — white cells
  - Capacity of green and black edges — 2
  - Capacity of red edges — 1, red edges connect neighboured cells
- Maximum flow in this network corresponds to some set of cycles
- If network is full — the answer is `YES`



J. Trees

Problem statement

- Two trees drawn on plane
- Find two largest isomorphic subtrees
  - Considering the order of children in all vertices
- $n \leq 200$
- Maximum degree of vertice doesn't exceed 50

J. Trees

Solution idea

- Split every edge on two going in opposite directions
- For every pair of edges (one in first graph, one in second graph) calculate the answer
    - Only for subtrees to which the edge directs
- Dynamic programming
    - Use the answers for all possible pairs of child vertices
    - Much like LCS when calculating actual result
- Store all results, do lazy DP
- Total complexity $O(n^2 \times maxDegree^2)$

J. Trees

Calculating the answer

- Check every pair of undirected edges
- Check two possible directions of matching
- Add up two answers for two pairs of directed edges

- Circle and triangle
- Calculate intersection area

K. Triangle and Circle

Possible solutions

- Precise solution
  - Check all possible cases of intersection
- Easy solution
  - Approximate the circle with convex polygon
  - Intersect this convex polygon with half-planes 3 times
  - Find an answer