(1) We can think about a more general version of the stable matching problem, in which certain man-woman pairs are explicitly *forbidden*. In the case of employers and applicants, picture that certain applicants simply lack the necessary qualifications or degree; and so they cannot be employed at certain companies, however desirable they may seem. Concretely, we have a set M of n men, a set W of n women, and a set $F \subseteq M \times W$ of pairs who are simply *not allowed* to get married. Each man m ranks all the women w for which $(m, w) \notin F$, and each woman w' ranks all the men m' for which $(m', w') \notin F$.

In this more general setting, we say that a matching S is *stable* if it does not exhibit any of the following types of instability.

(i) There are two pairs (m, w) and (m', w') in S with the property that m prefers w' to w, and w' prefers m to m'. *(The usual kind of instability.)*

(ii) There is a pair $(m, w) \in S$, and a man m', so that m' is not part of any pair in the matching, $(m', w) \notin F$, and w prefers m' to m. *(A single man is more desirable and not forbidden.)*

(ii$'$) There is a pair $(m, w) \in S$, and a woman w', so that w' is not part of any pair in the matching, $(m, w') \notin F$, and m prefers w' to w. *(A single woman is more desirable and not forbidden.)*

(iii) There is a man m and a woman w, neither of which is part of any pair in the matching, so that $(m, w) \notin F$. *(There are two single people with nothing preventing them from getting married to each other.)*

Note that under these more general definitions, a stable matching need not be a perfect matching.

Now we can ask: for every set of preference lists and every set of forbidden pairs, is there always a stable matching? Resolve this question by doing one of the following two things: (a) Giving an algorithm that, for any set of preference lists and forbidden pairs, produces a stable matching; or (b) Giving an example of a set of preference lists and forbidden pairs for which there is no stable matching.

(2) There are many other settings in which we can ask questions related to some type of "stability" principle. Here's one, involving competition between two enterprises.

Suppose we have two television networks; let's call them AOL-Time-Warner-CNN and Disney-ABC-ESPN, or $\mathcal{A}$ and $\mathcal{D}$ for short. There are n prime-time programming slots, and each network has n TV shows. Each network wants to devise a *schedule* — an assignment of each show to a distinct slot — so as to attract as much market share as possible.

Here is the way we determine how well the two networks perform relative to each other, given their schedules. Each show has a fixed *Nielsen rating*, which is based on the number of people who watched it last year; we'll assume that no two shows have exactly the same rating. A network *wins* a given time slot if the show that it schedules for the time slot has a larger rating than the show the other network schedules for that time slot. The goal of each network is to *win* as many time slots as possible.

Suppose in the opening week of the fall season, Network $\mathcal{A}$ reveals a schedule S and Network $\mathcal{D}$ reveals a schedule T. On the basis of this pair of schedules, each network wins certain of the time slots, according to the rule above. We'll say that the pair of schedules (S, T) is *stable* if neither network can unilaterally change its own schedule and win more time slots. That is, there is no schedule S' so that Network $\mathcal{A}$ wins more slots with the pair (S', T) than it did with the pair (S, T); and symmetrically, there is no schedule T' so that Network $\mathcal{D}$ wins more slots with the pair (S, T') than it did with the pair (S, T).

The analogue of Gale and Shapley's question for this kind of stability is: For every set of TV shows and ratings, is there always a stable pair of schedules? Resolve this question by doing one of the following two things: (a) Giving an algorithm that, for any set of TV shows and associated ratings, produces a stable pair of schedules; or (b) Giving an example of a set of TV shows and associated ratings for which there is no stable pair of schedules.

(3) Some of your friends are working for CluNet, a builder of large communication networks, and they are looking at algorithms for switching in a particular type of input/output crossbar.

Here is the set-up. There are n *input wires* and n *output wires*, each directed from a *source* to a *terminus*. Each input wire meets each output wire in exactly one distinct point, at a special piece of hardware called a *junction box*. Points on the wire are naturally ordered in the direction from source to terminus; for two distinct points x and y on the same wire, we say that x is *upstream* from y if x is closer to the source than y, and otherwise we say x is *downstream* from y. The order in which one input wire meets the output wires is not necessarily the same as the order in which another input wire meets the output wires. (And similarly for the orders in which output wires meet input wires.)

Now, here's the switching component of this situation. Each input wire is carrying a distinct data stream, and this data stream must be *switched* onto one of the output wires. If the stream of Input i is switched onto Output j, at junction box B, then this stream passes through all junction boxes *upstream* from B on Input i, then through B, then through all junction boxes *downstream* from B on Output j. It does not matter which input data stream gets switched onto which output wire, but each input data stream must be switched onto a *different* output wire. Furthermore — and this is the tricky constraint — no two data streams can pass through the same junction box following the switching operation.

Finally, here's the question. Show that for any specified pattern in which the input wires and output wires meet each other (each pair meeting exactly once), a valid switching of the data streams can always be found — one in which each input data stream is switched onto a different output, and no two of the resulting streams pass through the same junction box. Additionally, give an efficient algorithm to find such a valid switching. (The accompanying figure gives an example with its solution.)

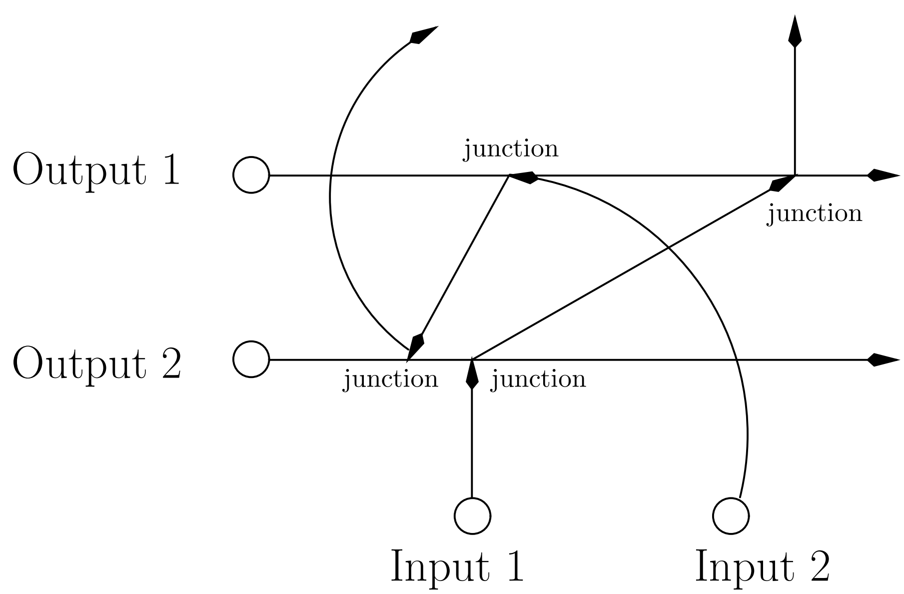

Figure 1: An example with two input wires and two output wires. Input 1 has its junction with Output 2 upstream from its junction with Output 1; Input 2 has its junction with Output 1 upstream from its junction with Output 2. A valid solution is to switch the data stream of Input 1 onto Output 2, and the data stream of Input 2 onto Output 1. On the other hand, if the stream of Input 1 were switched onto Output 1, and the stream of Input 2 were switched onto Output 2, then both streams would pass through the junction box at the meeting of Input 1 and Output 2 — and this is not allowed.