perfect matching $M$ and house prices $P$ are in *equilibrium* if, for all edges $(x, y) \in M$ and all other houses $y'$, we have

$$v(x, y) - P(y) \geq v(x, y') - P(y').$$

But can we find a perfect matching and a set of prices so as to achieve this state of affairs, with every buyer ending up happy? In fact, the minimum-cost perfect matching and an associated set of compatible prices provide exactly what we're looking for.

**(7.67)** *Let $M$ be a perfect matching of minimum cost, where $c_e = -v(x, y)$ for each edge $e = (x, y)$, and let $p$ be a compatible set of prices. Then the matching $M$ and the set of prices $\{P(y) = -p(y) : y \in Y\}$ are in equilibrium.*

**Proof.** Consider an edge $e = (x, y) \in M$, and let $e' = (x, y')$. Since $M$ and $p$ are compatible, we have $p(x) + c_e = p(y)$ and $p(x) + c_{e'} \geq p(y')$. Subtracting these two inequalities to cancel $p(x)$, and substituting the values of $p$ and $c$, we get the desired inequality in the definition of equilibrium. ■

# Solved Exercises

## Solved Exercise 1

Suppose you are given a directed graph $G = (V, E)$, with a positive integer capacity $c_e$ on each edge $e$, a designated source $s \in V$, and a designated sink $t \in V$. You are also given an integer maximum $s$-$t$ flow in $G$, defined by a flow value $f_e$ on each edge $e$.

Now suppose we pick a specific edge $e \in E$ and increase its capacity by one unit. Show how to find a maximum flow in the resulting capacitated graph in time $O(m + n)$, where $m$ is the number of edges in $G$ and $n$ is the number of nodes.

***Solution*** The point here is that $O(m + n)$ is not enough time to compute a new maximum flow from scratch, so we need to figure out how to use the flow $f$ that we are given. Intuitively, even after we add 1 to the capacity of edge $e$, the flow $f$ can't be that far from maximum; after all, we haven't changed the network very much.

In fact, it's not hard to show that the maximum flow value can go up by at most 1.

**(7.68)** *Consider the flow network $G'$ obtained by adding 1 to the capacity of $e$. The value of the maximum flow in $G'$ is either $v(f)$ or $v(f) + 1$.*

**Proof.** The value of the maximum flow in $G'$ is at least $\nu(f)$, since $f$ is still a feasible flow in this network. It is also integer-valued. So it is enough to show that the maximum-flow value in $G'$ is at most $\nu(f) + 1$.

By the Max-Flow Min-Cut Theorem, there is some $s$-$t$ cut $(A, B)$ in the original flow network $G$ of capacity $\nu(f)$. Now we ask: What is the capacity of $(A, B)$ in the new flow network $G'$? All the edges crossing $(A, B)$ have the same capacity in $G'$ that they did in $G$, with the possible exception of $e$ (in case $e$ crosses $(A, B)$). But $c_e$ only increased by 1, and so the capacity of $(A, B)$ in the new flow network $G'$ is at most $\nu(f) + 1$. ■

Statement (7.68) suggests a natural algorithm. Starting with the feasible flow $f$ in $G'$, we try to find a single augmenting path from $s$ to $t$ in the residual graph $G'_f$. This takes time $O(m + n)$. Now one of two things will happen. Either we will fail to find an augmenting path, and in this case we know that $f$ is a maximum flow. Otherwise the augmentation succeeds, producing a flow $f'$ of value at least $\nu(f) + 1$. In this case, we know by (7.68) that $f'$ must be a maximum flow. So either way, we produce a maximum flow after a single augmenting path computation.

## Solved Exercise 2

You are helping the medical consulting firm Doctors Without Weekends set up the work schedules of doctors in a large hospital. They've got the regular daily schedules mainly worked out. Now, however, they need to deal with all the special cases and, in particular, make sure that they have at least one doctor covering each vacation day.

Here's how this works. There are $k$ vacation periods (e.g., the week of Christmas, the July 4th weekend, the Thanksgiving weekend, . . . ), each spanning several contiguous days. Let $D_j$ be the set of days included in the $j^{\text{th}}$ vacation period; we will refer to the union of all these days, $\cup_j D_j$, as the set of all *vacation days.*

There are $n$ doctors at the hospital, and doctor $i$ has a set of vacation days $S_i$ when he or she is available to work. (This may include certain days from a given vacation period but not others; so, for example, a doctor may be able to work the Friday, Saturday, or Sunday of Thanksgiving weekend, but not the Thursday.)

Give a polynomial-time algorithm that takes this information and determines whether it is possible to select a single doctor to work on each vacation day, subject to the following constraints.

- For a given parameter $c$, each doctor should be assigned to work at most $c$ vacation days total, and only days when he or she is available.
- For each vacation period $j$, each doctor should be assigned to work at most one of the days in the set $D_j$. (In other words, although a particular doctor may work on several vacation days over the course of a year, he or she should not be assigned to work two or more days of the Thanksgiving weekend, or two or more days of the July 4th weekend, etc.)

The algorithm should either return an assignment of doctors satisfying these constraints or report (correctly) that no such assignment exists.

***Solution*** This is a very natural setting in which to apply network flow, since at a high level we're trying to match one set (the doctors) with another set (the vacation days). The complication comes from the requirement that each doctor can work at most one day in each vacation period.

