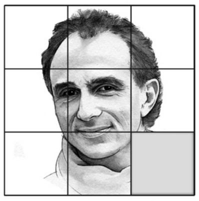


DOI:10.1145/3133244

Dennis Shasha

Upstart Puzzles
Partitioned Peace

IN A MYTHICAL land of rhetorically encouraged antagonism, different factions manage to co-exist, though poorly. Imagine a set of red and blue hill towns connected by a network of roads. People in the red towns deal well with one another. People in the blue towns deal well with one another. But when a person from a red town travels through a blue town or vice versa, things can get unpleasant. The leaders of the red and the blue towns get together and decide the best way to resolve their differences is to perform a series of swaps in which the inhabitants of *k* red towns swap towns with the inhabitants of *k* blue towns with the end result that a person from a blue town can visit any other blue town without passing through a red town and likewise for a person from a red town. We call such a desirable state "partitioned peace." The goal is to make *k* as small as possible.

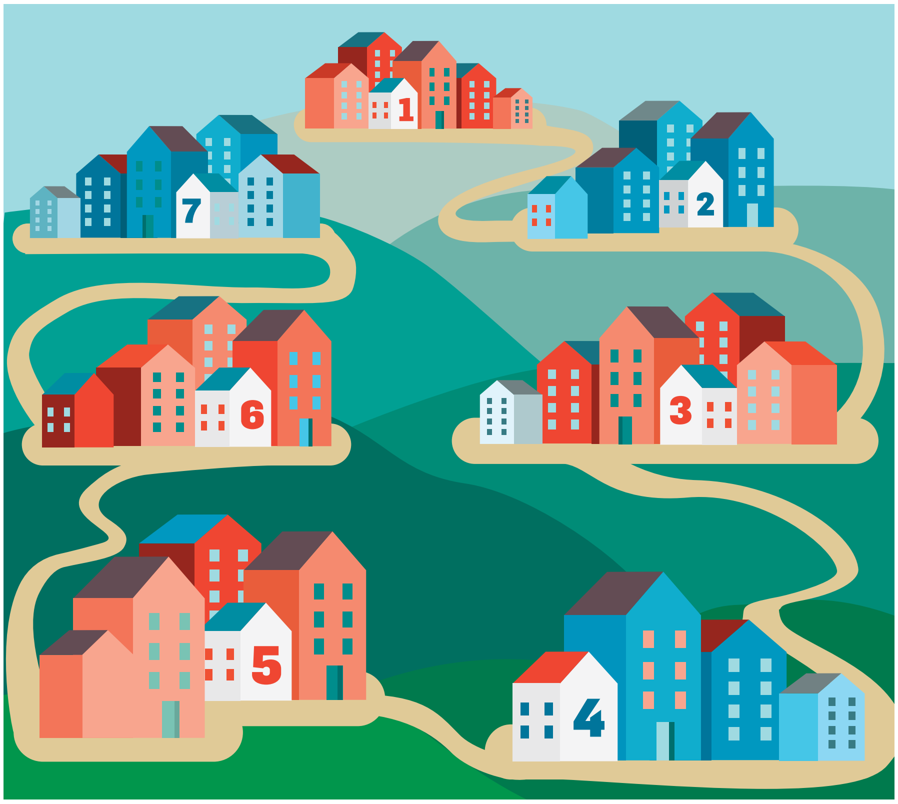
Warm-Up 1. Given the configuration in the figure here, what is the minimum number of swaps needed to achieve partitioned peace?

Solution to Warm-Up 1. Two swaps are sufficient: Red_1 with Blue_7 and Red_3 with Blue_4.

Because exchanging town populations is painful for the people who must move, the leaders seek other arrangements; they are willing to build, for example, a certain number of roads to reduce the number of swaps.

Warm-Up 2. Given the configuration in the figure, what is the minimum number of swaps needed to achieve partitioned peace if you were able to build a single new road?

Solution to Warm-Up 2. Build a road between Blue_7 and Red_1 and then swap Red_1 with Blue_4. The red town dwellers can travel to other red towns without crossing through blue towns. Likewise, the blue town dwellers can travel to other blue towns without having to cross through red towns.

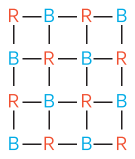


What is the minimum number of towns you can exchange so red and blue travelers never need to cross the other color's towns?

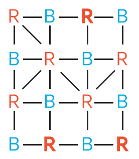
Can you create an algorithm that will perform a minimum number of swaps to achieve partitioned peace?

Challenge: So far, we have considered only a very simple configuration of towns; now consider that the red and blue towns alternate like the squares of a four-by-four checkerboard. Every town is connected to all its vertical and horizontal neighbors. You may build eight new roads between diagonally neighboring towns. Where should the roads go? And after the roads are built, which towns should swap populations to minimize the number of swaps needed to achieve partitioned peace, where the swaps are between towns directly connected by roads? The [CONTINUED ON P. 111]

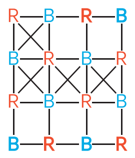
[CONTINUED FROM P. 112] goal of any solution is to minimize swaps.



Solution to challenge. Numbering the rows from top to bottom and left to right, we can build roads between red towns from (1,1) to (2,2), (2,4) to (3,3), (2,2) to (3,3), and (3,1) to (2,2). This leaves the red towns at (1,3), (4,2), and (4,4) surrounded by blue towns, highlighted in bold, in the following grid



Now build roads between the blue towns from (1,2) to (2,1), from (2,1) to (3,2), from (2,3) to (3,2), and from (2,3) to (3,4). This leaves certain towns isolated from their own colors, like those highlighted in bold in this grid



The two sides then need just three swaps: red and blue on top and the pairs on the bottom.

Upstart 1. Suppose you are given an arbitrary planar graph of connections and an arbitrary red/blue coloring of nodes. You are also given a budget of r planar edges you can add. Can you create an algorithm (and implementation) that will perform a minimum number of swaps to achieve partitioned peace? If so, please explain it in pseudo-code and send links to platform-independent software.

Upstart 2. Now consider the same question as in Upstart, 1 but allow non-planar edges.

All are invited to submit their solutions to upstartpuzzles@cacm.acm.org; solutions and discussion will be posted at http://cs.nyu.edu/cs/faculty/shasha/papers/cacmpuzzles.html

Dennis Shasha (dennisshasha@yahoo.com) is a professor of computer science in the Computer Science Department of the Courant Institute at New York University, New York, as well as the chronicler of his good friend the omniheurist Dr. Ecco.