(a) We'll say a set of advertisements is "valid" if it covers all paths in $\{P_i\}$. First, *Strategic Advertising* (SA) is in NP: Given a set of k nodes, we can check in $O(kn)$ time (or better) whether at least one of them lies on a path P_i, and so we can check whether it is a valid set of advertisements in time $O(knt)$.

We now show that *Vertex Cover* $\leq_P$ SA. Given an undirected graph $G = (V, E)$ and a number k, produce a directed graph $G' = (V, E')$ by arbitrarily directing each edge of G. Define a path P_i for each edge in E'. This construction involves one pass over the edges, and so takes polynomial time to compute. We now claim that G' has a valid set of at most k advertisements if and only if G has a vertex cover of size at most k. For suppose G' does have such a valid set U; since it meets at least one end of each edge, it is a vertex cover for G. Conversely, suppose G has a vertex cover T of size at most k; then, this set T meets each path in $\{P_i\}$ and so it is a valid set of advertisements.

(b) We construct the algorithm by induction on k. If $k = 1$, we simply check whether there is any node that lies on all paths. Otherwise, we ask the fast algorithm $\mathcal{S}$ whether there is a valid set of advertisements of size at most k. If it says "no," we simply report this. If it says "yes", we perform the following test for each node v: we delete v and all paths through it, and ask $\mathcal{S}$ whether, on this new input, there is a valid set of advertisements of size at most $k - 1$. We claim that there is at least one node v where this test will succeed. For consider any valid set U of at most k advertisements (we know one exists since $\mathcal{S}$ said "yes"): The test will succeed on any $v \in U$, since $U - \{v\}$ is a valid set of at most $k - 1$ advertisements on the new input.

Once we identify such a node, we add it to a set T that we maintain. We are now dealing with an input that has a valid set of at most $k - 1$ advertisements, and so our algorithm will finish the construction of T correctly by induction. The running time of the algorithm involves $O(n + t)$ operations and calls to $\mathcal{S}$ for each fixed value of k, for a total of $O(n^2 + nt)$ operations.

[1] ex685.1.698