ACM ICPC 2014–2015 Northeastern European Regional Contest Problems Review

Roman Elizarov

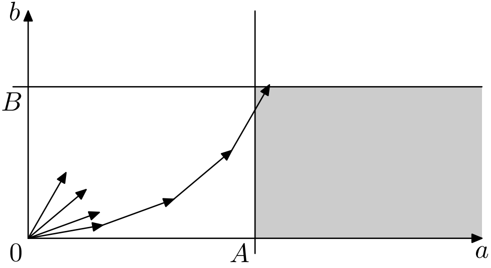
December 7, 2014

Problem A. Alter Board

- The minimal answer to this problem is $\lfloor n/2 \rfloor + \lfloor m/2 \rfloor$
- The solution is to make inversions on each even row and each even column
- To prove that the answer is minimal consider the first column with its n cells that form $n-1$ neighbouring pairs
  - to turn all cells of the first column in the same color inversions must span the first column
  - each spanning inversion makes at most two neighbouring pairs of the same color
  - so the minimum of $\lceil (n-1)/2 \rceil = \lfloor n/2 \rfloor$ inversions are needed
- Then consider the top row in the same way

Problem B. Burrito King

- Consider the problem as a sum of vectors in (a, b) coordinates
- The resulting vector may not go above $b = B$ line and must extend on a axis as far as possible
- It is optimal to greedily add (a_i, b_i) ingredient vectors starting from the ones that have the least angle to $0a$ line (or maximal a_i/b_i), until $b = B$ line is crossed
- Be careful about corner cases with $a_i = 0$ and/or $b_i = 0$

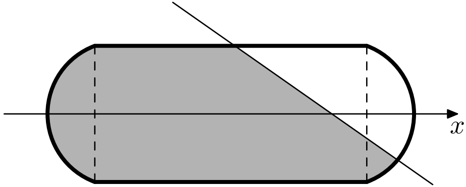


Problem C. Cactus Generator

- This is a straightforward problem for parsing and OO design
  - Define class for *graph* with a method to generate graph given index of the first and the vertices
  - Define class for various *range* types
  - Parse and construct classes tree
  - Build the resulting graph
- Connect arbitrary pairs of vertices of odd degree in the resulting graph using temporary edges
- Use classical algorithm for Eulerian path
- Remove temporary edges to get the minimal number of covering paths

Problem D. Damage Assessment

- Numerically integrate the square section by *dx*
- The square of the cut at a given *x* coordinate is a simple planar geometry problem
- Take care about leftmost point with infinite derivative
  - however, the required precision does not make this a big problem
  - the square section at this point is small



Problem E. Epic Win!

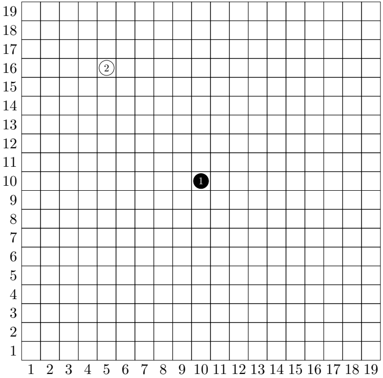
- There is a simple solution with up to n^2 states
- Build your FSM as n copies of a *winning* FSM with n states
  - Each state of a winning FSM corresponds to a state in the opponent FSM
  - Each move of a winning FSM is a winning move for the corresponding opponent's move
  - Next state in a winning FSM corresponds to the opponent move and opponent's next state
  - Leave other transitions undefined
- The first copy of a winning FMS starts in its first state and wins an opponent that stats in it first state by construction
- Model the behaviour of the opponent and your FSM for all opponent start states from the states 2 to n
  - When a yet undefined transition is reached, then insert a transition to a fresh copy of a winning FSM into the state corresponding to the opponent's, thus ensuring win in this copy
  - Stop modelling when loop is detected
  - Loop is inside one copy of a winning FSM and is always winning by construction

