2. Suppose you have algorithms with the six running times listed below. (Assume these are the exact number of operations performed as a function of the input size n.) Suppose you have a computer that can perform 10^{10} operations per second, and you need to compute a result in at most an hour of computation. For each of the algorithms, what is the largest input size n for which you would be able to get the result within an hour?

(a) n^2

(b) n^3

(c) $100n^2$

(d) $n \log n$

(e) 2^n

(f) 2^{2^n}

7. 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 setup. 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.) Figure 1.8 gives an example of such a collection of input and output 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 problem. 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 algorithm to find such a valid switching.

strong instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a strong instability; or give an algorithm that is guaranteed to find a perfect matching with no strong instability.

(b) A *weak instability* in a perfect matching S consists of a man m and a woman w, such that their partners in S are w' and m', respectively, and one of the following holds:

- m prefers w to w', and w either prefers m to m' or is indifferent between these two choices; or
- w prefers m to m', and m either prefers w to w' or is indifferent between these two choices.

In other words, the pairing between m and w is either preferred by both, or preferred by one while the other is indifferent. Does there always exist a perfect matching with no weak instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a weak instability; or give an algorithm that is guaranteed to find a perfect matching with no weak instability.

5. The Stable Matching Problem, as discussed in the text, assumes that all men and women have a fully ordered list of preferences. In this problem we will consider a version of the problem in which men and women can be *indifferent* between certain options. As before we have a set M of n men and a set W of n women. Assume each man and each woman ranks the members of the opposite gender, but now we allow ties in the ranking. For example (with $n = 4$), a woman could say that m_1 is ranked in first place; second place is a tie between m_2 and m_3 (she has no preference between them); and m_4 is in last place. We will say that w *prefers* m to m' if m is ranked higher than m' on her preference list (they are not tied).

With indifferences in the rankings, there could be two natural notions for stability. And for each, we can ask about the existence of stable matchings, as follows.

(a) A *strong instability* in a perfect matching S consists of a man m and a woman w, such that each of m and w prefers the other to their partner in S. Does there always exist a perfect matching with no

3. 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, whom we'll call $\mathcal{A}$ and $\mathcal{B}$. 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 *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 $\mathcal{B}$ reveals a schedule T. On the basis of this pair of schedules, each network wins certain 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' such 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' such that Network $\mathcal{B}$ 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 the following: 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) give an algorithm that, for any set of TV shows and associated ratings, produces a stable pair of schedules; or

(b) give an example of a set of TV shows and associated ratings for which there is no stable pair of schedules.