CS4335 Assignment 3

Due Dec. 2, 2016.
Each 24 hours late submission halves the score.
Submission: Hardcopy is required. Drop Hardcopy to the Mail Box75 (Floor 6, lift 9).

For all the problems in this assignment, STATE the running time of your methods. Polynomial time algorithms are expected.

Hint：Dynamic programming approach can solve all the 3 problems.

Question 1: Dancing List (30%)

A club wants to organize a party for the students. However, only part of the students can attend the party. The candidates are choosing by pairs with the following requirements: assume the students are lined-up, each student is given a lucky number, and some students may have the same lucky numbers. The club requires that only students who have the same luck numbers may be paired-up and join in the party while all the students locate within a chosen pair will lose the opportunity.

We have helped the club to formulate the problem formally.

Given a sequence of positive integers (represent the lucky numbers) A = (a_1, a_2, ... a_n), a pair (a_i, a_j) is defined as a ***dancing pair*** if a_i=a_j, and i<j. A **dancing pair list** L={(a_{i_1}, a_{j_1}),… (a_{i_k}, a_{j_k}) } is **good** if and only if for any two pairs (a_{i_x}, a_{j_x}), (a_{i_y}, a_{j_y}) of L, either $j_x<i_y$ or $j_y<i_x$ for $1 \leq x, y \leq k$.

For example, if A={1, 3, 4, 1, 2, 3, 2, 4, 3, 5, 3, 4, 5}, some good dancing lists might be:

{(1, 1), (2, 2), (3, 3)} (locations in A={1, 3, 4, 1, 2, 3, 2, 4, 3, 5, 3, 4, 5})

{(1, 1), (3, 3) (5, 5)} (locations in A={1, 3, 4, 1, 2, 3, 2, 4, 3, 5, 3, 4, 5})

{(3,3),(4,4)} (locations in A={1, 3, 4, 1, 2, 3, 2, 4, 3, 5, 3, 4, 5})

{(3,3), (5,5)} (locations in A={1, 3, 4, 1, 2, 3, 2, 4, 3, 5, 3, 4, 5})

Now your tasks are to help the club to identify the candidates; that is,

a) Given a sequence of positive integers A, design a polynomial time algorithm to find a good dancing pair list with maximum number of dancing pairs.
b) Assume the weight of the dancing pair (a_i, a_j) is a_i+a_j. The total weight of a good dancing list is the sum of all the weights of the dancing pairs in the list. Design a polynomial time algorithm to find a good dancing pair list with maximum total weight.

CS4335 Assignment 3

Due Dec. 2, 2016.
Each 24 hours late submission halves the score.
Submission: Hardcopy is required. Drop Hardcopy to the Mail Box75 (Floor 6, lift 9).

Question 2: (30 %) The input is the same as Question 1. However, the valid dancing pairs are either disjoint (i.e., for any two pairs (a_{i_x}, a_{j_x}), (a_{i_y}, a_{j_y}) of L, either $j_x<i_y$ or $j_y<i_x$ for $1 \le x, y \le k$ **as in Question 1) or nested (**i.e., for any two pairs (a_{i_x}, a_{j_x}), (a_{i_y}, a_{j_y}) of L, either $j_x<j_y$ and $i_x>i_y$ or $j_x>j_y$ and $i_x<i_y$ for $1 \le x, y \le k$).

For example, Let A={1, 3, 4, 1, 2, 2, 1,4, 3, 5, 3, 4, 5}.

(4,4), (1,1), (2,2) and (5,5) are valid dancing pairs and their locations in A are color as follows:

A={1, 3, 4, 1, 2, 2, 1,4, 3, 5, 3, 4, 5}, where there are three nested pairs: (4,4), (1,1), and (2,2).

Moreover, the pair (5,5) is disjoint with the other three nested pairs (4,4), (1,1), and (2,2).

Design a polynomial time algorithm to find a valid dancing pair list with maximum number of dancing pairs.

Question 3: (40%) (Three Player Games) Three people will play a game using 3n cards, where each card has an integer from 1 to 10. The 3n cards are organized in three rows, n cards each. The following figure gives an example for n=3, where X is the center and there is no card on center.

1

8

6

6 X 2

2 9

3 6

The three players will take a card in turn (player 1, player 2 and player 3 and repeat) at the three non-center ends. At the beginning, every player has three choices. If a row becomes empty, then the player can only take cards from the two remaining rows at the non-center ends. If two rows become empty, every play can only take the cards at the non-center end of the remaining row (i.e., only one choice is left). Every player will have n cards at the end of the game. The one who have the maximum total value of the cards will win. Suppose that you are the first player. Design an algorithm to get n cards that is the most advantage for you. Assume that the other two players are genius and they never make any mistake.