CS 473, Fall 2017
Homework 3 (due September 27 Wednesday at 8pm)

You may work in a group of at most 3 students. Carefully read `http://engr.course.illinois.edu/cs473/policies.html` and `http://engr.course.illinois.edu/cs473/integrity.html`. One member of each group should submit via Gradescope.

1. [*13 pts*] Consider the subset sum problem: given n positive integers $a_1, \ldots, a_n$ and a target integer T, decide whether there exists a subset $S \subset \{a_1, \ldots, a_n\}$ that sums to exactly T. In class, we have described an $O(nT)$-time algorithm via dynamic programming, which requires $O(nT)$ space if we want to output the subset when the answer is yes. Apply Hirschberg's space-saving trick to obtain an $O(nT)$-time, $O(T)$-space algorithm that outputs a subset summing to exactly T when the answer is yes. Describe the details by giving complete pseudocode.

2. [*20 pts*] Given a weighted *directed acyclic graph (dag)* $G = (V, E)$ with n vertices and an integer $k \leq n$, we want to find at most k vertex-disjoint paths that visit all vertices in V, minimizing the total weight of the paths. Describe a dynamic programming algorithm that solves this problem in time $O(n^{k+c})$ time for some constant c.

   Include the following steps: (a) first define your subproblems precisely, (b) give base cases, (c) derive the recursive formula, (d) specify the evaluation order, and (e) analyze the running time. For this problem, you do not need to provide pseudocode (if the description of the recursive formula etc. is sufficiently clear), and you do not need to describe how to output the optimal paths, just the total weight.

3. [*12 pts*] Suppose we have a ("Las Vegas") randomized algorithm A that always successfully returns a correct answer and runs in μ *expected* number of steps. We can obtain a new ("Monte Carlo") algorithm A$'$ that *always* runs in at most 100μ steps by simply stopping algorithm A and declaring failure after 100μ steps. The probability that A$'$ fails is at most 0.01 by Markov's inequality.

   But we can do better! If we run algorithm A for at most 10μ steps and rerun algorithm A for another 10μ steps (with an independent sequence of random bits), the probability of failure is still at most $0.1 \times 0.1 = 0.01$ but the number of steps is always at most 20μ.

   Describe and analyze a still better strategy that always runs in at most $c\mu$ steps with probability of failure at most 0.01, for as small a constant c as you can find.