Code Jam 2008 - World Finals

Analysis: The Year of Code Jam

The different appearances of perimeter

In spite of the nice story about the calendar and the fancy definition of happiness, we hope that you have discovered the following abstract description of this problem.

> On a board of **N** by **M** unit squares, some squares are blue, some are white, and some are undecided. For each undecided square, color it either blue or white, so that the total perimeter of blue region is maximized.

We see that the perimeter can be defined in two ways.

- (a) The sum of the contributions from each blue square, as indicated in our problem statement.
- (b) The sum of the contributions from each unit segment. Count 1 for each unit segment that separates two squares with different colors. (Assume all the squares outside the board are white.)

The second interpretation is useful for the following analysis.

There are some simple observations that seem useful. For example, for any undecided unit square, if we have already decided that two of its neighbors are blue, then in one optimal solution we can assign it to be white. Yet, as far as we know, no such heuristics give a complete and fast solution to our problem.

A similar problem

Let us discuss a problem that at least looks similar to ours. What if we wanted to minimize the perimeter instead of maximizing it?

We rephrase the problem in terms graph theory. Let the set of **NM** unit squares be our vertices, and let there be an edge between two adjacent squares. Our task is to color each undecided vertex either blue or white, so that the number of edges (in the graph) between the blue vertices and the white vertices is minimized.

This is rather nice, isn't it? If you add a source **s** and a destination **t**, add one edge from **s** to every blue vertex with infinite capacity (4 is enough) and one edge from every white vertex to **t** with infinite capacity, then the problem is asking for a *minimum cut* from **s** to **t**, which can be solved by your favorite max flow algorithm.

Solving the original problem

While minimum cut is polynomial-time solvable by max flow, the max-cut problem is too hard in general graphs. Therefore, our problem still seems much harder than the minimization version of the problem.

However, we have not used an important fact yet. Our graph is far from arbitrary. We play this game on the **N** by **M** board; the resulting graph is bipartite (imagine it as a chessboard). Maximizing the number of edges between different colored vertices is the same as minimizing the number of edges between vertices of the same color. In other words, if we flip the colors of

one half of the chessboard, the problem reduces to the minimization version we discussed above.

We mark the board in a chessboard fashion, label the unit squares as odd or even. Then we flip the color of all the even squares. Now look at property (b). A contribution becomes a non-contribution and vice versa. The total number of edges is fixed, so the maximization problem reduces to its minimization version.