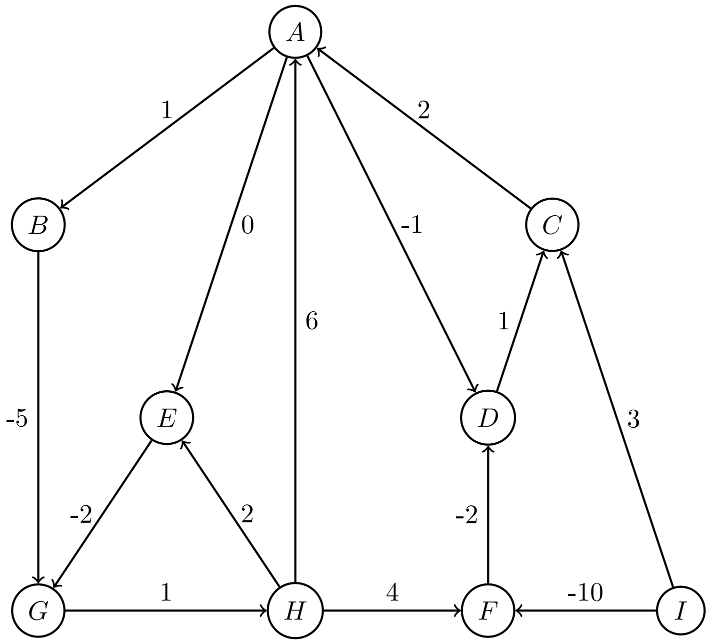
Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Bellman-Ford Practice

(a) Run the Bellman-Ford algorithm on the following graph, from source A. Relax edges (u, v) in lexicographic order, sorting first by u then by v.



(b) What problem occurs when we change the weight of edge (H, A) to 1? How can we detect this problem when running Bellman-Ford? Why does this work?

(c) Let $G = (V, E)$ be a directed graph. Under what condition does the Bellman-Ford algorithm returns the same shortest path tree (from source $s \in V$) regardless of the ordering on edges?

2 Finding Counterexamples

In this problem, we give example greedy algorithms for various problems, and your goal is to find an example where they are not optimal.

(a) In the travelling salesman problem, we have a weighted undirected graph $G(V, E)$ with all possible edges. Our goal is to find the cycle that visits all the vertices exactly once with minimum length.

One greedy algorithm is: Build the cycle starting from an arbitrary start point s, and initialize the set of visited vertices to just s. At each step, if we are currently at vertex u and our cycle has not visited all the vertices yet, add the shortest edge from u to an unvisited vertex v to the cycle, and then move to v and mark v as visited. Otherwise, add an edge from the current vertex to s to the cycle, and return the now complete cycle.

(b) In the maximum matching problem, we have an undirected graph $G(V, E)$ and our goal is to find the largest matching E' in E, i.e. the largest subset E' of E such that no two edges in E' share an endpoint.

One greedy algorithm is: While there is an edge $e = (u, v)$ in E such that neither u or v is already an endpoint of an edge in E', add any such edge to E'. (Can you prove that this algorithm still finds a solution whose size is at least half the size of the best solution?)

3 Select Activity

Assume there are n activities each with its own start time a_i and end time b_i such that $a_i < b_i$. All these activities share a common resource (think computers trying to use the same printer). A feasible schedule of the activities is one such that no two activities are using the common resource simultaneously. Mathematically, the time intervals are disjoint: $(a_i, b_i) \cap (a_j, b_j) = \emptyset$. The goal is to find a feasible schedule that maximizes the number of activities k.

Here are two potential greedy algorithms for the problem.

Algorithm A: Select the shortest-duration activity that doesn't conflict with those already selected until no more can be selected.

Algorithm B: Select the earliest-ending activity that doesn't conflict with those already selected until no more can be selected.

(a) Show that Algorithm A can fail to produce an optimal output.

(b) Show that Algorithm B will always produce an optimal output.

(c) **Challenge Problem:** Show that Algorithm A will always produce an output at least half as large as the optimal output.

4 Unique Shortest Path

Shortest paths are not always unique: sometimes there are two or more different paths with the minimum possible length. Show how to solve the following problem in $O((|V| + |E|) \log |V|)$ time.

Input: An undirected graph $G = (V, E)$; edge lengths $l_e > 0$; starting vertex $s \in V$.

Output: A Boolean array `usp[·]`: for each node u, the entry `usp[`u`]` should be `true` if and only if there is a *unique* shortest path from s to u. (Note: `usp[`s`] = true`.)

[Provide 3 part solution.]