CS140: Algorithms

Fall 2001

Solutions to Assignment 6

1. An *independent set* of a graph G is a subset W of the vertices of G such that no two vertices in W are connected by an edge in G. Finding a largest independent set of an arbitrary graph is NP-hard. The problem is solvable in polynomial time, however, if the input is restricted to be a forest.

   (a) Let F be a forest and let v be a leaf in F. Prove that v is in some largest independent set of F.

   Let W be a largest independent set of F and suppose $v \notin F$. Let u be the node adjacent to v in F. If u is not in W then $W + \{v\}$ is a larger independent set of F than W, which contradicts our assumption. So u must be an element of W. But then $W - \{u\} + \{v\}$ is also a largest independent set of F and it contains u.

   (b) Suppose I give you a black box that will find a largest independent set of a forest with $n - 1$ or fewer nodes. Explain how to use this box to build an algorithm for finding independent sets in forests with n or fewer nodes. Be sure to describe any base cases. (How do you handle isolated nodes?)

   If F has no edges then the vertices of F are a largest independent set of F. If F has an edge then it has a leaf, call it v. Let u be the node adjacent to v in F. Compute a largest independent set W' of the forest $F - \{u, v\}$. Then $W' + \{v\}$ is a largest independent set of F.

   (c) How would you implement this algorithm? Carefully describe the data structures you'll use. You may specify the data structure used to represent the input. Analyze the running time of the algorithm implemented with your data structures.

   (Left to the student.)

2. In the *weighted independent set* problem, the input is a graph G in which each vertex is assigned a positive weight. The *weight* of an independent set W is the sum of the weights of the vertices in W. The objective is to find an independent set with largest weight. This problem is NP-hard but can be solved in polynomial time if we restrict the input to be a weighted tree.

   (a) Give an example of a weighted tree T for which your algorithm from problem 2 fails to find a maximum weight independent set.

   (Left to the student.)

(b) Give an example of a tree T rooted at some node r such that the union of the maximum weight independent sets of the children of r is a maximum weight independent set T.

(Left to the student.)

(c) Give an example of a tree T rooted at some node r such that the union of the maximum weight independent sets of the children of r is *not* a maximum weight independent set T. Give a maximum weight independent set of this tree.

(Left to the student.)

(d) What information would you like about the subtrees of a rooted tree in order to determine the weight of a maximum weight independent set of the tree? (You should be able to do this with two functions on the nodes of the tree.)

I(v): weight of a maximum weight independent set of the tree rooted at v that includes v. E(v): weight of a maximum weight independent set of the tree rooted at v that excludes v.

(e) Design an algorithm that computes the functions you defined in the previous part.

If v is a leaf then $I(v) = w(v)$ and $E(v) = 0$. If v is an interior node then $I(v) = w(v) + \sum_{\text{children } u \text{ of } v} E(u)$ and $E(v) = \sum_{\text{children } u \text{ of } v} max(E(u), I(u))$.

(f) Show how to use this algorithm to determine the weight of a maximum weight independent set of a tree.

Root the tree at an arbitrary node r. Compute $E(v)$ and $I(v)$ at all nodes starting at the leaves and working up to the root. Then the weight of a maximum weight independent set of the tree is $max(I(r), E(r))$.

(g) Show how to modify your algorithms to compute a maximum weight independent set of an input tree.

At each node v we include the set of vertices forming the independent sets corresponding to $I(v)$ and $E(v)$.

(h) Briefly describe implementation details. Note that an $\Theta(n)$ algorithm is possible.

Left to the student.