So to begin, let's see how we'd solve the problem without that requirement, in the simpler case where each doctor $i$ has a set $S_i$ of days when he or she can work, and each doctor should be scheduled for at most $c$ days total. The construction is pictured in Figure 7.23(a). We have a node $u_i$ representing each doctor attached to a node $v_\ell$ representing each day when he or she can

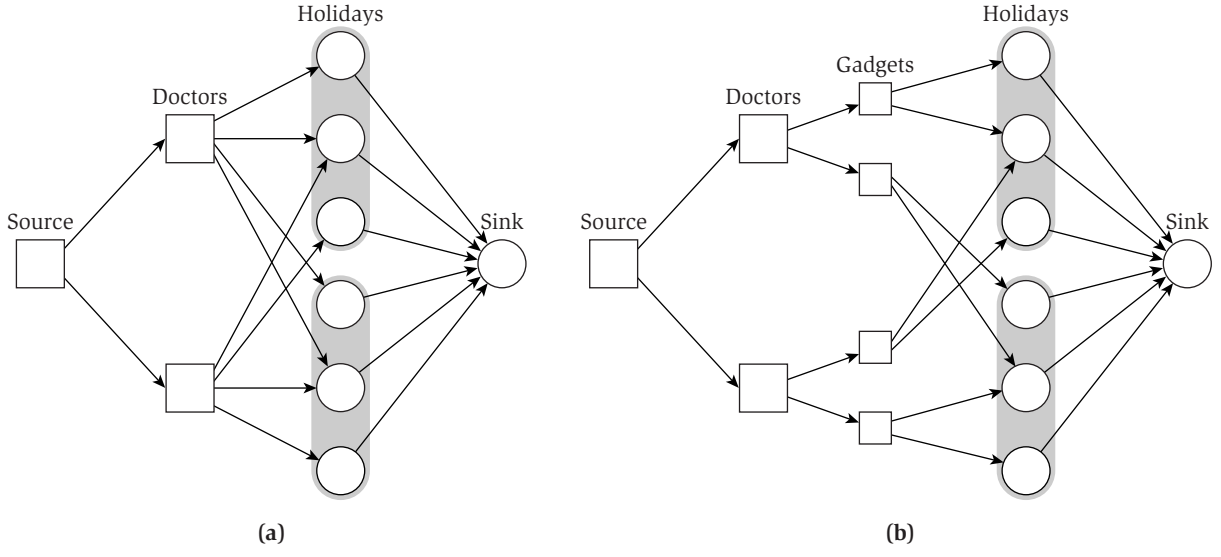


**Figure 7.23** (a) Doctors are assigned to holiday days without restricting how many days in one holiday a doctor can work. (b) The flow network is expanded with "gadgets" that prevent a doctor from working more than one day from each vacation period. The shaded sets correspond to the different vacation periods.

work; this edge has a capacity of 1. We attach a super-source $s$ to each doctor node $u_i$ by an edge of capacity $c$, and we attach each day node $v_\ell$ to a super-sink $t$ by an edge with upper and lower bounds of 1. This way, assigned days can "flow" through doctors to days when they can work, and the lower bounds on the edges from the days to the sink guarantee that each day is covered. Finally, suppose there are $d$ vacation days total; we put a demand of $+d$ on the sink and $-d$ on the source, and we look for a feasible circulation. (Recall that once we've introduced lower bounds on some edges, the algorithms in the text are phrased in terms of circulations with demands, not maximum flow.)

But now we have to handle the extra requirement, that each doctor can work at most one day from each vacation period. To do this, we take each pair $(i, j)$ consisting of a doctor $i$ and a vacation period $j$, and we add a "vacation gadget" as follows. We include a new node $w_{ij}$ with an incoming edge of capacity 1 from the doctor node $u_i$, and with outgoing edges of capacity 1 to each day in vacation period $j$ when doctor $i$ is available to work. This gadget serves to "choke off" the flow from $u_i$ into the days associated with vacation period $j$, so that at most one unit of flow can go to them collectively. The construction is pictured in Figure 7.23(b). As before, we put a demand of $+d$ on the sink and $-d$ on the source, and we look for a feasible circulation. The total running time is the time to construct the graph, which is $O(nd)$, plus the time to check for a single feasible circulation in this graph.

The correctness of the algorithm is a consequence of the following claim.

**(7.69)** *There is a way to assign doctors to vacation days in a way that respects all constraints if and only if there is a feasible circulation in the flow network we have constructed.*

**Proof.** First, if there is a way to assign doctors to vacation days in a way that respects all constraints, then we can construct the following circulation. If doctor $i$ works on day $\ell$ of vacation period $j$, then we send one unit of flow along the path $s$, $u_i$, $w_{ij}$, $v_\ell$, $t$; we do this for all such $(i, \ell)$ pairs. Since the assignment of doctors satisfied all the constraints, the resulting circulation respects all capacities; and it sends $d$ units of flow out of $s$ and into $t$, so it meets the demands.

Conversely, suppose there is a feasible circulation. For this direction of the proof, we will show how to use the circulation to construct a schedule for all the doctors. First, by (7.52), there is a feasible circulation in which all flow values are integers. We now construct the following schedule: If the edge $(w_{ij}, v_\ell)$ carries a unit of flow, then we have doctor $i$ work on day $\ell$. Because of the capacities, the resulting schedule has each doctor work at most $c$ days, at most one in each vacation period, and each day is covered by one doctor.

■