1. (Pebbling a checkerboard) We are given a checkerboard which has 4 rows and n columns, and has an integer written in each square. We are also given a set of $2n$ pebbles, and we want to place some or all of these on the checkerboard (each pebble can be placed on exactly one square) so as to maximize the sum of the integers in the squares that are covered by pebbles. There is one constraint: for a placement of pebbles to be legal, no two of them can be on horizontally or vertically adjacent squares (diagonal adjacency is fine).

   (a) Determine the number of legal patterns that can occur in any column (in isolation, ignoring the pebbles in adjacent columns) and describe these patterns.

   Call two patterns compatible if they can be placed on adjacent columns to form a legal placement. Let us consider subproblems consisting of the first k columns $1 \leq k \leq n$. Each subproblem can be assigned a type, which is the pattern occurring in the last column.

   (b) Using the notions of compatibility and type, give an $O(n)$-time dynamic programming algorithm for computing an optimal placement.

2. (The garage sale problem) On a given Sunday morning, there are n garage sales going on, $g_1, g_2, \ldots, g_n$. For each garage sale g_j , you have an estimate of its value to you, v_j. For any two garage sales you have an estimate of the transportation cost d_{ij} of getting from g_i to g_j. You are also given the costs d_{0j} and d_{j0} of going between your home and each garage sale. You want to find a tour of a subset of the given garage sales, starting and ending at home, that maximizes your total benefit minus your total transportation costs.

   Give an algorithm that solves this problem in time $O(n^2 2^n)$.

3. Let T be a weighted tree and for each node u, let $w(u)$ be the weight of u. Recall that an independent set is a subset U of V such that no edge exists in between any two nodes in U. Give a polynomial time algorithm to compute the weight of largest weight Independent Set in T.

4. Given an array of prices, we say that the index range $[i \ldots j]$ defines a **sideways trend of percent** p if no two prices in that range differ by more than p percent. That is, for any two indices, s and t, lying in that range, neither is more than p percent larger than the other:

   $$\text{prices}[s] \leq (1 + p/100)\text{prices}[t]$$

   Design an efficient algorithm to find the longest sideways trend in the given data, i.e., the range of indices $[i \ldots j]$ defining a sideways trend of percent p with $j - i$ as large as possible.

5. Let G be directed weighted graph and s, t be two vertices of G. Design an efficient algorithm for finding a shortest path from s to t that uses exactly k vertices, where k is an integer.

6. Consider the 3-PARTITION problem. Given a set of integers $\{a_1, \ldots, a_n\}$, we want to find if it is possible to partition $\{1, \ldots, n\}$ into 3 disjoint sets I, J and K such that the following holds:

   $$\sum_{i \in I} a_i = \sum_{j \in J} a_j = \sum_{k \in K} a_k = \frac{\sum_{i=1}^{n} a_i}{3}$$

Design an efficient algorithm for solving this problem that runs in time polynomial in n and $\sum_{i=1}^{n} a_i$.

7. Given a schedule containing the arrival and departure times of trains at a station, design an efficient algorithm to find the minimum number of platforms required to avoid delays in any train's arrival.

8. Given a list of strings where no string is a substring of another, design an efficient algorithm to find a shortest string that contains each string in the list as a substring.