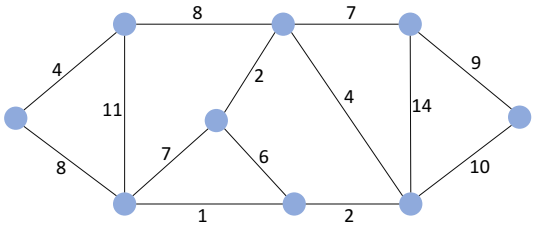
Example sheet week 6

Trees. DAGs. Flows.
Algorithms—DJW*–2017/2018

Questions labelled ∘ are warmup questions. Questions labelled ∗ involve more thinking and you are not expected to attempt all of them. The questions are arranged in the order that the course is taught; consult your supervisor to find out which questions to answer for which supervision.

Question 1 (FS55, FS56)∘. Try to find, by hand, a minimum spanning tree for this graph. Now run, by hand, the Kruskal and Prim algorithms.



Question 2. In an undirected graph with edge weights ≥ 0, let u–v be a minimum-weight edge. Show that u–v belongs to a minimum spanning tree.

Question 3. Prove the final claim in Section 5.8 (Prim's algorithm), that $\hat{f}$ is a spanning tree. *[A model solution will be provided at the end of the course.]*

Question 4 (FS53)∘. Give an example DAG with 9 vertices and 9 edges. Pick some vertex that has one or more edges coming in, and run through `toposort` starting on line 6 with this vertex. Mark each vertex with two numbers as you proceed: the discovery time (when the vertex is coloured grey) and the exit time (when the vertex is coloured black). Then draw a linearized DAG by arranging the vertices on a line in order of their finishing time, and reproducing the appropriate arrows between them. Do all the arrows go in the same direction?

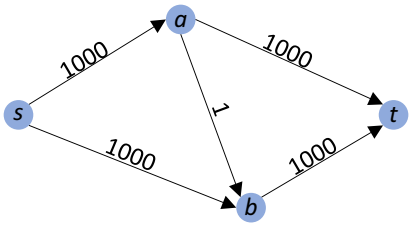
Question 5∗. Sometimes we want to impose a total order on a collection of objects, given a set of pairwise comparisons that can be thought of as a 'DAG with noise'. For example, let vertices represent movies, and write $v_1 \to v_2$ to mean "The user has said she prefers v_1 to v_2." A user is likely to give answers that are by and large consistent, but with some exceptions. Discuss what properties you would like in an 'approximate total order', and how you might go about finding it. *[This is an open-ended question, and a prelude to data science and machine learning courses.]*

Question 6. Give pseudocode for an algorithm that takes as input an arbitrary directed graph g, and returns a boolean indicating whether or not g is a DAG.

Question 7 (FS54)∗. Write out a formal proof of the correctness of the `toposort` algorithm, filling out all the details that are skipped over in the handout. Pay particular attention to the third case, 'v_2 is coloured grey', where it is claimed "The call stack corresponds to a path in the graph from v_2 to v_1."

*Questions labelled FS are from Dr Stajano. Questions labelled CLRS are from *Introduction to Algorithms, 3rd ed.* by Cormen, Leiserson, Rivest and Stein.

Question 8°. Use the Ford-Fulkerson algorithm, by hand, to find the maximum flow from s to t in the following graph. How many iterations did you take? What is the largest number of iterations it might take, with unfortunate choice of augmenting path?



Question 9°. Consider a flow f on a directed graph with source vertex s and sink vertex t. Let $f(u \to v)$ be the flow on edge $u \to v$, and set $f(u \to v) = 0$ if there is no such edge.

(i) Show that

$$\sum_{w \neq s,t} \left(\sum_v f(w \to v) - \sum_v f(v \to w) \right) = 0.$$

(ii) The value of the flow is defined to be the net flow out of s,

$$\text{value}(f) = \sum_u f(s \to u) - \sum_u f(u \to s).$$

Prove that this is equal to the net flow into t. *[Hint. Add the left hand side of the equation from part (i).]*

Question 10. Prove the assertion on line 25 of the `ford_fulkerson`, i.e. that after adding flow δ along an augmenting path, we end up with a valid flow.

Question 11*. Devise an algorithm that takes as input a flow f on a network, and produces as output a decomposition $[(f_1, p_1), \ldots, (f_n, p_n)]$ where each p_i is a path from the source to the sink, and each f_i is a positive number. The decomposition must satisfy $f = \sum_i f_i p_i$, by which we mean "put flow f_i along path p_i, and add together all these flows-along-paths, and the answer must be equal to f". Explain why your algorithm works.

Question 12. The Russian mathematician A.N. Tolstoy 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 $s_v > 0$. Tolstoy 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 Tolstoy's problem into a max-flow problem of the sort we have studied.

Question 13. In the London tube system (including DLR and Overground), there are occasional signal failures that prevent travel in either direction between a pair of adjacent stations. Find the minimum number of disruptions that will prevent travel between Kings Cross and Embankment. Justify your answer carefully using a max-flow formulation. *[Hint. Read the Analysis section of Section 6.1.]*

Question 14*. Consider a bipartite graph, in which edges go between the left vertex set L and the right vertex set R. A matching is called *complete* if every vertex in L is matched to a vertex in R, and vice versa. For a complete matching to exist, we obviously need $|L| = |R|$. The following result is known as Hall's Theorem:

> A complete matching exists if and only if, for every subset $X \subseteq L$, the set of vertices in R connected to a vertex in X is at least as big as X.

Prove Hall's Theorem, using a max-flow formulation. *[Hint: Use the same construction as we used in lectures, except with capacity ∞ on the edges between L and R. In this graph, some cuts have infinite capacity, and some cuts have finite capacity. If a cut has finite capacity, what can you deduce about its capacity?]*