Flow Report

Bjarke Brodin (bjal), Bjørnar Haugstad Jåtten (bjja), Helle Friis (hefr) and Simon Boye Jørgensen (sboj)

October 14, 2022

Results

Our implementation successfully computes a flow of 163 on the input file, confirming the analysis of the American enemy. For the motherland.

We have analysed the possibilities of decreasing the capacities near Minsk. Our analysis is summaries in the following table:

Case	4W–48	4W–49	Effect on flow
1	30	20	no change
2	30	10	no change
3	20	30	no change
4	10	30	no change
5	20	20	no change
6	20	10	153 (-10)
7	10	20	153 (-10)
8	10	10	143 (-20)

In case 6, the new bottleneck becomes

10–2, 11–24, 11–25, 12–23, 18–22, 20–21, 20–23, 27–2, 28–30

In case 7, the new bottleneck becomes

10 25, 11–24, 11–25, 12–23, 18–22, 20–21, 20–23, 27–2, 28–30

In case 8, the new bottleneck becomes

10–25, 11–24, 11–25, 12–23, 18–22, 20–21, 20–23, 27–26, 28–30

The comrade from Minsk is advised to use case 2, 4 or 5, to decrease the capacity as much as possible while preserving the maximum flow.

Implementation details

We use a straightforward implementation of Ford-Fulkerson's flow algorithm as described in Kleinberg and Tardos, *Algorithms Design*, S.7.1. We use Depth-First-Search to find an augmenting path and also implement the neat trick of flow by [. . .][1]

[1] Replace by a description of what you actually do.

The running time is [...].

We have implemented each undirected edge in the input graph as two directed edges in each direction. In the corresponding residual graph, the edge is represented by an object of the Edge class. Our datatype for edge is this:

```
  class Edge {
    int u,v,c;
    public Edge(int u, int v, int c, int f) {
      this.u = u; this.v = v; this.c = c;
    }

    @Override
    public String toString() {
      return u + " " + v + " " + c;
    }
  }
Compares! For the motherland!
```