Homework # 3
Hard copy is due in class on Wednesday, Nov 29, 2017.

Two questions will be chosen for grading. The same guidelines as for Homework 1 apply here. Make sure both to reason about the correctness of your algorithms, as well as their running times.

1. Recall the vertex cover problem: given an undirected graph $G = (V, E)$, find the smallest subset $U$ of $V$ that covers all edges of $E$. That is, each edge of $E$ has at least one end-point in the set $U$.

Consider the following randomized algorithm for the vertex cover problem. Prove as good a bound as your can on its approximation ratio.

**Algorithm:** Consider the edges in arbitrary order. If the edge under consideration is not covered, pick one of its end points uniformly at random and add it to the vertex cover.

2. You have won a prize at your local grocery store that entitles you to as many groceries as you want up to a certain total dollar amount $T$. You can take an unlimited supply of any of the $n$ items in stock at the store, as long as you stay within your total dollar amount. You want to determine whether it is possible to spend the maximum possible dollar amount $T$, i.e. whether there is a set of items in stock at the store (possibly duplicating some items), whose prices sum to exactly $T$.

   (a) Design an algorithm to solve this problem in $O(nT)$ time. The algorithm takes as input a list of items in stock at the store, along with their costs, as well as your prize amount $T$. You don’t have to output the set of items which total $T$, only whether it is possible to spend exactly $T$ dollars.

   (b) Explain how you would adapt your algorithm so that it outputs the maximum amount that you could spend at the store (which may be less than $T$).

   (c) Suppose that your prize amount $T$ is very large, and you don’t have time to run your $O(nT)$ algorithm before the prize expires. Describe an algorithm that takes the same input as part (a), plus a positive constant $\epsilon < 1$, and finds a set $S$ of items with the following property: the maximum amount that you could spend is no more than $(1 + \epsilon)$ times the total value of the items in $S$. Your algorithm should run in time polynomial in $n$ and $1/\epsilon$.

3. (From Jeff Erickson, see http://jeffe.cs.illinois.edu/teaching/algorithms/notes/14-mincut.pdf)

   You are given a undirected graph with weighted edges, and your goal is to find a cut whose total weight (not just number of edges) is smallest.

   Prove that if you choose edges proportional to their weight rather than uniformly at random, the probability that the GuessMinCut algorithm returns a minimum-weight cut is still $\Omega(1/n^2)$, where $n$ is the number of nodes of the graph.

   The adapted GuessMinCut algorithm is as follows:

   **GuessMinCut($G$)**

   For $i \leftarrow n$ downto 2
   
   Pick an edge $e$ in $G$ at random, with probability proportional to its weight
   
   $G \leftarrow G/e$  // $G/e$ is the graph obtained by collapsing edge $e$

   Return the only cut in the resulting $G$