Problem F. Filter

- Nothing fancy here
- Just implement what the problem statement asks for in a straightforward way
- The hardest part seems to be reading and understanding the problem statement

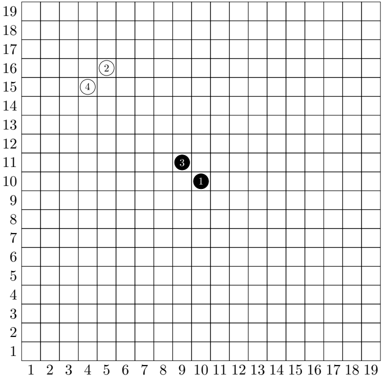
Problem G. Gomoku

- The first player's strategy has pretty strict priorities in the moves it makes and it can be exploited
- Make the first move into the free space of the board



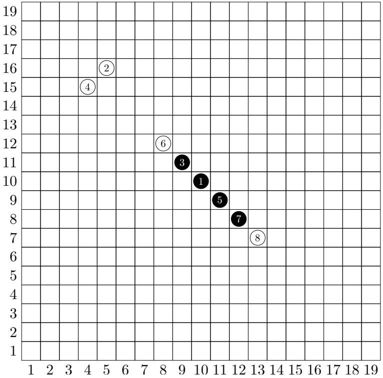
Problem G. Gomoku cont'd

- The opponent must play around the center and you form a diagonal



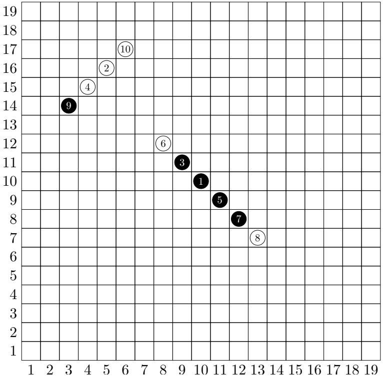
Problem G. Gomoku cont'd

- The opponent forms three in a row and you make defensive moves



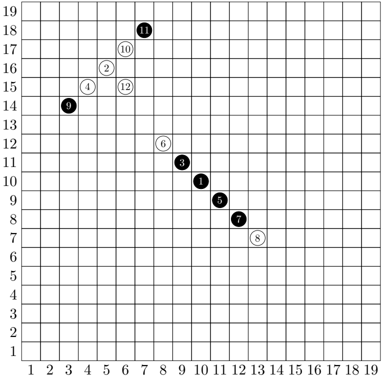
Problem G. Gomoku cont'd

- The opponent closes two in a row at one side, and you extend in on the other



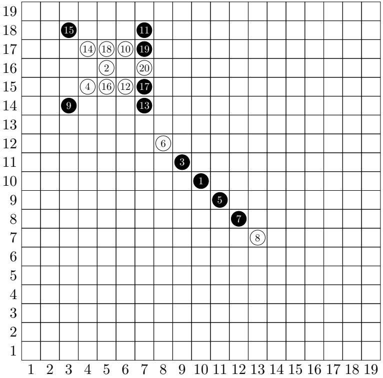
Problem G. Gomoku cont'd

- The opponent closes the three on the other side, but you continue offence at building a winning position



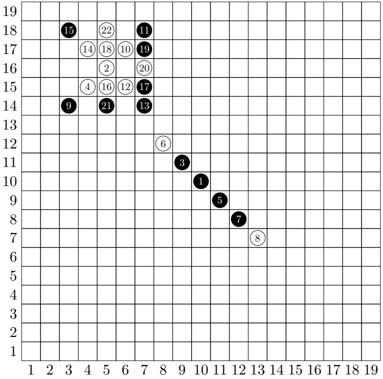
Problem G. Gomoku cont'd

- Force the opponent into a sequence of defensive moves
- Then close four in a row with a hole that is formed by the opponent defence



