CSC373— Algorithm Design, Analysis, and Complexity — Spring 2016

Solution Sketches for Tutorial Exercise 1: Greedy Algorithms

---

1. Truck Driver's Problem. Prove that no optimal solution for the Truck Driver's Problem (see lecture notes on Greedy Algorithms, pp 29-32) backtracks to a gas station that it has already been passed.

That is, suppose $\mathcal{O}$ is any optimal solution. Assume it uses exactly q stops at gas stations along the way (the minimum number possible). Suppose these stops are at the distances $\{f_k\}_{k=1}^{q}$ down the road. Prove that $f_{k+1} > f_k$ for all $k = 1, \ldots, q-1$.

2. Certificate for Minimum Max-Lateness. Consider the problem of minimizing the maximum lateness for a set of jobs, as discussed on pp.17-22 of the lecture notes. Given a general instance of this problem, and the corresponding optimal solution $\mathcal{O}$ that you have computed (say, with the greedy algorithm described in the lecture notes), imagine that you wish to show your boss (who is intelligent but not technically inclined) that there is no better solution than your optimal solution $\mathcal{O}$. That is, given the input data for the problem, namely $\{(t_j, d_j)\}_{j=1}^{n}$ where t_j indicates the processing time required for the j^{th} job and d_j is that job's deadline, derive a simple formula to convince your boss that no solution can have a maximum lateness that is less than that of your optimal solution. Such a formula is called a **certificate** for the optimality of your solution.

Hint #1: If the maximum lateness for your optimal solution $\mathcal{O}$ is bigger than zero, how can you prove to your boss that no solution can have a maximum lateness of zero?

Hint #2: Make use of your optimal solution $\mathcal{O}$ to select your certificate.

3. Greedy Vertex Cover for Trees. Professor Jot has proposed the following algorithm (see the next page) to solve the minimum vertex cover problem, which is itself described below. Decide whether or not Prof. Jot's algorithm is correct when the input graph is a tree. If you believe the algorithm is correct, prove it using an exchange argument (i.e., show that at each stage it is consistent with some optimal solution). Otherwise show a counter-example.

Terminology. An undirected graph $G = (V, E)$, consisting of vertices $v \in V$ and edges $(u, v) \in E$ (with $u \neq v$) is a **tree** if G is connected (i.e., there is a path in G from each vertex $u \in V$ to every other vertex $v \in V$) and there are no cycles in G (i.e., there is no path from a vertex u back to u without retraversing an edge in the opposite direction). A vertex v with at most one edge $(u, v) \in E$ is called a **leaf**. Also, an undirected graph G with no cycles is called a **forest** (in which each connected set is a tree).

A useful property is that, if $G = (V, E)$ is a tree then the number of edges $|E|$ satisfies $|E| = |V| - 1$.

A **vertex cover** is a subset $C \subseteq V$ such that for every edge $(u, v) \in E$ either $u \in C$ or $v \in C$ (or both). That is, the vertices in C "cover" (at least one end of) every edge in E.

We wish to find a **minimum-sized vertex cover** C. For example, if G is a simple chain, then the minimum size vertex cover can be formed by using every other vertex along the chain (skipping the first). Or if G is star shaped, with one center vertex connected to every other vertex, and these other vertices are all leaves, then the minimum vertex cover consists of just that center vertex.

[S] = MinTreeVertexCover(V, E)
% Return a vertex cover of minimum possible size for
% the input tree T = (V, E).
% Precondition: T is a valid undirected tree

Initialize $F \leftarrow (V, E)$ and $S \leftarrow \{\ \}$.
While the forest F is not empty:
    Find a leaf vertex u in F.
    If there is no edge in F which includes u as an endpoint:
        Delete u from F.
    Else:
        Let (u, v) be such an edge in F.
        Set $S \leftarrow S \bigcup \{v\}$.
        Remove u, v, (u, v), and all other edges which include v from F.
    End if
End while
return S