CS302 - Problem Set 10

1. (a) Write pseudocode to output the shortest path using the array produced in the first part of the Bellman-Ford algorithm.

Algorithm 1: `Shortest Path`(G, w, A)

Input : A graph $G = (V, E)$, an array of weights w where $w[u, v]$ gives the weight of edge (u, v), and an array A where the first dimension of A is indexed by the elements of V, and the second dimension is labeled by indices from 0 to $|V| - 1$. Then $A[v, i]$ contains the length of the shortest path from s to v that uses at most i edges.

Output: The shortest path from s to t.

```
1 if A[t, n − 1] = INFINITY then
2 |  return "No Path";
3 end
  // Your pseudocode here!
```

   (b) What is the runtime of your pseudocode if you are given G as an adjacency matrix? Explain.

   (c) What is the runtime of your pseudocode if you are given G as an adjacency list? Do you want a regular or reverse list? Explain.

2. (a) In the version of Bellman-Ford we looked at in class, we assume that there are no negative cycles in the graph. (Or at least none that you can get to from s by following a path - there could be disconnected negative cycles, in which case all of the nodes on those cycles will have distance INFINITY.) Sketch using words how you would change the Bellman-Ford algorithm so that if the input does not have a negative cycle reachable from s, it returns the shortest path, and if the input graph does have a negative cycle reachable from s, it returns "Negative Cycle Detected." You don't need to be super precise - just give the idea of the approach and roughly why it works.

   (b) **(Challenge)** Describe more specifically how you would modify the algorithm, (in a way that causes the smallest increase possible to the runtime), and prove your algorithm detects a negative cycle.

3. Consider the following variations to problems we've solved using dynamic programming algorithms. If the algorithm no longer works, explain why, and then briefly describe a fix and the impact of the fix on the runtime. If the algorithm can stay the same, you don't need to explain.

(a) In MWIS on a line, we only considered graphs where the vertex weights were positive integers. Would we have to change the algorithm if we used negative integers instead?

(b) In MWIS on a line, we only considered graphs where the vertex weights were positive integers. Would we have to change the algorithm if we used positive real numbers instead?

(c) In the Knapsack problem, we only considered items whose values were positive integers. Would we have to change the algorithm if the values of items were allowed to be negative integers instead? (Assume that each item's cost is still a positive integer.)

(d) In the Knapsack problem, we only considered items whose values were positive integers. Would we have to change the algorithm if the values of items were allowed to be positive real numbers instead? (Assume that each item's cost is still a positive integer.)

(e) [**Challenge**] In the Knapsack problem, we only considered items whose costs were positive integers and the total capacity was a positive integer. Would we have to change the algorithm if the costs of items were allowed to be positive or negative integers? (A negative cost would mean that adding that item to your knapsack would increase the size of your knapsack.) (Assume that each item's value is still a positive integer.)

(f) In the Knapsack problem, we only considered items whose costs and values were positive integers and the total capacity was a positive integer. Would we have to change the algorithm if we used both negative and positive integers for item costs and values?

4. Approximately how long did you spend on this assignment (round to the nearest hour)?