7-2

(a) The flow has value 18. It is not a maximum flow.

(b) Define the set A to be s and the top node. The cut $(A, V - A)$ is a minimum cut and has capacity 21.

7-1

(a) Let A denote the set in the cut that includes node 's'. Then the following sets of nodes are minimum cut-sets for A:
{s}, {s,u,v},{s,v}

(b) The minimum cutset is ({s,v}, {u,t})

[1] ex700.306.334

7-2

(a) The flow has value 18. It is not a maximum flow.

(b) Define the set A to be s and the top node. The cut $(A, V - A)$ is a minimum cut and has capacity 21.

7-1

(a) Let A denote the set in the cut that includes node 's'. Then the following sets of nodes are minimum cut-sets for A:
{s}, {s,u,v},{s,v}. They all have capacity of 2.

(b) The minimum cutset is ({s,v}, {u,t})
The minimum capacity is 4

[1] ex700.306.334

7-3

(a) The value of this flow is 10. It is not a maximum flow.

(b) The minimum cut is $(\{s, a, b, c\}, \{d, t\})$. Its capacity is 11.

[1] ex84.475.557

7-4

This is is false. Consider a graph with nodes s, v, w, t, edges (s, v), (v, w), (w, t), capacities of 2 on (s, v) and (w, t), and a capacity of 1 on (v, w). Then the maximum flow has value 1, and does not saturate the edge out of s.

[1] ex948.608.156

7-5

This is false. Consider a graph with nodes s, v_1, v_2, v_3, w, t, edges (s, v_i) and (v_i, w) for each i, and an edge (w, t). There is a capacity of 4 on edge (w, t), and a capacity of 1 on all other edges. Then setting $A = \{s\}$ and $B = V - A$ gives a minimum cut, with capacity 3. But if we add one to every edge then this cut has capacity 6, more than the capacity of 5 on the cut with $B = \{t\}$ and $A = V - B$.

[1] ex820.292.535

7-6

We build the following bipartite graph $G = (V, E)$. V is partitioned into sets X and Y, with a node $x_i \in X$ representing switch i, and a node $y_j \in Y$ representing fixture j. (x_i, y_j) is an edge in E if and only if the line segment from x_i to y_j does not intersect any of the m walls in the floor plan. Thus, whether $(x_i, y_j) \in E$ can be determined initially by m segment-intersection tests; so G can be built in time $O(n^2 m)$.

Now, we test in $O(n^3)$ time whether G has a perfect matching, and declare the floor plan to be "ergonomic" if and only if G does have a perfect matching. Our answer is always correct, since a perfect matching in G is a pairing of the n switches and the n fixtures in such a way that each switch can see the fixture it is paired with, by the definition of the edge set E; conversely, such a pairing of switches and fixtures defines a perfect matching in G.

[1] ex527.636.149

8-1

(1a) Yes. One solution would be: *Interval Scheduling* can be solved in polynomial time, and so it can also be solved in polynomial time with access to a black box for *Vertex Cover*. (It need never call the black box.) Another solution would be: *Interval Scheduling* is in NP, and anything in NP can be reduced to *Vertex Cover*. A third solution would be: we've seen in the book the reductions *Interval Scheduling* $\leq_P$ *Independent Set* and *Independent Set* $\leq_P$ *Vertex Cover*, so the result follows by transitivity.

(1b) This is equivalent to whether P = NP. If P = NP, then *Independent Set* can be solved in polynomial time, and so *Independent Set* $\leq_P$ *Interval Scheduling*. Conversely, if *Independent Set* $\leq_P$ *Interval Scheduling*, then since *Interval Scheduling* can be solved in polynomial time, so could *Independent Set*. But *Independent Set* is NP-complete, so solving it in polynomial time would imply P = NP.

[1] ex370.181.361

8-2

The problem is in $\mathcal{NP}$ because we can exhibit a set of k customers, and in polynomial time is can be checked that no two bought any product in common.

We now show that *Independent Set* $\leq_P$ *Diverse Subset*. Given a graph G and a number k, we construct a customer for each node of G, and a product for each edge of G. We then build an array that says customer v bought product e if edge e is incident to node v. Finally, we ask whether this array has a diverse subset of size k.

We claim that this holds if and only if G has an independent set of size k. If there is a diverse subset of size k, then the corresponding set of nodes has the property that no two are incident to the same edge — so it is an independent set of size k. Conversely, if there is an independent set of size k, then the corresponding set of customers has the property that no two bought the same product, so it is diverse.

