CS3230 Tutorial 7 (Dynamic Programming) Sample Solutions

March 6, 2017

1 Question 1

1.a

$C(6, 3)$ can be computed by filling the following table:

	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10
6	1	6	15	**20**

1.b

Yes, the table can also be filled column-by-column, with each column filled top-to-bottom starting with 1 on the main diagonal of the table.

2 Question 2

$C(n, k)$ is the number of ways to choose a k-element subset from a set of n elements, which is the same as selecting $n - k$ elements not to be included in a k-element subset. (A formal proof was given in tutorial class.)

Compute $C(n, k)$ if $k \leq n - k$ (i.e., if $k \leq n/2$) and $C(n, n - k)$ otherwise.

3 Question 3

3.a

Applying Warshall's algorithm yields the following sequence of matrices (in which newly updated elements are shown in bold):

$$R^{(0)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(1)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(2)} = \begin{bmatrix} 0 & 1 & \mathbf{1} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(3)} = \begin{bmatrix} 0 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$R^{(4)} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T == \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

3.b

With the book's definition of the transitive closure (which considers only nontrivial paths of a digraph), a digraph has a directed cycle if and only if its transitive closure has a 1 on its main diagonal. The algorithm that finds the transitive closure by applying Warshall's algorithm and then checks the elements of its main diagonal is cubic. This is inferior to the quadratic topological sorting algorithms for checking whether a digraph represented by its adjacency matrix is a dag.

4 Question 4

4.a

Applying Floyd's algorithm to the given weight matrix generates the following sequence of matrices:

$$D^{(0)} = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$

$$D^{(1)} = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \mathbf{14} \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \mathbf{5} & \infty & \mathbf{4} & 0 \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 2 & \mathbf{5} & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & \mathbf{8} & 4 & 0 \end{bmatrix}$$

$$D^{(3)} = \begin{bmatrix} 0 & 2 & 5 & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & 8 & 4 & 0 \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 2 & \mathbf{3} & 1 & \mathbf{4} \\ 6 & 0 & 3 & 2 & \mathbf{5} \\ \infty & \infty & 0 & 4 & \mathbf{7} \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & \mathbf{6} & 4 & 0 \end{bmatrix}$$

$$D^{(5)} = \begin{bmatrix} 0 & 2 & 3 & 1 & 4 \\ 6 & 0 & 3 & 2 & 5 \\ \mathbf{10} & \mathbf{12} & 0 & 4 & 7 \\ \mathbf{6} & \mathbf{8} & 2 & 0 & 3 \\ 3 & 5 & 6 & 4 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 2 & 3 & 1 & 4 \\ 6 & 0 & 3 & 2 & 5 \\ 10 & 12 & 0 & 4 & 7 \\ 6 & 8 & 2 & 0 & 3 \\ 3 & 5 & 6 & 4 & 0 \end{bmatrix}$$

4.b

As a simple counter example, one can suggest Figure 1:

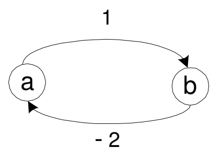

Figure 1: The counter example.

Floyd's algorithm will yield:

$$D^{(0)} = \begin{bmatrix} 0 & 1 \\ -2 & 0 \end{bmatrix}$$

$$D^{(1)} = \begin{bmatrix} 0 & 1 \\ -2 & -1 \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} -1 & 0 \\ -3 & -2 \end{bmatrix}$$

None of the four elements of the last matrix gives the correct value of the shortest path, which is, in fact,$-\infty$ because repeating the cycle enough times makes the length of a path arbitrarily small.

NOTE: You may find the result of $D^{(2)}$ unexpected. The reason for the output in $D^{(2)}$ is that in the standard implementation of Floyd's algorithm, there are three nested loops and the column and row where we placed the "bars" in the lecture are not excluded from the calculation. So it looks like this:

```
for k from 1 to |V|         // Standard Floyd-Warshall implementation.
   for i from 1 to |V|
      for j from 1 to |V|
         if (dist[i][k] + dist[k][j] < dist[i][j]) then
            dist[i][j] <- dist[i][k] + dist[k][j]
```

See https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm.

In our manual computation in the lecture we excluded the entries in the "bars" (where $i == k$ or $j == k$). As you can see in the code above, there is no such exclusion in the standard algorithm. If the weights are positive then entries in the "bars" will not change, because the value that is already there will be the minimum. However, if there are negative weights in the matrix, then those "bar" entries will also be affected. Therefore, with a negative weight cycle the result is not correct.

Any bugs and typos, please report to Roger Zimmermann (rogerz@comp.nus.edu.sg).