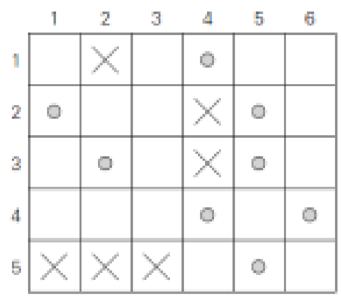
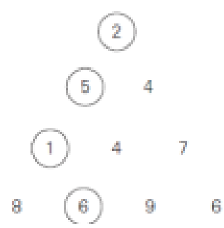
1. (20 points) How would you modify the dynamic programming algorithm for the coin-collecting problem if some cells on the board are inaccessible for the robot? Apply your algorithm to the board below, where the inaccessible cells are shown by X's. How many optimal paths are there for this board?



2. (20 points) Some positive integers are arranged in a triangle with n numbers in its base like the one shown in the figure below for $n = 4$. The problem is to find the smallest sum in a descent from the triangle apex to its base through a sequence of adjacent numbers (shown in the figure by the circles). Design a dynamic programming algorithm for this problem and indicate its time efficiency. Your solution should be general and not just for this example. It should be applicable to any triangle of any size.



3. (20 points) Design an efficient algorithm for computing the binomial coefficient $C(n, k)$ that uses no multiplications. What are the time and space efficiencies of your algorithm?

4. (20 points) Design a dynamic programming algorithm for the version of the knapsack problem in which there are unlimited quantities of copies for each of the n item kinds given. Indicate the time efficiency of the algorithm.

5. (20 points) How would you construct an optimal binary search tree for a set of n keys if all the keys are equally likely to be searched for? What will be the average number of comparisons in a successful search in such a tree if $n = 2k$ (n is an even number)?