Red Scare! Report

by Alice Cooper.

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-separted text file `results.txt`.

Methods

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.