You may discuss these problems with your 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. Your homework should be submitted via Autolab (https://autolab.cs.cmu.edu/02713-s13/) as a typeset PDF.

1. You’re driving from Los Angeles, CA to Pittsburgh, PA. There are gas stations along the way at distance $x_1, x_2, \ldots, x_n$ from Los Angeles. Because of different wait times and pump speeds, filling up at gas station $x_i$ takes $c_i$ minutes (the gas costs the same everywhere, so we ignore its cost). Your car can hold enough gas to go 100 miles, and you start with a full tank of gas. If you decide to stop at a gas station, you have to fill your entire tank up. Give a dynamic programming algorithm that finds where you should stop to spend the minimum amount of time at gas stations during your trip.

Hint: you know you’ll have to stop at a gas station within 100 miles of Pittsburgh, for example.

2. Let’s change the problem above slightly: suppose if you stop, you don’t need to fill up the entire tank. Instead, if you put in $m$ miles worth of gas, it will take you $c_i + mg_i$ minutes at station $x_i$. For simplicity, assume that you start out with an empty tank but you start at a station $x_1$ in LA, and that $x_n$ is your destination in Pittsburgh. Give a dynamic programming algorithm to solve this problem.

3. In some languages, such as Chinese, words are sometimes not separated by spaces. For another example, in German, numbers are sometimes written together “Dreihundertfünfundfünftausend” and compound words are often created: “Glückszahl” means “lucky Glück number (zahl).” We would like to decompose such strings into the component words that were used to form them. Assume you have a function word($s$) that takes a string $s$ and returns a score indicating how likely it is that $s$ is a indivisible word. For example, in German, “zahl” would receive a high score, but “kszahal” would not. Give a dynamic programming algorithm to break a given string $a = a_1a_2\ldots,a_n$ into words $w_1,\ldots,w_k$ to maximize $\sum_i \text{word}(w_i)$. (Note that you are not given $k$ as input.)

4. Programming problem. Suppose you are trying to find where genes are in a long string of DNA $d$. Researchers have created a number of programs that predict possible gene locations, but they often include a lot of false and contradictory predictions. Suppose you are given a set $G = \{(s_i, e_i, r_i)\}$ of predictions where $0 \leq s_i < |d|$ is the start of predicted gene $i$, $1 \leq e_i < |d|$ is the end of the predicted gene, and $0 < r_i < 1$ is probability that the prediction is correct. The predicted genes in $G$ can overlap, but in real genomes overlapping genes are somewhat rare (they actually happen, but for the purposes of this problem, assume that they do not).

Write a Python program that implements a dynamic programming algorithm to find a subset $G' \subseteq G$ of predictions such that (1) no pair of predictions in $G'$ overlap, and (2) $\sum_{g \in G'} r_g$ is maximized. (For simplicity, we are dealing with only 1 strand of DNA at a time.)

Input: A file containing $|G|$ lines of the form 10 100 0.234, meaning that a gene was predicted to go from position 10 to position 100 with probability 0.234.

Output: A series of lines of the form Include: 10 100 0.234, meaning to include the gene with those parameters. A single line of the form Sum Probability: 0.01, that gives the sum of the probabilities of the chosen set of genes.

Technicalities: Your program should be called choosegenes.py. You can use matplotlib, numpy, or networkx, or anything in the python standard library. Otherwise, you can use no other code that you didn’t write.