Problem Set 6
Due: Monday, November 18, 2019 (within the first 5 minutes of class).
At most one late day allowed.

Problem 1. Lempel-Ziv encoding
(a) [10 pts] Encode the following string using Lempel-Ziv encoding:
B B A A B B A B A B A A
Show the contents of the (new) dictionary and the actual encoded string.
(b) [10 pts] Decode the following string using Lempel-Ziv decoding:
66 256 66 65 258 259
Show the contents of the (new) dictionary and the actual decoded string.

Problem 2. Shortest paths
(a) [12 pts] Run Dijkstra’s algorithm on the graph of Figure 24.2 in the text (page 648) using vertex z (not s!) as the source. Show (1) the order in which nodes are removed from the heap [5 pts], and (2) the updates made to the distances and predecessors of each node [7 pts].
(b) [10 pts] Suppose we want to send goods by truck from a production facility in New York to a warehouse in Chicago. We would like to make as few trips as possible (since we have only one truck), and the critical issue is that each road segment (the piece of a road between two intersections) has a weight limit for the truck. For example, if a road segment has weight limit 10 tons, then we either have to put at most 10 tons of goods in the truck, or not use that road segment. Suppose we know the weight limit for each road segment.
Our goal is to find a route which allows the heaviest truck load (i.e. with the highest weight limit), even if it is longer than a route with a lower weight limit. For example, for the graph in Figure 24.2, if the edge weights are the loads, then the best path from s to x is s − y − z − x with a weight limit of 5.
Carefully describe an efficient algorithm to find such a route. Specifically, if you are using a modified version of an algorithm we’ve discussed, describe the changes. You do not need to provide pseudocode.
(c) [5 pts] If there are r road segments (which connect two intersections) and j intersections (where we can change roads), what is the running time of your solution in (b)?

Problem 3. Dynamic programming
You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts a1 < a2 < . . . < an, where each ai is measured from the starting point. The only places you are allowed to stop are at these hotels, and you can choose which of the hotels you stop at. You must stop at the final hotel (at distance an), since that is your destination. Ideally, you would like to travel exactly 300 miles between hotel stops, but this may not be possible (due to the spacing of the hotels). If you travel x miles during a day, the “penalty” for that day is |300 − x|.
You want to plan your trip so as to minimize the total penalty. Give an efficient dynamic programming solution that determines the minimum penalty that can be incurred from this trip.
Hint: You can solve this problem with a 1D array T[i]. Think about what T[i] should store (remember that you eventually want to return the minimum penalty for up to hotel n). It may help to (1) look back at other dynamic programming problems we have seen in this class and (2) work through the example below by hand.
For example, if the mile posts are 10, 200, 240, 350, 410, and 630 then the minimum penalty to reach the last hotel is 30 with stops at hotels 2, 5, and 6.

As usual, your solution should include:

- [2] The size of the array or matrix you will use to compute your solution (the minimum penalty to get to the last hotel).
- [3] What each entry of the array or matrix holds (in words).
- [7] The dynamic programming formulation (how to fill each entry, i.e. mathematically), including the base cases.
- [3] How to fill the entire array.
- [2] Where the optimal value (the minimum penalty to get to the last hotel) is located once the array is filled.
- [6] How to find the sequence of hotels at which to stop that yields the minimum penalty.