ACM ICPC 2013–2014 Northeastern European Regional Contest Problems Review

Roman Elizarov

December 1, 2013

Problem A. ASCII Puzzle

- The problem is solved by exhaustive search
  - fill each spot in the trivial puzzle from the top-left to the bottom-right corner
  - try to place each piece that fits
  - backtrack after trying all pieces for a place
- Must check which pieces can be placed on borders
  - and place them only onto the corresponding borders
  - otherwise time-limit will be exceeded

Problem B. Bonus Cards

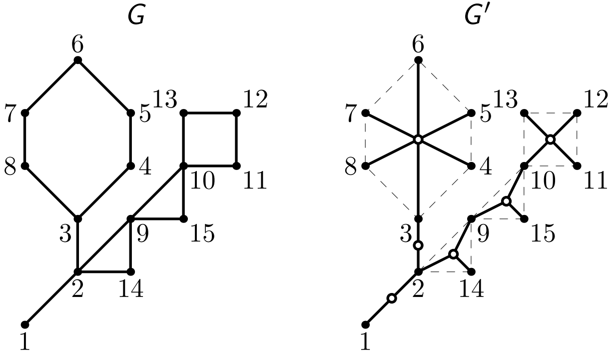
- The problem is solved by dynamic programming
- Let k be the total number of tickets already distributed, $0 \le k \le n$
- Let g be the number of ICPC card holders who already got tickets, $\max(0, k-b) \le g \le \min(a, k)$
- Let $P_{s,k,g}$ be the probability of Dmitry getting a ticket with a card that has s slots in each draw round
  - $s = 2$ for ICPC card, and $s = 1$ for ACM card
- Use the following equation to compute the desired probability $P_{s,0,0}$ for each s:

$$P_{s,k,g} = \frac{s + 2(a-g)P_{s,k+1,g+1} + (b-k+g)P_{s,k+1,g}}{s + 2(a-g) + (b-k+g)}$$

- Here $s + 2(a-g) + (b-k+g)$ is the total number of slots in this draw round for Dmitry's card, for $a-g$ remaining ICPC cards, and for $b-k+g$ remaining ACM cards

Problem C. Cactus Automorphisms

- Use depth-first-search to find all cycles in the given graph G
- Build graph G' with original vertices, and where each cycle in G is a new vertex, and each edge which is a part of a cycle is a new vertex (new vertices are in white)

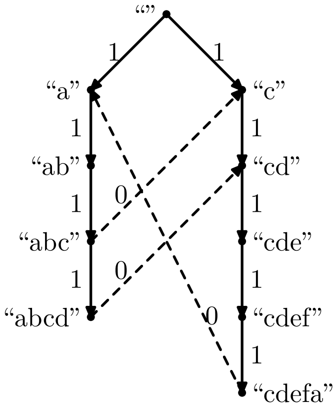


Problem C. Cactus Automorphisms (2)

- Graph G' is a tree
- G' has an even diameter and has the unique center
- The center of G' is either a vertex, a cycle or an edge in G
- Hang the graph G' using its center as a root and count a number of automorphisms on a tree in bottom-up fashion
  - k identical children of a vertex can be rearranged for $k!$ combinations
  - children of a cycle in G can be rearranged for 2 combinations if the sequence of children on this cycle can be reversed
- The root of tree G' needs a special attention when it corresponds to a cycle in G
  - it may have rotational symmetries and/or a mirror symmetry
  - it may have a lot of children, so an efficient algorithm (like Knuth-Morris-Pratt) must be used to find those symmetries

Problem D. Dictionary

- Let P be a set of prefixes for a given set of words
- Build a weighted directed graph with nodes P
  - add an edge of weight 1 from a prefix p to all prefixes pc (for all characters c)
  - add an edge of weight 0 from a prefix p to a prefix q when q is a suffix of p
- 1-edges of this graph constitute a trie for a given set of words
  - but it is not an optimal solution
- Minimum spanning tree in this weighted directed graph corresponds to the problem answer
  - use Chu–Liu/Edmonds algorithm



