Advanced Algorithms. Assignment 3

Exercise 5.
Suppose that we have a source of random bits, that is, we can generate a random number 0 or 1, each with probability 1/2, and we can do this independently and as often as we want. You can assume that generating each random bit costs $O(1)$ time.

(a) Now we want to use these random bits to generate a random integer in the range $\{1, \ldots, n\}$, where every outcome has probability $1/n$. Give an algorithm for this task. Note that it must work for every given positive integer n (not only, e.g., for powers of 2, which is a simple case). What is the expected time that you need? It should be a "small" function of n, and your analysis should be rigorous, without handwaving arguments.

(b) As a more complex task, we want to generate a random vector of m bits, where exactly k bits are 1 and $m - k$ bits are 0. That means, each of the $\binom{m}{k}$ such vectors must be produced with the same probability $1/\binom{m}{k}$. But the expected time must be polynomial in m. Therefore you cannot simply list all these vectors and apply your method from (a) with $n := \binom{m}{k}$, as this would take exponential time. Give instead a fast algorithm (there are several possible approaches), explain why it correctly solves the problem, and again, do a rigorous time analysis.

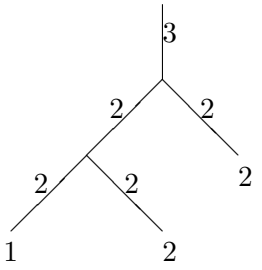
The assignments end with some garbage:

Exercise 6.
A modern waste collection system used in some cities consists of pipes that are connected like the edges of a tree[1]. The root of the tree contains a waste collection station, every leaf contains an inlet where people can dispense their garbage, and every inner node contains valves to the incident pipes. All valves can be opened and closed individually.
During an emptying process, the waste is transported from the leaves to the root by air suction, through the open pipes. However, every pipe has a given capacity, and the total amount of waste going through the pipe must not exceed its capacity. An inlet can only be emptied completely (not partially), or it is not emptied now at all (and has to wait for a future process). Due to the limited pipe capacities, not all inlets can be emptied at once. The

[1]The graph is meant here and in the following, not a real tree.

system must carefully select a subset of inlets and open the pipes on the ways from these inlets to the station. The amount of waste in every inlet is measured by a sensor. It is desirable to collect as much waste as possible in one process.

Actually you can skip the entire text until **this point**. Now we describe the resulting problem formally. We are given a rooted binary tree of n nodes, with edge capacities c_e and with weights x_v at the leaves v. Let L_e denote the set of all leaves in the subtree rooted at edge e (the "subtree below e", so to speak). We want to find a subset S of leaves such that $\sum_{v \in S} x_v$ is maximized, but $\sum_{v \in S \cap L_e} x_v \leq c_e$ holds for every edge e. It is assumed that all c_e and x_v are non-negative integers; let k be their maximum.



For example, in the tree displayed here, an optimal solution is to choose the first and third leaf and thus collect an amount of $1 + 2 = 3$. If the second leaf were chosen, it would block the two others.

Develop an algorithm for this problem, using dynamic programming on trees. It should run in $O(nk^2)$ time. Of course, you should also explain your solution.

Some advice: At first glance this looks like a flow problem, but you may recognize why a maximum-flow algorithm is not applicable here. Next, it may take some time to construct a dynamic programming "rule" that works. (Do not give up early.) Note that we have at most four choices at each node: take the waste from the left edge, or from the right edge, or from both, or from none. Think carefully: which information must be propagated upwards in the tree, in order not to miss an optimal solution in the end? You don't have to find a "nice" formula; a clear verbal desscription is also fine. Check whether your algorithm really yields the optimal result on small examples like the one above.