CSE 222 (ADA) Homework Assignment 2 (Theory)

Deadline : March 4 (Friday) 10am.

The theory assignment has to be done in a team of at most two members, as already selected by you. The solutions are to be typed either as a word document or latex-ed and uploaded as pdf on GC. We shall strictly not accept solutions written in any other form. Remember that both team members need to upload the HW solution on GC. Collaboration across teams or seeking help from any sources other than the lectures, notes and texts mentioned on the homepage will be considered an act of plagiarism

General instructions:

Each question is worth 15 points. For each question we need you to provide the following:

1. (3 points) A precise description of the subproblems you want to solve in the dynamic program. *Notice this is just the subproblem, not how to solve it, or how it was obtained.*
2. (3 points) A recurrence which relates a subproblem to "smaller" (Whatever you define "smaller" to mean) subproblems. *Notice this is just the recurrence, not the algorithm or why the recurrence is correct.*
3. (1 points) Identify the subproblem that solves the original problem
4. (3 points) A pseudo-code for a dynamic program which solves the recurrence efficiently *You do not need to prove correctness of the pseudocode*
5. (5 points) An argument for the running time of your dynamic program and the space requirement

We are only interested in the *value* of the optimal solution, not the optimal solution itself. So you do not need to give the reconstruction algorithm.

If your solution does not have the structure above, you will be awarded *a zero* (yes, you read that right). There will be *no* concessions on this.

Problem 1. Remember Mr. Monkey from Skull Island ? Well, he is back to King Kong's garden, but this time he stands in front of a row of magical banana trees that Kong has grown - let the trees be numbered as $t_1, t_2, \cdots t_n$ as usual. The bananas give Mr. Monkey an amazing power to jump over a certain number of trees. Eager to relive the jumping days of his youth, Mr. Monkey chalks the following plan. He will start walking from some point, say s which is to the left of the leftmost tree. From this point, assume that he moves only to his right. Once he reaches a tree, he has two choices - either to walk past it or climb it and eat exactly one banana. Eating a banana will magically make him jump a certain distance (even if he doesn't want to !), possibly over a few trees. In case he lands on the ground, he can continue the above exercise from that point. In case the first jump lands him on a tree, he can either climb down and continue the exercise or he can eat the bananas from there and continue jumping.

Let *dist* be an array that contains the distance of t_i from the starting point s and $jump$ be an array that contains the distance that Mr. Monkey can jump if he eats a banana from t_i. Write an algorithm that determines the maximum distance that Mr. Monkey can **jump**. Note that this does not include the distance he walks or climbs. You need to design an $\mathcal{O}(n^2)$ or faster for credit.

Problem 2. Mr. Monkey is so excited with his last adventure that this time he brings his partner Miss Chimp so that they can jump together. However, to their utter bewilderment, they find that Kong has removed most of the magical bananas and each tree has just one banana left. Assuming they both start at s and move only to the right, design an algorithm which determines the maximum length that the two of them together can jump. You get full credit for designing an $\mathcal{O}(n^3)$-time algorithm and 70% of the credit for designing an $\mathcal{O}(n^4)$-time algorithm. Anything slower than that will not fetch any credit even if correct.

Problem 3. The IIIT-Delhi robotics club has organized a Robothon. n robots are placed along the edge of a circular area at the middle of the OAT. Each robot will move along arbitrary tracks inside the circle while leaving behind a heat signature along its trail. However, they have been programmed not to cross their own trail or the trail of another robot, neither will they ever move out of the circle. In case a pair of robots i and j meet at any point, they are removed from the scene and the club will pay a reward sum of $M[i,j]$ to the owners of these robots. Note that some robots can keep moving infinitely without ever meeting another one. Given the reward matrix M where $M[i,j] = M[j,i]$, design a polynomial time algorithm that determines the maximum money the club might potentially end up spending. For this particular problem, give a very brief justification of the recurrence.