An example for words “abcd” and “cdefa”

Problem E. Easy Geometry

- Let $(x, y_t(x))$ be the top point of the polygon at a given coordinate x and $(x, y_b(x))$ be the bottom point of the polygon
  - these functions can be computed by a binary search
- Let $s_w(x)$ be the max generalized square of a rectangle of the fixed width w with the left edge at x

$$s_w(x) = w \times (\min\{y_t(x), y_t(x+w)\} - \max\{y_b(x), y_b(x+w)\})$$

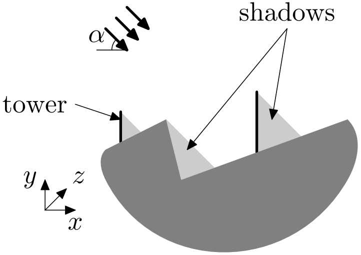
- Let $s(w) = \max_x s_w(x)$ be the max square of a rectangle of the fixed width w
  - $s_w(x)$ is convex, so $s(w)$ can be found by a ternary search
- Let $s = \max_w s(w)$ be the max square of a rectangle — the answer to the problem
  - $s(w)$ is convex, so s can be found by a ternary search

Problem F. Fraud Busters

- This is the simplest problem in the contest
- It is solved by going over a list of codes and checking each one against a code that was recognized by the scanner

Problem G. Green Energy

- Compute coordinate z for each point — coordinate of the projection onto a line perpendicular to the sun
- Place the largest tower at a point with the max z coordinate
- Place other towers in any order on points with decreasing z coordinates so that they do not obscure each other
- If min z coordinate is reached and some towers are left, then place them anywhere



Problem H. Hack Protection

- Compute cumulative *xor* values $x_i = \otimes_{j=1}^{j<i} a_j$ ($\otimes$ for *xor*)
  - this way, *xor* for any subarray $[i, j)$ is equal to $x_i \otimes x_j$
- Create a map M which keeps for each value of x_i the list of indices i with this value of x_i
- Compute $b_{i,j}$ — the first index at or after i where j-th bit of a_i becomes zero
- Loop for all i_0 from 1 to n
  - using $b_{i,j}$ one can quickly find consecutive ranges $[i_k, i_{k+1})$ of indices where *and* of subarrays $[i_0, t)$ $(i_k \le t \le i_{k+1})$ has the same value b
  - note, that there are at most 32 such ranges for each i_0
  - use a map M to find a list of all indices with value of $x_{i_0} \otimes b$
  - use a binary search on this list (twice) to find how many indices from this list are in the range $[i_k, i_{k+1})$
  - that is the number of matching values for all subarrays $[i_0, t)$

Problem I. Interactive Interception

- The state space of a point can be kept in array of min and max possible position for each speed
  - There are at most 10^5 possible speeds, so this array can be scanned in a loop on each turn
- Find R that splits a state space roughly in half using binary search
- Use "`check` 0 R" query
- Update the state space after reading the answer
- Repeat until the point's position can be unambiguously determined

Problem J. Join the Conversation

- The problem is solved by dynamic programming
- For each author maintain a map M from an author to a pair of an index and a length of the maximal conversation with the last message from this author
- Process messages in order, find all mentions in a message, and update map M for the author of this message
  - if you find mentions by looking at '`@`' then do not forget to check for a space before it
  - the easiest way to find mentions is to split the message by spaces

Problem K. Kabaleo Lite

- $n = 1$ is a special case
  - the answer depends on the chip of the last player
- For $n > 1$ analyze the best strategy for other players:
  - they place all chips onto the chips of your hidden color h
  - they will obscure as many as possible of your chips on the board, and will place as many as possible of other colors onto the board
- Compute the maximal possible number of chips of each color on the board according to the above
- Check each possible move of yours to find the answer
  - you win only if the number of your color h on the board exceeds any other number
  - you need to maintain the number of only two best other color to figure if the above is true