Red Scare! Report

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

November 10, 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.

The runnning time for the shortest path algorithm is the same as in "None", as we use the same algorithm.

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

Some

For problem A, I solved each instance G by $\cdots$[1] The running time of this algorithm is $\cdot$, and my implementation spends $\cdots$ seconds on the instance $\cdots$ with $n = \cdots$.

I solved problem $\cdots$ for all $\cdots$[2] graphs using $\cdots$.

I was unable to solve problem $\cdots$ except for the $\cdots$ instances. This is because, in generality, this problem is $\cdots$. To see this, consider the following reduction from $\cdots$. Let ...

I was also unable to solve $\cdots$ for $\cdots$, but I don't know why.[3]

[1] Describe what you did. Use words like "building a inverse anti-tree without self-loops where each vertex in G is presented by a Strogatz–Wasserman shtump. I then performed a standard longest hash sorting using the algorithm of Bronf (Algorithm 5 in [1])." Be neat, brief, and precise.

[2] For instance, "planar, bipartite"

[3] Remove or expand as necessary.

References

1. *APLgraphlib—A library for Basic Graph Algorithms in APL*, version 2.11, 2016, Iverson Project, `github.com/iverson/APLgraphlib`.[4]

2. A. Lovelace, *Algorithms and Data Structures in Pascal*, Addison–Wesley 1881.

[4] If you use references to code, books, or papers, be professional about it. Use whatever style you want, but be consistent.