Problem 1. We know that it is possible to find a cycle in a graph with complexity of $O(n + m)$. In this problem, you have to describe an algorithm to determine whether or not a given undirected graph $G = (V, E)$ contains a cycle. Your algorithm should run in $O(n)$ time only, independent of $m$. Hint: Note that your algorithm should decide if there is a cycle or not, but it doesn’t need to return the cycle.

(a) Provide pseudocode for your algorithm.

(b) Justify your algorithm’s running time.

Problem 2. Consider graphs where each vertex $v$ is labeled with a real number $x(v)$. The $x(v)$ are all distinct (no two vertices share the same value). A node $v$ is a local minimum if $x(v) < x(u)$ for all of $v$’s neighbors $u$. In this problem, we aim to design efficient algorithms for finding a local minima.

The labels are initially unknown to you. You can obtain the value $x(v)$ of a vertex $v$ by calling a subroutine $\text{query}(v)$. We would like to design algorithms that make few calls to this subroutine. Suppose that $G$ is a complete $n$-node binary tree, where $n = 2^d - 1$ for some $d$. Design an algorithm that finds a local minima of $G$ using $O(\log n)$ calls to the subroutine $\text{query}$.

Problem 3. Let $G = (V, E)$ be an undirected graph with $n$ nodes. A subset of the nodes is called an independent set if no two of them are joined by an edge. Finding large independent sets is difficult in general; but here we’ll see that it can be done efficiently if the graph is simple enough.

Call a graph $G = (V, E)$ a path if its nodes can be written as $v_1, v_2, ..., v_n$, with an edge between $v_i$ and $v_{i+1}$, for $i \in \{1, 2, \ldots, n - 1\}$. With each node $v_i$, we associate a positive integer weight $w_i$. The problem we want to solve is the following problem: Find an independent set in a path $G$ whose total weight is as large as possible.

(a) Give an example to show that the algorithm depicted in Figure 1 does not always find an independent set of maximum total weight.

(b) Give an example to show that the algorithm depicted in Figure 2 also does not always find an independent set of maximum total weight.

(c) Give an algorithm that takes an $n$-node path $G$ with weights and returns the value of the independent set of maximum total weight in running time $O(n)$. You must argue about why this running time is correct.
You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains $n$ numerical values - so there are $2n$ values total - and you may assume that no two values are the same. You’d like to determine the median of this set of $2n$ values, which we will define here to be the $n$th smallest value.

However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value $k$ to one of the two databases, and the chosen database will return the $k$th smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

(a) Give an algorithm that finds the median value using at most $O(\log n)$ queries.

(b) Explain why your algorithm solves the problem.