Advanced Algorithms. Assignment 3

Exercise 5.
A river together with all its tributaries forms a "tree", in the mathematical sense: The nodes of the tree are the confluences, and water flows from the leaves (sources) towards the root (the river mouth).
Suppose that some pollutant has been poured into the water at some place. Then the water is polluted all the way down, and the pollutant can be detected there by testing the water. We wish to figure out where the accident or offense happened. In a more abstract language, this yields the following problem:

We are given a directed binary tree, that is, every non-leaf node has exactly two children. 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 testing some nodes. We aim at a minmum number of tests.
Let k be the number of edges from p to the root. Note carefully that k is unknown.

5.1. Give a deterministic algorithm that needs at most $2k + O(1)$ tests in the worst case. (This is only a warm-up and a benchmark for comparison to the next results.)

5.2. Now 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, making proper use of proababilities and random variables.

5.3. The $1.5k + O(1)$ result is not yet optimal. Finally, give a randomized algorithm that further improves this expected number. – This part can be more sketchy. It suffices to give an idea and to argue why it beats the previous result. If you are ambitious: Can you even get close to k expected tests?

Find Exercise 6 on the reverse page.

Exercise 6.
Suppose that we want to store n different items in a warehouse. There, a row of n shelves indexed by the integers $1, \ldots, n$ from left to right is located along a wall. Every item shall be stored in exactly one shelf, but we can decide the order (permutation) of items. It would be desirable to arrange the items in such a manner that items that are typically ordered together are close to each other in the shelves, such that the workers that pick up the ordered[1] items have only short walks. More specifically, we have representative data consisting of m subsets of items that have been requested in the past. The quantities of items are not relevant for the problem, only the requested subsets S matter. The length of the walk along the wall needed to fetch the items in S equals $r - l$, whare r and l is the index of the rightmost and leftmost shelf, respectively, that contains an item from S. We wish to minimize the average length of the walk. This is equivalent to minimizing the sum of the differences $r - l$, for all requested sets S. After this explanation of the background we formulate the problem now in a more abstract way, for the sake of clarity.

We are given a set V of n elements, and m subsets $S_i \subset V$, $i = 1, \ldots, m$. (It is not assumed that the S_i are all different.) A linear ordering of V is a one-to-one mapping π of V onto the set of integers $\{1, \ldots, n\}$. That is, every $\pi(v)$ is some integer from 1 to n, and all $\pi(v)$, $v \in V$, are distinct. For a set $S \subset V$ and an ordering π of V, let $l_\pi(S)$ and $r_\pi(S)$ denote, respectively, the smallest and largest value $\pi(v)$ among all $v \in S$. The problem is to find an ordering π that minimizes $\sum_{i=1}^{m}(r_\pi(S_i) - l_\pi(S_i))$.
The problem is NP-hard, a fact that we will not prove here. Next, a naive $O^*(n!)$-time algorithm would simply examine all $n!$ permutations, compute the sums, and compare them.
Your task: Give a deterministic algorithm that needs only $O^*(2^n)$ time.
Comments and advice: This time bound is a significant improvement in practice, as it allows to plan bigger warehouses in reasonable time. As the time bound already suggests, you should use the technique of dynamic programming on subsets. For $k = 1, 2, 3, \ldots$, fill the first k shelves in all possible ways that may lead to an overall optimal solution. The details of the step from k to $k+1$ might appear a bit tricky, but they are not extremely complicated either. The fact that we want to miminize a sum should help a lot.

[1] Note that the word "order" appears in two different meanings!