This code solves the Resourceful Students - heuristic

The code is separated logically into two parts:

- If N<5000: Simulated annealing method.

The code builds an initial solution by greedily matching pairs of vertices that have not been matched yet. Then, it attempts to improve (reduce) the number of chosen edges by repeatedly: removing a randomly chosen edge (if the result still satisfies the constraints) or swapping a chosen edge with an unchosen edge (if the result remains valid). There is a time limit of 27 seconds. The best valid solution found so far is then output.

- If N≥5000: Priority queue–based heuristic.

For each edge (u,v), it defines a priority deg(u)+deg(v).
It repeatedly picks the edge with the largest priority and, if both endpoints are still unmatched, selects that edge.
Once an edge (u,v) is chosen, all edges adjacent to u and v become inactive (they are no longer considered). This continues until no more edges can be chosen.
The chosen set of edges is output as the representation.

The function **isIndependentSet** ensures that no two unchosen vertices remain adjacent. The function **buildInitialSolution** greedily selects edges to form an initial matching. The function **tryRemoveEdge** tries removing a chosen edge, while **trySwapEdge** attempts swapping one chosen edge with an unchosen one. The function **attemptMove** picks either removal or swapping and updates the best solution if it improves. The function calcTemp calculates a temperature for simulated annealing. For large graphs, **solveLarge** uses a priority-based heuristic, repeatedly picking edges with the highest degree sum. And the function **main** reads input, decides which approach to use, and outputs the chosen pairs.

Other approaches that have been tried

- Max-matching + 2-approximation. The problem was that 2-approximation didn't improve the initial matching.
- Simulated annealing. It works efficiently for smaller graphs. It is not able to minimize the number of pairs for extremely large inputs
- Simulated annealing for smaller graphs, max-matching for large ones. The solution was fast enough, yet the implementation didn't minimize the number of pairs correctly
- Fast min vertex cover with heuristics. Was not fast enough and the implementation didn't minimize the number of pairs correctly
- Priority Queue heuristic (multiple versions like: sum of degrees in edge, sum of degrees in edge minus n of common neighbours, the previous 2 but with some random perturbations, previous ones but with priority modifier if for example degree of node was 1). It works well for large inputs, and was able to beat swats2, but it was possible to optimize the solution for smaller graphs using simulated annealing.

This solution was submitted on optil.io by Dariia Shevchuk under the nickname of dashashevchuk. Score: 93.97, time: 774.55 seconds.

The group:
Cezary Suchorski
Michał Żarnowski
Dariia Shevchuk
Piotr Szymiec