(a) Prove this property carefully.

(b) Here is the new MST algorithm. The input is some undirected graph $G = (V, E)$ (in adjacency list format) with edge weights $\{w_e\}$.

```
sort the edges according to their weights
for each edge $e \in E$, in decreasing order of $w_e$:
   if $e$ is part of a cycle of $G$:
      $G = G - e$ (that is, remove $e$ from $G$)
return $G$
```

Prove that this algorithm is correct.

(c) On each iteration, the algorithm must check whether there is a cycle containing a specific edge e. Give a linear-time algorithm for this task, and justify its correctness.

(d) What is the overall time taken by this algorithm, in terms of $|E|$?

5.23. You are given a graph $G = (V, E)$ with positive edge weights, and a minimum spanning tree $T = (V, E')$ with respect to these weights; you may assume G and T are given as adjacency lists. Now suppose the weight of a particular edge $e \in E$ is modified from $w(e)$ to a new value $\hat{w}(e)$. You wish to quickly update the minimum spanning tree T to reflect this change, without recomputing the entire tree from scratch. There are four cases. In each case give a linear-time algorithm for updating the tree.

(a) $e \notin E'$ and $\hat{w}(e) > w(e)$.

(b) $e \notin E'$ and $\hat{w}(e) < w(e)$.

(c) $e \in E'$ and $\hat{w}(e) < w(e)$.

(d) $e \in E'$ and $\hat{w}(e) > w(e)$.

5.24. Sometimes we want light spanning trees with certain special properties. Here's an example.

Input: Undirected graph $G = (V, E)$; edge weights w_e; subset of vertices $U \subset V$
Output: The lightest spanning tree in which the nodes of U are leaves (there might be other leaves in this tree as well).

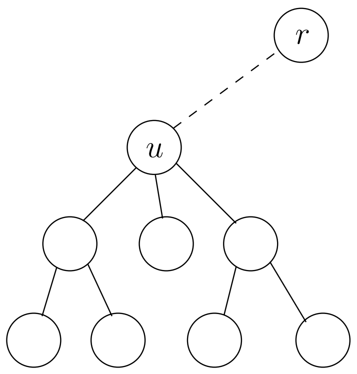
(The answer isn't necessarily a minimum spanning tree.)

Give an algorithm for this problem which runs in $O(|E| \log |V|)$ time. (*Hint:* When you remove nodes U from the optimal solution, what is left?)

5.25. A binary counter of unspecified length supports two operations: `increment` (which increases its value by one) and `reset` (which sets its value back to zero). Show that, starting from an initially zero counter, any sequence of n `increment` and `reset` operations takes time $O(n)$; that is, the amortized time per operation is $O(1)$.

5.26. Here's a problem that occurs in automatic program analysis. For a set of variables $x_1, \ldots, x_n$, you are given some *equality* constraints, of the form

Figure 6.11 $I(u)$ is the size of the largest independent set of the subtree rooted at u. Two cases: either u is in this independent set, or it isn't.



Exercises

6.1. A *contiguous subsequence* of a list S is a subsequence made up of consecutive elements of S. For instance, if S is

$$5, 15, -30, 10, -5, 40, 10,$$

then $15, -30, 10$ is a contiguous subsequence but $5, 15, 40$ is not. Give a linear-time algorithm for the following task:

Input: A list of numbers, $a_1, a_2, \ldots, a_n$.
Output: The contiguous subsequence of maximum sum (a subsequence of length zero has sum zero).

For the preceding example, the answer would be $10, -5, 40, 10$, with a sum of 55.

(*Hint:* For each $j \in \{1, 2, \ldots, n\}$, consider contiguous subsequences ending exactly at position j.)

6.2. You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \cdots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance a_n), which is your destination.

You'd ideally like to travel 200 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel x miles during a day, the *penalty* for that day is $(200 - x)^2$. You want to plan your trip so as to minimize the total penalty—that is, the sum, over all travel days, of the daily penalties. Give an efficient algorithm that determines the optimal sequence of hotels at which to stop.