B505/I500: Applied Algorithms

HW3 (Due: **Mar. 11 Monday 5pm**)

https://iu.instructure.com/courses/1771436

Please submit your completed homework through canvas using pdf file format

1. (20 pts) Suppose two teams A and B are playing a match to see who is the first to win n games (from given n). Assume that A and B are equally competent, so each has a 50% chance of winning any particular game. Suppose they have already played i+j games, of which A won i and B has won j. Given an efficient algorithm to compute the probability that A will go on to win the match. For example, if i=n-1 and j=n-3 then the probability that A will win the match is 7/8, since it must win any of the next three games.

2. (20 pts) Give an O(nt) algorithm for the following task.

   Input: a list of n positive integers $a_1, a_2, \ldots, a_n$; a positive integer t.
   Question: does some subset of the a_i's add up to t?

3. (20 pts) Consider the following problem: given an array A with n numbers A[1], A[2], …, A[n], find two numbers A[i] and A[j], i < j, whose difference A[j] – A[i] is maximum. a) devise a divide-and-conquer algorithm to solve the problem; what is the running time complexity? B) devise a dynamic programming algorithm to solve it. What is the running time complexity.

4. (20 pts) Alice wants to organize a party and is deciding whom to call. She has *n* people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least five other people whom they know and five people whom they don't know. Given as input the list of *n* people and the list of pairs who know each other, devise an algorithm to output the best choice of party invitees.

5. (20 pts) Consider the following variation on the Scheduling Problem. You have a processor that can operate 24 hours a day, every day. People submit requests to run daily jobs on the processor. Each such job comes with a start time and an end time; if the job is accepted to run on the processor, it must run continuously, every day, for the period between its start and end times. Given a list of *n* such jobs, your goal is to accept as many jobs as possible (regardless of their length), subject to the constraint that the processor can run at most one job at any given point in time. Provide an algorithm to do this with a running time that is polynomial in *n*. You may assume for simplicity that no two jobs have the same start or end times.

   Example. Consider the following four jobs, specified by (start time, end-time)

pairs. (6 P.M., 6 A.M.), (9 P.M., 4 A.M.), (3 A.M., 2 P.M.), (1 P.M., 7 P.M.). The optimal solution would be to pick the two jobs (9 P.M., 4 A.M.) and (1P.M., 7 P.M.), which can be scheduled without overlapping.