Flow Contest Editorial

Maxim Akhmedov

September 28, 2018

1 Problem A. Maximum Flow

Just implement the maximum flow algorithm. Note that in order to deal with undirectness of a graph, you should duplicate each undirected edge creating two its directed copies in different directions. This may result in flow going into both directions through this edge, but all you need is the difference of these two flows.

Also mind the multiple edges.

2 Problem B. Minimum Cut

Find the maximum flow and then find the minimum cut by running the DFS from the source going only through non-saturated edges. Due to Ford-Fulkerson theorem, the cut between the visited vertices and unvisited will be the minimum one.

3 Problem C. Decomposition

Find a maximum flow and then start cutting off paths from it one by one. You may end up with some non-trivial circulation, but since you are allowed to choose any maximum flow you want, you may just discard the remaining circulation.

4 Problem D. Cards

There are 26 English letters and $26^2 = 576$ different kinds of cards. Create a bipartite graph whose left part vertices l_{ij} correspond to all of the kinds of cards and right part vertices r_j correspond to the letters. Connect s with a vertex l_i with an edge with capacity equal to the number of cards that have ij written on them, connect r_j to t with an edge with capacity equal to the number of letter j in a desired string. Finally, connect l_{ij} to r_i and r_j with an edge of infinite capacity.

5 Problem E. Cows

Build a graph corresponding to the stones and bridges. Now perform a following transformation: if there was a vertex v, replace it with two new vertices v^- and v^+ and put an edge (v^-, v^+) of unit capacity. Now if there was an edge (v, u) in the original graph, put an edge (v^+, u^-) of infinite capacity. As any path in the resulting flow blocks the unit capacity of all the visited vertices, all the paths will be disjoint.

6 Problem F. Matan

If theme v depends on theme u, put an edge (v, u) of an infinite capacity. Add a source vertex s and a sink vertex t. For any theme v, if it has a positive usefulness x_v, put an edge (v, t) with capacity x_v, otherwise put an edge (s, v) with capacity $-x_v$. Suppose we have a minimum $s - t$ cut (A, B). It is easy to see that if theme v depends on theme u and $v \in B$, then $u \in B$ (otherwise the cut value would be at least infinity).

Let us calculate the cost of the cut. Fix a theme v with $x_v < 0$. Then $-x_v$ is included in cut value if $v \in B$ and is not included otherwise. Now let v have $x_v > 0$. If $v \in A$, x_v is added to the cut value. It is equivalent to think that all x_v are included from the beginning and the fact that $v \notin A \Leftrightarrow v \in B$ subtracts x_v from the cut value. Thus, in both cases (positive or negative usefulness) the usefulness of a vertex in B is accounted in the cut value with a factor of -1.

Thus, the cut value is (sum of all positive usefulnesses $- \sum_{v \in B} x_v$, and the first value is constant not depending on cut choice. So, minimizing this value is equivalent to maximizing $\sum_{v \in B} x_v$ and, having a restriction on dependencies met, we reach the objective of the problem.

7 Problem G. Perspective

First of all, our team should win all matches and all of the remaining teams should lose all the matches with the other divisions.

Now we know an upper limit d for the score for any of the remaining teams and we are to distribute wins in some of the matches. Create a bipartite graph with its right part representing the teams, let the j-th team correspond to the vertex v_j and connect v_j to t with an edge of capacity d. Let the left part represent matches, namely, the match between teams i and j corresponds to the vertex u_{ij} and s is connected to u_{ij} with an edge of capacity a_{ij}. Each unit of flow going through u_{ij} represents one game and it should go to one of the v_i and v_j, thus connect u_{ij} to v_i and v_j with edges of infinite capacity.

Now take a look on a maximum flow in this graph. If it saturates all the edges from source, then we have a distribution of wins such that no team have won more than d times, otherwise we can't have such a distribution.

8 Problem H. Broken Parquet

First, if $a \geq 2b$, then it is always optimal to replace a tile 1×2 with two tiles 1×1, thus the answer is simply the area of the field times b. Otherwise, we want to maximize the number of tiles 1×2, i.e. the problem is equivalent to finding the most number of disjoint dominoes in a given field. This problem may be easily reduced to a matching problem: consider a graph on cells and connect the pairs of adjacent cells with edges. Any matching corresponds to a correct domino set and vice versa. Finally, this graph is bipartite because of chessboard coloring. So we may need to simply find the maximum matching, but the constraints are pretty tight, so we have to find it using the Hopcroft-Karp algorithm which is equivalent to Dinic algorithm applied to a corresponding network.

9 Problem I. Full Orientation

Let's find the minimum possible maximum degree with a binary search. Fix some upper limit d for degree. We now want to assign each edge to one of its endpoints, so create a bipartite graph with left part representing edges e_{ij} and right part representing vertices v_i. Connect s with e_{ij} with an edge of capacity 1, connect v_i with t with an edge of capacity d and connect e_{ij} with both v_i and v_j with an edge of unit capacity. If the maximum flow saturates all edges from the source, the desired assignment exists and we should decrease d, otherwise we should increase d.

10 Problem J. Binary Tree on Plane

Binary tree is such an acyclic graph in which any vertex has an ingoing degree of at most 1, an outgoing degree of at most 2 and which contains exactly $n - 1$ edges. We can formulate it in terms of flow: put an edge from s to any vertex v of capacity 1 and from any vertex v to the sink with capacity 2. Also connect each vertex v with all vertices u below it with an edge of capacity 1. Then any flow of value $n - 1$ will correspond to some correct binary tree.

Finally, assign to each of the edges between vertices a cost equal to the Euclidean length of the edge. Now the problem became equivalent to finding the flow of value $n - 1$ of minimum cost.

11 Problem K. Hard Life

A key idea: let us guess the answer with a binary search. Does there exist a set with a hardness (that is normally called density) of at least α? If this graph contains a edges and b vertices, the condition of $\frac{a}{b} \geq \alpha$ equivalent to $a - \alpha b \geq 0$. Let's express the chosen subgraph as an A-component of some cut (A, B). All we need is to make the vertices inside A account with a weight 1 and the edges whose both endpoints are inside A account with a weight of $-\alpha$. This is an exercise for you, for full details refer to: `https://www2.eecs.berkeley.edu/Pubs/TechRpts/1984/CSD-84-171.pdf`