Railway Planning Report

Adam Tovatt

May 12, 2025

1 Results

All test cases passed when I ran the provided `check_solutions.sh` script. My program successfully completed even the largest test case with 1,000 nodes, 100,000 edges, and 10,000 planned removals. On my machine, the runtime for the largest case was approximately 156 milliseconds. The majority of the time is spent solving the max-flow problem repeatedly during the binary search loop.

Section	**Time (ms)**	**Percent of Total**
readInput	17.5	11.1%
solve	139	88.9%
fullSolve	156	100.0%

2 Implementation details

I implemented the Goldberg–Tarjan Push–Relabel algorithm (FIFO variant) to compute the maximum flow. I used a capacity matrix `int[,]` for fast edge lookup and cloning. To find the maximum number of removable routes while maintaining enough flow, I performed a binary search over the removal plan, invoking the max-flow algorithm on each iteration.

Data structures

- `int[,]` capacity matrix — for edge capacities
- `int[,]` flow matrix — for current flow state
- `int[]` excess — amount of flow waiting to be pushed per node
- `int[]` height — height label per node to guide push direction
- `Queue<int>` — active node queue in FIFO order

Modifications

- I initialized the preflow by pushing full capacity from the source to its neighbors.
- I tracked which neighbor to push to next using a pointer array per node.
- I used binary search over the removal plan to reduce the number of max-flow calls from $O(P)$ to $O(\log P)$.

Time Complexity

The Push–Relabel algorithm I used runs in $O(V^2 \cdot E)$, where V is the number of nodes and E is the number of edges. Each node can be relabeled at most $2V - 1$ times, and pushes occur at most $O(V \cdot E)$. Since I binary search over P planned removals, the total running time becomes $O(V^2 \cdot E \cdot \log P)$.

Discussion Questions

What is the time complexity, and why?
The time complexity is $O(V^2 \cdot E \cdot \log P)$. Push–Relabel runs in $O(V^2 \cdot E)$ in the worst case, and I use it inside a binary search loop of depth $O(\log P)$ to find the maximum number of removable routes.

Which other well-known algorithmic problems can be solved using Network Flow?
Bipartit graphs. Network flow can also be used in modeling flow in for exampeling water pipes and electrical systems. It can also be used in ecological applications modeling flow of nutrients.

If the capacities of the edges are very large, how can one get a different (better) time complexity?
In some max-flow algorithms like Ford–Fulkerson, large edge capacities can cause poor performance if only one unit of flow is pushed at a time. To address this, capacity scaling techniques can be used, which push flow in large chunks and reduce the number of iterations. Push–Relabel algorithms like mine already push full available flow in each step, so they are not directly affected by large capacities in the same way.