Give an efficient greedy algorithm that finds an optimal vertex cover of a tree in linear time.

In a vertex cover we need to have at least one vertex for each edge.
Every tree has at least two leaves, meaning that there is always an edge which is adjacent to a leaf. Which vertex can we never go wrong picking? The non-leaf, since it is the only one which can also cover other edges!
After trimming off the covered edges, we have a smaller tree. We can repeat the process until the tree as 0 or 1 edges. When the tree consists only of an isolated edge, pick either vertex.
All leaves can be identified and trimmed in $O(n)$ time during a DFS.
Dealing with $NP$-complete Problems
Option 1: Algorithm fast in the Average case

Examples are Branch-and-bound for the Traveling Salesman Problem, backtracking algorithms, etc.
Option 2: Heuristics

Heuristics are rules of thumb; fast methods to find a solution with no requirement that it be the best one. Note that the theory of $NP$-completeness does not stipulate that it is hard to get close to the answer, only that it is hard to get the optimal answer. Often, we can prove performance bounds on heuristics, that the resulting answer is within $C$ times that of the optimal one.
Approximating Vertex Cover

As we have seen, finding the minimum vertex cover is $NP$-complete. However, a very simple strategy (heuristic) can get us a cover at most twice that of the optimal.

While the graph has edges

- pick an arbitrary edge $v, u$
- add both $u$ and $v$ to the cover
- delete all edges incident on either $u$ and $v$

If the graph is represented by an adjacency list this can be implemented in $O(m + n)$ time.
This heuristic must always produce cover, since an edge is only deleted when it is adjacent to a cover vertex.
Further, any cover uses at least half as many vertices as the greedy cover.

Why? Delete all edges from the graph except the edges we selected.
No two of these edges share a vertex. Therefore, any cover of just these edges must include one vertex per edge, or half the
greedy cover!
Things to Notice

- Although the heuristic is simple, it is not stupid. Many other seemingly smarter ones can give a far worse performance in the worst case.

Example: Pick one of the two vertices instead of both (after all, the middle edge is already covered) The optimal cover is one vertex, the greedy heuristic is two vertices, while the new/bad heuristic can be as bad as $n - 1$. 
• Proving a lower bound on the optimal solution is the key to getting an approximation result.

• Making a heuristic more complicated does not necessarily make it better. It just makes it more difficult to analyze.

• A post-processing clean-up step (delete any unnecessary vertex) can only improve things in practice, but might not
help the bound.
The Euclidean Traveling Salesman

In the traditional version of TSP - a salesman wants to plan a drive to visit all his customers exactly once and get back home.

Euclidean geometry satisfies the triangle inequality, $d(u, w) \leq d(u, v) + d(v, w)$.

TSP remains hard even when the distances are Euclidean distances in the plane.
Note that the cost of airfares is an example of a distance function which violates the triangle inequality. However, we can approximate the optimal Euclidean TSP tour using minimum spanning trees. **Claim:** the cost of a MST is a lower bound on the cost of a TSP tour.

**Why?** Deleting any edge from a TSP tour leaves a path, which is a tree of weight at least that of the MST! If we were allowed to visit cities more than once, doing a depth-first traversal of a MST, and then walking out the tour specified is at most twice the cost of MST. Why? We will be using each edge exactly twice.
Every edge is used exactly twice in the DFS tour: 1. However, how can we avoid revisiting cities? We can take a shortest path to the next unvisited vertex. The improved tour is $1 - 2 - 3 - 5 - 8 - 9 - 6 - 4 - 7 - 10 - 11 - 1$. 
Because we replaced a chain of edges by the edge, the triangle inequality ensures the tour only gets shorter. Thus this is still within twice optimal!