Advanced Algorithms. Assignment 2

Exercise 3.
Some network is modelled as an undirected and connected graph $G = (V, E)$. Every edge has a given positive weight that models the strength of the link. Some subset $I \subset V$ of nodes is highlighted: The nodes in I are important, while all other nodes merely serve as intermediate auxiliary nodes.
In order to see where the network is most vulnerable, we want to determine a set $F \subset E$ of edges such that:
(a) removal of F disconnects the important nodes, that is, not all nodes of I are in the same connected component of $G = (V, E \setminus F)$;
(b) the sum of weights of the edges in F is minimal, among all edge sets that satisfy (a).

Let $n = |V|$, $k = |I|$, $m = |E|$. Give an algorithm that solves this problem. Aim at a time bound (as a function of n, k, m) that is as low as possible.
Of course, you may use the knowledge about flows and cuts. Do not forget to argue why your algorithm is correct (that is, cannot miss an optimal solution F). Pay special attention to the fact that G is undirected.

Exercise 4.
The floor of a storage hall is partitioned like a chess board into black and white unit squares, say of 1×1 meters. Many of the squares are occupied by heavy objects that cannot be easily moved. That is, only a certain set F of the squares are free. Nothing special is assumed about the shape of F.
Now a number of boxes of size 1×2 meters shall be stored. Every box must be placed exactly on two neighbored free squares, but the boxes may be rotated (stand in N-S or W-E direction). The problem is to place as many boxes as possible on the given set F of squares.

4.1. Solve this problem efficiently, by reducing it to Maximum Flow. Make sure to give a time bound and to prove that your approach guarantees an optimal number of boxes.

4.2. Suppose that you have computed an optimal solution, and later on, one more square becomes free. That is, one square is added to F somewhere. This provides a chance to place more boxes. In order to decide correctly whether now a better solution exists or not, you may, of course, re-compute an optimal solution from scratch. But can you do that more efficiently?