Red Scare! Report

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

November 15, 2022

Results

The following table gives my results for all graphs of at least 500 vertices.

Instance name	n	A	F	M	N	S
rusty-5762	5,762	true	16	–	?	5
wall-p-10000	10,000					
⋮						

The columns are for the problems Alternate, Few, Many, None, and Some. The table entries either give the answer, or contain '?' for those cases where I was unable to find a solution within reasonable time. For those questions where there is a reason for my inability to find a good algorithm (because the problem is hard), I wrote '?!'.

For the complete table of all results, see the tab-separated text file `results.txt`.

Methods

For all solutions we process the input to produce a graph G, which we assume to be simple.

None

We solve this problem efficiently for all graphs. Here we did not put any of the red vertex in the graph. Then when running the shortest path algorithm we were guaranteed to find the shortest path without any red vertexes in the path if there exists one.

We use BFS for the shortest path algorithm which has a running time of $O((|E| + |V|) \log |V|)$.

Alternate

To solve alternate, we first removed all reflective edges, meaning the edges that go from black to black or red to red. Afterwards, we applied a shortest path algorithm. If there exists a path between t and

s, it must be alternating between black and red, as all the relfective edges are removed from the graph.

Running time is dominated by shortest path and is thus simply $O((|E| + |V|) \log |V|)$.

Few

To solve *few*, we realize that we can manipulate edge weights to allow a reduction to shortest path. We set weights to 1 for all arcs that are incident on a red vertex and all other weights to 0, then simply run an appropriate shortest path algorithm and return the distance. Running time is dominated by shortest path and is thus simply $O((|E| + |V|) \log |V|)$.

Many

We solve many for directed acyclic graphs, by using much the same approach as in few, only, instead of setting 1 weighted red arcs, we use -1 weighted red arcs. Running shortest path then yields the negation of our desired result. We take care to use a shortest path algorithm that does not stop early when reaching a valid path, but rather examines all options.

In the general case we cannot solve many, because the general case includes $R = G$. Solving many in this case implies solving the longest path problem exactly, and since this is known to be NP-hard we can safely give up.

Some

First we just on the background of many decided wether that Some was possible or not. Now we solve it in a more independent way by making it to a flow problem. ...

References

1. Something
2. Something else
3. Something else else