15-351 / 15-650 / 02-613 Homework #7

Due: Friday, Dec. 11 by 4:59pm

You may discuss these problems with your current classmates, but you must write up your solutions independently, without using common notes or worksheets. You must indicate at the top of your homework who you worked with. Your write up should be clear and concise. You are trying to convince a skeptical reader that your answers are correct. Avoid pseudocode if possible. Your homework should be submitted via **GradeScope** as a typeset PDF. A LaTeX tutorial and template are available on the class website if you choose to use that system to typeset. For problems asking for an algorithm: describe the algorithm, an argument why it is correct, and an estimation of how fast it will run. Use O notation for running times.

1. **How Universal?** In the lecture videos, we talked about universal hash functions. There is an extension of this idea to L-universality. Specifically, a function is called L-universal if for every set of L distinct keys $x_1, x_2, \ldots, x_L$ and every set of L values $v_1, v_2, \ldots, v_L \in \{0, \ldots, M-1\}$, we have

$$\Pr_{h \leftarrow H}[h(x_1) = v_1 \text{ AND } h(x_2) = v_2 \text{ AND } \ldots \text{ AND } h(x_L) = v_L] = 1/M^L.$$

   It is easy to see that if H is 2-universal then it is universal. Note that the matrix-based hash family that we saw in the lecture videos was not 2-universal since the hash functions all mapped 0 to 0.

   Show that if we choose $A \in \{0,1\}^{m \times u}$ (where u is the key length and $M = 2^m$) and $b \in \{0,1\}^m$ independently and uniformly at random, then the hash family $h(x) = Ax + b$ is 2-universal.

2. **Hard to approximate.** Prove that the problem of finding a k-approximation of the TSP is NP-hard for any constant k. I matrix between n cities; is there a TSP tour of total le ete for any value of k. Hint: reduce from Haml

3. **Fewest suitcases.** Suppose we are given a set of n t i is s_i, with $0 < s_i < 1$. We want to put these objects into a series of suitcases, each of which has a capacity of 1 unit of weight. We have as many suitcases as we want, but we want to use the fewest suitcases. This is an NP-hard problem.

   The "first-fit" algorithm: process the objects and suitcases in arbitrary order and put it in the first suitcase that can still hold it. If no suitcase can, grab a new suitcase and place it in that.

   (a) Let $S = \sum_{i=1}^{n} s_i$. Show that the optimal number of suitcases is $\geq \lceil S \rceil$.

   (b) Prove that first-fit never uses more than $\lceil 2S \rceil$ suitcases.

   (c) Explain why this shows that first-fit is a 2-approximation algorithm for this problem.