Algorithms SV worksheet 5

Bálint Molnár

August 30, 2025

I understand that the order in which the topics are taught has changed slightly from previous years. Before attempting the questions, make sure you know the following concepts/definitions: DAG, Minimum Spanning Tree/Kruskal's Algorithm, Max-flow problem/Ford-Fulkerson algorithm, Topological sort, Strongly Connected Components, Bipartite Graph Matching. You may want to use the notes from last year.

1. If we take a DAG and reverse all the edges, do we get a DAG? Justify your answer.
2. Let G be a weighted graph with positive weights. Prove that, if you replace all the weights with their squares, the the resulting graph has the same Minimum Spanning Tree.
3. You are given an $N \times N$ grip represented y a matrix A consisting of 0s and 1s. You want to choose N cells in such a way that
   - You choose exactly one cell from every row.
   - You choose exactly one cell from every column.
   - You can only choose cells where the corresponding entry is 1.

   How would you find such a set of cells?
4. The Russian mathematician A.N. Tolstoı introduced the following problem in 1930. Consider a directed graph with edge capacities, representing the rail network. There are three types of vertex: supplies, demands, and ordinary interconnection points. There is a single type of cargo we wish to carry. Each demand vertex v has a requirement $d_v > 0$. Each supply vertex v has a maximum amount it can produce s_v. Tolstoı asked: can the demands be met, given the supplies and graph and capacities, and if so then what flow will achieve this? Explain how to translate Tolstoı's problem into a max-flow problem of the sort we studied in section 6.2 (last year's notes) .
5. *(Difficult.)* You are given a DAG $G = (V, E)$, and a list of special nodes, $[a_1, ..., a_k]$. It is guaranteed that, for all $1 \leq 1 < k$, there is a path from a_i to a_{i+1}.

   Design an algorithm that finds a topological sort of G which minimises the ranks for all a_i in the resulting list.
6. *(Difficult.)* In a country, there are direct train routes between railway stations, each with a given travel time (The train network is a weighted undirected graph). We define the distance between two nodes as the length of the shortest path between them.

   Train tickets in this country specify only the departure and destination stations. Travelers can take multiple routes between these stations, but with one restriction: at no point during the journey can they travel farther from the destination than they were when they boarded the train (i.e. you cannot take a train from city C to D if D is further away from your destination than C. You are allowed to take a train multiple times as long as this constraint is not violated).

   You have a ticket between cities A and B. The country has beautiful railway stations and you want to visit as many of them as possible.

   Design an algorithm that calculates the maximum number of stations you can visit using a ticket between cities A and B.