Advanced Algorithms. Assignment 3

Exercise 5.
A river together with all its tributaries forms a tree, in an obvious sense: The nodes of the tree are the confluences, and water flows from the leaves (sources) towards the root (the river mouth).
Now suppose that some pollutant is poured into the water at some place. Then the water is polluted all the way down, and the pollutant can be detected there in the water. We wish to figure out where the accident or offense happened.

In a more abstract language: We are given a directed tree, where all nodes on the path P from some unknown node p to the root are contaminated. We can pick arbitrary nodes and test whether they are contaminated or not. The goal is to find p by an algorithm that tests a minmum number of nodes. Let k be the number of edges from p to the root. Note carefully that p is unknown, hence k is unknown, too.
For simplicity, we consider the case of a binary tree, that is, every non-leaf node has exactly two children. The notation $O(1)$ means an unspecified constant.

5.1. Give a deterministic algorithm that needs at most $2k + O(1)$ tests in the worst case. This is quite trivial and serves only as a benchmark for comparison to the next result.

5.2. Here is the actual exercise: Give a randomized algorithm that only needs an expected number of at most $1.5k + O(1)$ tests. Give an accurate proof of this expected test number for your algorithm.

Optional: We do not claim that the above test numbers are already the best possible results. Perhaps you can beat them?

See the reverrse page.

Exercise 6.
The problem of finding a large clique in a graph is notoriously difficult. It is even NP-hard to remotely approximate it. However, with randomization, parameterization, and brute force we can at least achieve a modest result and detect a small piece of an existing large clique in an affordable time. In detail:

Suppose that a given graph with n nodes contains some clique with cn nodes, where $c < 1$ is some fixed positive fraction. The number c is known in advance, but not the clique itself. Furthermore let k be some prescribed integer (which may be much smaller than n).

Propose a randomized algorithm that finds a clique with k nodes. Its running time should be polynomial in n but it may be exponential in k. Your algorithm may be of Monte Carlo or Las Vegas type. In either case, give rigorous proofs of all properties you claim.

Exercise 7.
Some set of k jobs must be done within a period of n days, where $k < n$. Every job needs exactly one day, and at most one job can be done per day. Moreover, every job can be done only at certain days (because it needs some resource that is available only at those days). For every job we know the list of feasible days. The order in which the k jobs are executed is arbitrary. However there is yet another restriction: It is not possible to execute jobs at two consecutive days, that is, there must be a break of at least one day between any jobs.

Give an algorithm that outputs a schedule for the k jobs (or outputs that no solution exists). Its running time must be polynomial in n, but not necessarily in k.

Optional: What if we drop the requirement of having breaks?