```
CS 519-005, Algorithms (MS/MEng-level), Winter 2018
HW8 - DP (part 3), Graph Algorithms (part 1)

Due on Monday Mar 5, 11:59pm.
No late submission will be accepted.

Include in your submission: report.txt, lis.py, topol.py, viterbi.py.
viterbi.py will be graded for correctness (1%).

Textbooks for References:
[1] CLRS Ch. 15
[2] KT Ch. 6, freely available online (strongly recommended!):
    http://www.aw-bc.com/info/kleinberg/assets/downloads/ch6.pdf
[3] Wikipedia: Longest Increasing Subsequence
[4] my DP tutorial (up to page 16):
    http://web.engr.oregonstate.edu/~huanlian/slides/COLING-tutorial-anim.pdf
[5] DPV Ch. 3, 4.2, 4.4, 4.7, 6 (Dasgupta, Papadimitriou, Vazirani)
    https://www.cs.berkeley.edu/~vazirani/algorithms/chap3.pdf
    https://www.cs.berkeley.edu/~vazirani/algorithms/chap4.pdf
    https://www.cs.berkeley.edu/~vazirani/algorithms/chap6.pdf

Please answer time/space complexities for each problem in report.txt.

(LIS is not discussed in class)

0. (a) Describe a greedy algorithm for LIS and show a counter example.
   (b) Describe an exhaustive algorithm for TSP and analyze complexity.

1. Longest (Strictly) Increasing Subsequence

   input/output are lower-case strings:

   >>> lis("aebbcg")
   "abcg"

   >>> lis("zyx")
   "z"

   tiebreaking: arbitrary. any optimal solution is ok.

   Q: What are the time and space complexities?

   filename: lis.py

2. Topological Sort

   For a given directed graph, output a topological order if it exists.

   Tie-breaking: ARBITRARY tie-breaking. This will make the code
   and time complexity analysis a lot easier.

   e.g., for the following example:

     0 --> 2 --> 3 --> 5 --> 6
        /    \   |   /    \
       /      \  v /       \
     1         > 4          > 7

   >>> order(8, [(0,2), (1,2), (2,3), (2,4), (3,4), (3,5), (4,5), (5,6), (5,7)])
   [0, 1, 2, 3, 4, 5, 6, 7]

   If we flip the (3,4) edge:

   >>> order(8, [(0,2), (1,2), (2,3), (2,4), (4,3), (3,5), (4,5), (5,6), (5,7)])
   [0, 1, 2, 4, 3, 5, 6, 7]

   If there is a cycle, output None

   >>> order(4, [(0,1), (1,2), (2,1), (2,3)])
   None

   filename: topol.py

   questions:
   (a) did you realize that bottom-up implementations of DP use (implicit) topological
orderings?
       e.g., what is the topological ordering in your (or my) bottom-up bounded knapsack
code?
   (b) what about top-down implementations? what order do they use to traverse the graph?
   (c) does that suggest there is a top-down solution for topological sort as well?

3. [WILL BE GRADED]
   Viterbi Algorithm For Longest Path in DAG (see DPV 4.7, [2], CLRS problem 15-1)

   Recall that the Viterbi algorithm has just two steps:
   a) get a topological order (use problem 1 above)
   b) follow that order, and do either forward or backward updates

   This algorithm captures all DP problems on DAGs, for example,
   longest path, shortest path, number of paths, etc.

   In this problem, given a DAG (guaranteed acyclic!), output a pair (l, p)
   where l is the length of the longest path (number of edges), and p is the path. (you
can think of each edge being unit cost)

   e.g., for the above example:

   >>> longest(8, [(0,2), (1,2), (2,3), (2,4), (3,4), (3,5), (4,5), (5,6), (5,7)])
   (5, [0, 2, 3, 4, 5, 6])

   Tie-breaking: arbitrary. any longest path is fine.

   Filename: viterbi.py

   Note: you can use this program to solve LIS, TSP, knapsacks, MIS, etc.


Debriefing (required!): --------------------------

0. What's your name?
1. Approximately how many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Did you work on it mostly alone, or mostly with other people?
4. How deeply do you feel you understand the material it covers (0%–100%)?
5. Which part(s) of the course you like the most so far?
6. Which part(s) of the course you dislike the most so far?

This section is intended to help us calibrate the homework assignments.
Your answers to this section will *not* affect your grade; however, skipping it
will certainly do.
```