[1] ex640.690.659

8-3

The problem is in NP since, given a set of k counselors, we can check that they cover all the sports.

Suppose we had such an algorithm $\mathcal{A}$; here is how we would solve an instance of *Vertex Cover*. Given a graph $G = (V, E)$ and an integer k, we would define a sport S_e for each edge e, and a counselor C_v for each vertex v. C_v is qualified in sport S_e if and only if e has an endpoint equal to v.

Now, if there are k counselors that, together, are qualified in all sports, the corresponding vertices in G have the property that each edge has an end in at least one of them; so they define a vertex cover of size k. Conversely, if there is a vertex cover of size k, then this set of counselors has the property that each sport is contained in the list of qualifications of at least one of them.

Thus, G has a vertex cover of size at most k if and only if the instance of *Efficient Recruiting* that we create can be solved with at most k counselors. Moreover, the instance of *Efficient Recruiting* has size polynomial in the size of G. Thus, if we could determine the answer to the *Efficient Recruiting* instance in polynomial time, we could also solve the instance of *Vertex Cover* in polynomial time.

[1] ex195.705.667

8-4

(a) The general *Resource Reservation* problem can be restated as follows. We have a set of m resources, and n processes, each of which requests a subset of the resources. The problem is to decide if there is a set of k processes whose requested resource sets are disjoint.

We first show the problem is in NP. To see this, notice that if we are given a set of k processes, we can check in polynomial time that no resource is requested by more than one of them.

To prove that the *Resource Reservation* problem in NP-complete we use the independent set problem, which is known to be NP-complete. We show that

Independent Set $\leq_P$ *Resource Reservation*

Given an instance of the independent set problem — specified by a graph G and a number k — we create an equivalent resource reservation problem. The resources are the edges, the processes correspond to the nodes of the graph, and the process corresponding to node v requests the resources corresponding to the edges incident on v. Note that this reduction takes polynomial time to compute. We need to show two things to see that the resource reservation problem we created is indeed equivalent.

First, if there are k processes whose requested resources are disjoint, then the k nodes corresponding to these processes form an independent set. This is true as any edge between these nodes would be a resource that they both request.

If there is an independent set of size k, then the k processes corresponding to these nodes form a set of k processes that request disjoint sets of resources.

(b) The case $k = 2$ can be solved by brute force: we just try all $O(n^2)$ pairs of processes, and we see whether any pair has disjoint resource requirements. This is a polynomial-time solution.

(c) This is just the Bipartite Matching Problem. We define a node for each person and each piece of equipment, and each process is an edge joining the person and the piece of equipment it needs. A set of processes with disjoint resource requirements is then a set of edges with disjoint ends — in other words, it is a matching in this bipartite graph. Hence we simply check whether there is a matching of size at least k.

(d) We observe that our reduction in (a) actually created an instance of *Resource Reservation* that had this special form. (Each edge/resource in the reduction was requested only by the two nodes/processes that formed its ends.) Thus, even this special case is NP-complete.

[1]ex588.290.312

8-5

Hitting Set is in NP: Given an instance of the problem, and a proposed set H, we can check in polynomial time whether H has size at most k, and whether some member of each set S_i belongs to H.

Hitting Set looks like a covering problem, since we are trying to choose at most k objects subject to some constraints. We show that *Vertex Cover* $\leq_P$ *Hitting Set*. Thus, we begin with an instance of *Vertex Cover*, specified by a graph $G = (V, E)$ and a number k. We must construct an equivalent instance of *Hitting Set*. In *Vertex Cover*, we are trying to choose at most k nodes to form a vertex cover. In *Hitting Set*, we are trying to choose at most k elements to form a hitting set. This suggests that we define the set A in the *Hitting Set* instance to be the V of nodes in the *Vertex Cover* instance. For each edge $e_i = (u_i, v_i) \in E$, we define a set $S_i = \{u_i, v_i\}$ in the *Hitting Set* instance.

Now we claim that there is a hitting set of size at most k for this instance, if and only if the original graph had a vertex cover of size at most k. For if we consider a hitting set H of size at most k as a subset of the nodes of G, we see that every set is "hit," and hence every edge has at least one end in H: H is a vertex cover of G. Conversely, if we consider a vertex cover C of G, and consider C as a subset of A, we see that each of the sets S_i is "hit" by C.

[1] ex635.897.959