Tutorial-4
IIIT Delhi
Instructor: Debarka Sengupta

Problem 1.

Suppose you are managing the construction of billboards on the Stephen Daedalus Memorial Highway, a heavily travelled stretch of road that runs west-east for M miles. The possible sites for billboards are given by numbers $x_1, x_2, \ldots, x_n$, each in the interval [0, M] (specifying their position along the highway, measured in miles from its western end). If you place a billboard at location x_i, you receive revenue of $r_i > 0$. Regulations imposed by the county's Highway Department require that no two of the billboards be within less than or equal to 5 miles of each other. You'd like to place billboards at a subset of the sites so as to maximize your total revenue, subject to this restriction.

Example. Suppose M = 20, n = 4,
{x 1 , x 2 , x 3 , x 4 } = {6, 7, 12, 14},
and
{r 1 , r 2 , r 3 , r 4 } = {5, 6, 5, 1}.
Then the optimal solution would be to place billboards at x 1 and x 3 , for total revenue of 10.
Give an algorithm that takes an instance of this problem as input and returns the maximum total revenue that can be obtained from any valid subset of sites. The running time of the algorithm should be polynomial in n.

Problem 2.

The problem of searching for cycles in graphs arises naturally in financial trading applications. Consider a firm that trades shares in n different companies. For each pair i = j, they maintain a trade ratio r_{ij}, meaning that one share of i trades for r_{ij} shares of j. Here we allow the rate r to be fractional; that is, $r_{ij} = 2/3$ means that you can trade three shares of i to get two shares of j. A trading cycle for a sequence of shares $i_1, i_2, \ldots, i_k$ consists of successively trading shares in company i_1 for shares in company i_2, then shares in company i_2 for shares i_3, and so on, finally trading shares in i_k back to shares in company i_1. After such a sequence of trades, one ends up with shares in the same company i_1 that one starts with. Trading around a cycle is usually a bad idea, as you tend to end up with fewer shares than you started with. But occasionally, for short periods of time, there are opportunities to increase shares. We will call such a cycle an opportunity cycle, if trading along the cycle increases the number of shares. This happens exactly if the product of the ratios along the cycle is above 1. In analyzing the state of the market, a firm engaged in trading would like to know if there are any opportunity cycles.
Give a polynomial-time algorithm that finds such an opportunity cycle, if one exists

Problem 3.

Given a tree, color as many nodes black as possible, without coloring two adjacent nodes.

Problem 4.

Implement wildcard pattern matching with support for '?' and '*'.

- '? ': Matches any single character.
- '*' : Matches any sequence of characters (including the empty sequence).

The matching should cover the entire input string (not partial).

Examples:

isMatch("aa","a") → 0
isMatch("aa","aa") → 1
isMatch("aaa","aa") → 0
isMatch("aa", "*") → 1
isMatch("aa", "a*") → 1
isMatch("ab", "?*") → 1
isMatch("aab", "c*a*b") → 0

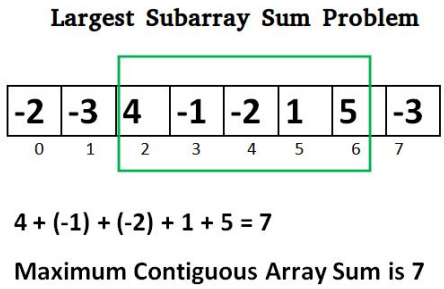
(isMatch is example function for the question)

Problem 5.

Given a string containing just the characters '(' and ')', find the length of the longest valid (well-formed) parentheses substring.
For "(()", the longest valid parentheses substring is "()", which has length = 2.
Another example is ")()())", where the longest valid parentheses substring is "()()", which has length = 4.

Problem 6.

Write an efficient program to find the sum of contiguous subarray within a one-dimensional array of numbers which has the largest sum.



(source: GFG)

Problem 7.

Imagine you have a special keyboard with the following keys:
Key 1: Prints 'A' on screen
Key 2: (Ctrl-A): Select screen
Key 3: (Ctrl-C): Copy selection to buffer
Key 4: (Ctrl-V): Print buffer on screen appending it
after what has already been printed.
If you can only press the keyboard for N times (with the above four keys), write a program to produce maximum numbers of A's. That is to say, the input parameter is N (No. of keys that you can press), the output is M (No. of As that you can produce).

Examples:
Input: N = 3
Output: 3
We can at most get 3 A's on screen by pressing
following key sequence.
A, A, A

Input: N = 7
Output: 9
We can at most get 9 A's on screen by pressing
following key sequence.
A, A, A, Ctrl A, Ctrl C, Ctrl V, Ctrl V

Input: N = 11
Output: 27
We can at most get 27 A's on screen by pressing
following key sequence.
A, A, A, Ctrl A, Ctrl C, Ctrl V, Ctrl V, Ctrl A,
Ctrl C, Ctrl V, Ctrl V