Flow Report

Dennis Nguyen - Thor Olesen - Daniel Hansen - Nicoline Scheel - Maria Techt

October 17, 2017

Results

Our implementation successfully computes a maximum flow of 146 on the input file, comrades! It does not confirm the analysis of the American enemy given by result.txt, as we get the minimum cut:

24 33 17
28 30 19
29 30 5
31 41 10
32 40 29
34 39 36
38 46 30

We have analysed the possibilities of descreasing the capacities near Minsk. Our analysis is summaries in the following table:[1]

[1] Complete and correct the table.

Case	4W–48	4W–49	Effect on flow
1	30	20	no change
2	20	30	no change
3	20	20	no change
4	10	20	−10
5	20	10	−10
6	10	10	−20

In case 3, the new bottleneck becomes

1–18, 12–23, 17–18, 20–21, 20–23, 24–33, 28–30, 29–30, 31–41, 32–40

The rails between 4W–48 & 4W–49 has now become bottlenecks and will directly influence the flow. The comrade from Minsk is discouraged to decrease the flow on these two rails, because they are not part of the minimum cut. For the good of Mother Russia, the capacity of the rails specified by the minimum cut should be decreased for the largest impact on the enemy rails.

Implementation details

We use a straighforward implemenation of Ford Fulkerson's flow algorithm using a double int array as in described in Algorithm

Design by Jon Kleinberg, Eva Tardos, chap. 7. We use Breadth First Search algorithm to find an augmenting path.

The running time of the Ford Fulkerson implementation is

$$O(f^{'} * (m + n)) \tag{1}$$

where f' is maximum flow times the number of BFS (m+n)

We have implemented each directed edge as a spot in the int[][] array. That is, the edge from u to v has a capacity c which is graph[u][v].

In the corresponding residual graph, the edge is represented in the same way, since the residual graph is an int[][] as well. The opposite edge of residualGraph[u][v] is residualGraph[v][u].