The homework assignment is available at:
`http://www.jennylam.cc/courses/146-s17/homework10.html`

1. Suppose that a directed graph G has the property that every shortest path from the starting vertex s to every other vertex has at most four edges. What would this fact imply about the running time of the Bellman-Ford algorithm for finding shortest paths starting from s in G?

   Solution. Bellman-Ford stops updating any distance values after four iterations. So with the optimization that checks whether any distance values were changed after each iteration and stops if there were no changes, the Bellman-Ford algorithm runs five iterations of the outer-loop. There are m relaxations in each loop, so the running time is $5m$ or $O(m)$. •

2. In the lecture on dynamic programming (part 1), we give a version of the Floyd-Warshall algorithm that uses a 3-dimensional array. Notice that the values `dist[-][-][k]` in the table only ever depend on values of the form `dist[-][-][k-1]`, and not on any smaller values of the 3rd index.

   (a) Using this observation, propose a minor modification to this algorithm which uses two 2-dimensional arrays instead. Give your answer in code or pseudocode.

   Solution. The idea is that we only two 2-dimensional tables: one which we will call `dist1` for the distances computed on the previous iteration of k, and one which we call `dist2` to store the distances computed on the current instance of k. We are careful to create a temporary variable to swap the roles of the arrays created at the beginning of the code, rather than create a new table on every iteration and letting garbage collection handling memory recouping.

```
int[][][] dist1 = new int[V][V]
int[][][] dist2 = new int[V][V]
for (int i = 0; i < V; i++)
   for (int j = 0; j < V; j++)
      dist2[i][j] = i -> j ? w(i -> j) : (i == j ? 0 : inf)
for (int k = 1; k < V+1; k++)
   int[][][] temp = dist1
   dist1 = dist2
   dist2 = temp
   for (int i = 0; i < V; i++)
      for (int j = 0; j < V; j++)
         dist2[i][j] = min(dist1[i][j],
               dist1[i][k] + dist1[k][j])
return dist2;
```

   •

   (b) How are the time and space complexities affected by this modification? If there is a change, what is the new complexity?

   Solution. The time complexity remains unchanged, as this is only an optimization to the space complexity, which is reduced from $O(V^3)$ downs to $O(V^2)$. •