Spring 2017. Algorithms qualifying exam solutions.

Closed books, closed notes.

1. (30 points) Suppose you are choosing between the following three algorithms:
   - Algorithm A solves problems of size n by recursively solving 3 subproblems of size n-1 and then obtaining the solution to the original problem in constant time.
   - Algorithm B solves problems of size n by recursively solving 2 subproblems, one of size n/4 and the other one of size 3n/4, and then combining the solutions in linear time.
   - Algorithm C solves problems by dividing them into 12 subproblems of size n/3, recursively solving each subproblem, and then combining the solutions in O(n^3) time.

a) Compute the running time of each of these algorithms (in big-O notation). Show your work (use one of the methods for solving recurrencies, e.g., iteration method, recursion tree method, Master theorem, substitution method).
b) Which algorithm would you choose? Explain your answer.

Solution.
a) Algorithm A: T(n) = 3T(n-1) + c. Use iteration method or recursion tree method to get T(n) = O(3^n).
Algorithm B: T(n) = T(n/4)+T(3n/4)+cn. Use recursion tree method to get T(n) = O(n log n).
Algorithm C: T(n) = 12 T(n/3) + O(n^3). By the Master theorem, T(n) = O(n^3).

b) Algorithm B is the best because it has the smallest asymptotic running time.

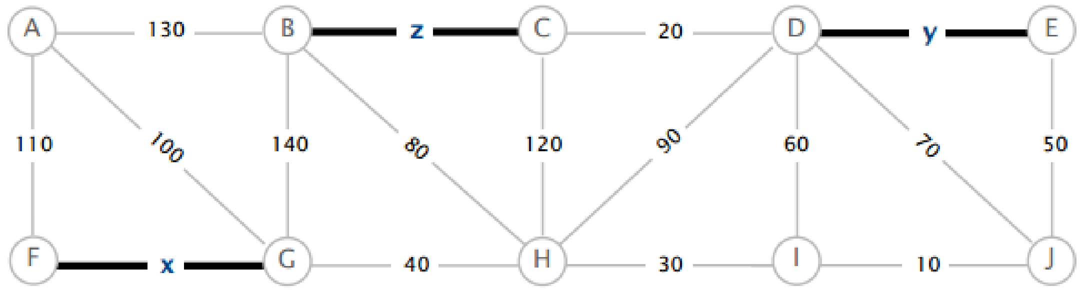
2. (20 points) Assume G is a directed acyclic graph (DAG). A vertex in G is a *sink* if it has no outgoing edges. A *forward path* from vertex v is a path that ends in a sink. Give an efficient algorithm to determine the number of forward paths from every node in DAG G in linear time. If a vertex is a sink then the number of forward paths from it is 1. Show that the running time of your algorithm is indeed linear.

Solution.
Do a DFS of G=(V,E). In postvisit procedure (that is, just before explore(G,v) returns), insert the following step:

```
if (v has no outgoing edges)
        then NumForPaths[v]=1
        else NumForPath[v] = sum of NumForPath[u] over all u such that (v,u)∈ E.
```

Running time of DFS is linear. Modified version also takes linear time.

3. (20 points) Suppose that a minimum spanning tree T of the following edge-weighted graph G contains the edges with weights x, y, and z (that is, edges (FG), (DE), and (BC)).



(a) What other edges in G must be in T?

Solution uses the Cut property:
Cut property Suppose edges X are part of a minimum spanning tree of G = (V,E). Pick any subset of nodes S for which X does not cross between S and V - S, and let e be the lightest edge across this partition. Then X ∪ {e} is part of some MST.