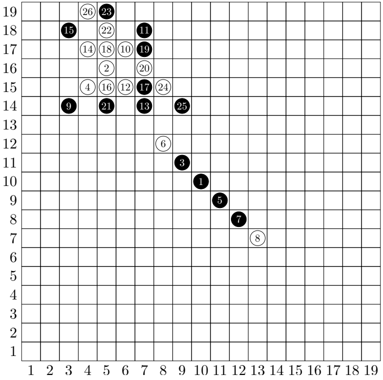
Problem G. Gomoku cont'd

- The opponent closes your open three, you extend it, forming a winning fork



Problem G. Gomoku win

- You win
- It is very hard to win otherwise, because playing first in gomoku gives an enormous advantage even to such a simple strategy



Problem H. Hidden Maze

- Make a rooted tree
- Lets compute how many times each edge is a median
  - Start with an edge with lowest c_i and work in increasing order of c_i
  - For each edge c_i look at its lowest vertex j in the tree
  - For each path from j down into the subtree, let the *balance* be the number of edges with c higher than current c_i minus the number of edges with c lower than current c_i
- For each vertex j maintain an array b_j
  - with $2d_j + 1$ elements $b_j[\delta]$ for $|\delta| \leq d_i$, where d_j is a depth of subtree rooted at j
  - each item $b_j[\delta]$ contains a number of paths down from j with a balance δ
  - including an empty path with balance zero

Problem H. Hidden Maze cont'd

- Initial $b_j[\delta]$ is the number of paths of a length δ down from vertex j
  - It is easy to compute recursively in $O(\sum d_j)$ while building rooted tree
- From the current vertex j walk up the tree
  - For all vertices k up tree from j compute the number of paths with balance zero going from down up to j, then up to k then down to other subtree of k
  - paths with zero balance are the ones where c_i is the median

$$\sum_{\delta=-d_j \dots d_j} b_j[\delta] \cdot (b_k[-\delta - \Gamma_{k,j\uparrow}] - b_{k\downarrow}[\delta - \Gamma_{k,j} - \Gamma_{k,k\downarrow}])$$

  - where $k\downarrow$ is the next vertex from k down on the path to j and $j\uparrow$ is the next vertex up from j
  - and $\Gamma_{k,j}$ is the sum of balances on a path from k to j
- The total complexity is $O(\sum d_j \cdot h_j)$, where h_j is the *height* of vertex j — length of path from root

Problem H. Hidden Maze cont'd

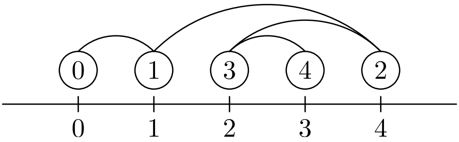
- Update $b_j[\delta]$ when done with an edge c_i
  - For all vertices k up tree from j update the b_k arrow taking into account that c_i balance changes from -1 to 1

$$b_k[\delta] \leftarrow b_k[\delta] + b_j[\delta - \Gamma_{k,j\uparrow} + 1] - b_j[\delta - \Gamma_{k,j\uparrow} - 1]$$

- The total complexity is also $O(\sum d_j \cdot h_j)$
- However, for the graph randomly generated as described in the problem statement $\sum(d_j \cdot h_j) = O(n\sqrt{n})$

Problem I. Improvements

- Consider transposition a_j — the number of ship at coordinate j, that is reverse to what is given in the input
- It is easy to prove that the chain of ships that remain on their initial position corresponds to a subsequence of a_j with a special property:
  - it is an increasing sequence of numbers a_j followed by decreasing sequence of numbers a_j
- Increasing/decreasing subsequence is a well-known problem with $O(n \log n)$ solution using dynamic programming



Problem J. Jokewithpermutation

- This problem is solved with exhaustive search
  - for each number try all positions that it can occupy
  - start search with numbers that can occupy fewest number of possible positions

Problem K. Knockout Racing

- Nothing fancy here
- Just implement what the problem statement asks for in a straightforward way
- This is the easiest problem in the contest