Alex Fetisov contest 4

Problem analysis

Artem Vasilev Pavel Krotkov

Brazilian ICPC Summer School, 2016

A. Arrays

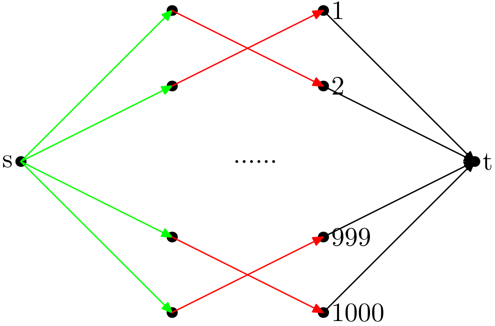
Problem statement

- We have n arrays with numbers from 1 to 1000
- We need to choose at most k numbers in each array
- Sum of all numbers should be maximized
- All numbers should be different

A. Arrays

Problem solution

- Let's build a network with n vertices in first level and 1000 vertices in second level
- Green edges: capacity $= k$, price $= 0$
- Red edges: capacity $= 1$, price $= 0$
- Black edges: capacity $= 1$, price $= 1000 - x$
- The maximum flow of minimal cost in such network is the answer



B. Goondex

Problem statement

- We have a set of strings
- Each string has it's rank
- Update orepations
  - Add a new string with rank 1
  - Increase rank of some string by 1
- Query operation
  - Get the string with prepix p and the highest rank

B. Goondex

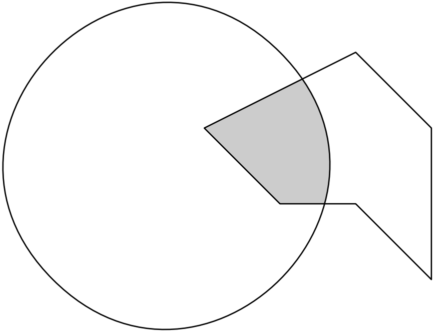
Problem solution

- Trie with all the strings
- Each vertex contains link to a vertex with maximum rank in it's subtree
- Data structure analysis:
  - Query operation: $O(1)$
  - Update operation for string S: $O(|S|)$

C. Intersection

Problem statement

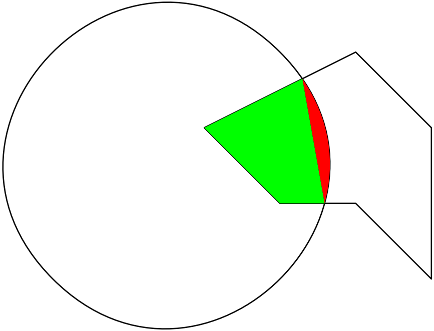
- We have a convex polygon and a circle
- We need to find area of their intersection



C. Intersection

Precise solution

- Intersection always contains of 2 parts
- One of the parts is convex polygon, another one – circular segment.
- We just need two intersection point to figure all of it out.



D. LCA

Problem statement

- We have a tree and a root in it
- We need to find all vertices with maximum value of function f
- f is defined as *Amount of pair of distinct vertices for which this vertex is LCA*

- For every vertex we'll store size of it's subtree $S(v)$
- A set of children for particular vertex is called $C(v)$
- The f value for the vertex is calculated as $\frac{\sum_{s,t \in C(v), s \neq t} S(s) \times S(t)}{2}$

E. Tree picture

Problem statement

- We have tree drawing (field filled with some characters)
- 0 means vertex
- - and + mean edge
- We need to get a tree from it's drawing

E. Tree picture

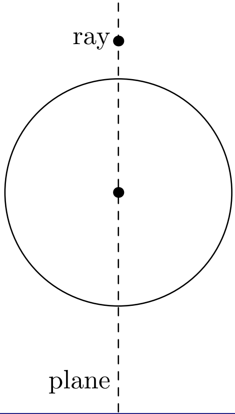
Problem solution

- Assign numbers to all occurences of 0
- For every occurence of 0
  - Find all not processed edges going out of it
  - Walk along them until we find another vertex
  - Add the edge between two found vertices

- We have a sphere, a point and a ray out of this point
- We need to find a plane which contains ray and touches the sphere

F. Planet
Problem solution (1)

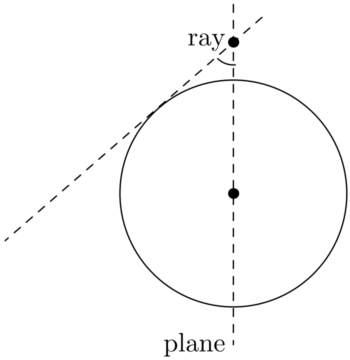
- Let's find a plane that contains a ray and a sphere center
- Let's fine a plane, which contains a sphere center and is perpendicular to the ray
- Les't project everything on the second plane



F. Planet

Problem solution (2)

- Let's find the angle between plane projection and tangent line
- Let's rotate the plane on this angle around the ray



G. Poker

Problem statement

- We have poker game played by Texas Holdem rules
- We know hands of both players, flop, turn and river
- We need to calculate probabilities of winning for each player during each round

- Thorough implementation all rulles of combinations
- Brute-force all possible outcomes during each round
- Optimizations are required to make it fit into TL

G. Poker

Possible optimizations

- When checking for straight, ignore suits
- Use precomputed results, when possible (see above)
- Use numerical representation for possible combinations for easier comparisons
- Use bitmask optimizations, when possible

H. Toys

Problem statement

- You are given a line of colored cubes and are asked to answer some queries:
  1. Take the subsegment of cubes and rotate it 90 degrees in one of 6 directions
  2. Calculate the number of each color on the upper side of cubes

H. Toys

Solution

- We'll store all the cubes in a treap, allowing us to use range operation as well as reversal of a segment.
- For each node in treap, maintain the number of each color on each side of all the cubes covered by this subtree.
- Every query of the first type can be expressed as application of a permutation to all of the array in the segment, and, possibly, reverse it.

H. Toys

Solution

- Application of a permutation to the subsegment can be done using common lazy propagation technique, reversal of a segment is done using an additional flag.
- The same array can be used to output the final state of all cubes.

I. Union

Problem statement

- You are given a tree and some queries.
- On the path from u to v, find the number of edges that have weight $\leq k$.

I. Union

Solution

- Split each query into three queries from root to some vertex: $ans(u, v) = ans(root, u) + ans(root, v) - 2 \cdot ans(root, LCA(u, v))$. Collect all the queries for one vertex in a separate list.
- Process all the queries with one DFS: after we've come to the vertex v, answer all the queries associated to it.
- After that, choose an edge, add its weight to the data structure, and recurse into child. After returning from child, remove that edge.

I. Union

Solution

- We need the data structure to support following queries:
  - Insert a number x
  - Remove a number y
  - Find how many numbers are smaller than z
- All of this can be implemented using segment tree, treap or Fenwick tree in $O(\log N)$ time.

- You are fighting an enemy, each of you having some hit points.
- You have T weapons and P places to shoot. Each pairs deals some damage with some probability.
- What's the probability that you win?

J. WH.A.T.S.

Solution

- All the input parameters are small: use dynamic programming.
- $dp_{HP_{player}, HP_{enemy}}$ is the probability of first player winning with HP_{player} hit points and enemy having HP_{enemy} hit points.
- For each weapon and body part precalculate the probability for that weapon to hit exactly k times.
- Calculate transitions: iterate over all weapons and body parts, choose the pair which maximizes the probability to win.
- Time complexity: $O(HP_{player} \cdot HP_{enemy} \cdot T \cdot P \cdot VP_{max})$