X={(FG), (BC), (DE)}
We will be adding edges to X using the cut property until we find a MST:
Edge (I,J): S={J}. By the cut property, {(FG), (BC), (DE), (I,J)} is part of a MST.
Edge (CD): S = {D,E}. By the cut property, {(FG), (BC), (DE), (I,J), (CD)} is part of a MST.
Edge (HI). S={I,J}. By the cut property, {(FG), (BC), (DE), (I,J), (CD),(HI)} is part of a MST.
Edge (GH). S={H,I,J}. By the cut property, {(FG), (BC), (DE), (I,J), (CD),(HI),(GH)} is part of a MST.
Edge (EJ). S={F,G,H,I,J}. By the cut property, {(FG), (BC), (DE), (I,J), (CD),(HI),(GH),(EJ)} is part of a MST.
Edge (AG). S={A}. By the cut property, {(FG), (BC), (DE), (I,J), (CD),(HI),(GH),(EJ),(AG)} is part of a MST.
Therefore, a MST consists of edges (FG), (BC), (DE), (I,J), (CD),(HI),(GH),(EJ),(AG). We will call this tree T.

(b) Circle which one or more of the following can be the value of x?
5 15 25 35 45 55 65 75 85 95 105 115 125 135 145

Solution: (FG) must be the lightest edge connecting {F} to the rest of the vertices. That is, x must be <=110. Otherwise, by removing (FG) from T and adding (AF) we would get a tree with a smaller cost. Therefore, the following could be the values of x:
5 15 25 35 45 55 65 75 85 95 105

(c) Circle which one or more of the following can be the value of y?
5 15 25 35 45 55 65 75 85 95 105 115 125 135 145
Solution: (BC) must be the lightest edge connecting {B} to the rest of the vertices. That is, z must be <=60. Otherwise, by removing (DE) from T and adding (DI) we would get a tree with a smaller cost. Therefore, the following could be the values of y:
5 15 25 35 45 55

(d) Circle which one or more of the following can be the value of z?

5 15 25 35 45 55 65 75 85 95 105 115 125 135 145

Solution: (DE) must be the lightest edge connecting {B,C,D} to the rest of the vertices. That is, y must be <=80. Otherwise, by removing (BC) from T and adding (BH) we would get a tree with a smaller cost. Therefore, the following could be the values of z:

5 15 25 35 45 55 65 75

4. (30 points) Let A be a matrix of integers with N rows and M columns. We use A(i,j) to denote integer in row i and column j, where 1≤i≤N and 1≤j≤M. A path in the matrix is a sequence of integers from the matrix A(i_1, j_1), A(i_2, j_2), …, A(i_k, j_k), such that each next integer is to the **right** of the previous integer or **below** the previous integer in the matrix (1≤k). For example, in the matrix A shown below, A(1,1), A(2,1), A(2,2) (which is 7, 1, 2) is a path, and A(2,4), A(2,5), A(2,6), A(3,6) (which is 12, 5, 3, 2) is a path.

A:

7	4	25	16	9	4	11
1	2	1	12	5	3	-4
10	7	5	-3	8	2	8
4	8	7	10	4	6	17

A path is called **decreasing** if each next integer on the path is strictly less than the previous integer. For example, path A(2,4), A(2,5), A(2,6), A(3,6) (which is 12, 5, 3, 2) is decreasing, while path A(1,1), A(2,1), A(2,2) (which is 7, 1, 2) is not. The **length** of a path is the number of integers in the path. For instance, path 12, 5, 3, 2 has length 4. A path consisting of just one number has length 1.

The problem is to find the length of a longest decreasing path in a given matrix of integers and a sequence of integers in a longest path. In the above example, the length of a longest path is 6. There are several paths with length 6; one of them is 25, 16, 12, 5, 3, 2.

The input-output description of this problem is the following:

Input: A matrix A of integers with N rows and M columns.

Output:

1. The length L of a longest decreasing path in A, and
2. The longest (length = L) decreasing path in A (or one of them, if more than one exists).

Your task is to do the following:

a) Design a most efficient dynamic programming solution to this problem.
b) Write a detailed pseudocode of your solution.
c) Analyze in detail the running time of your algorithm.

Solution.

Subproblems: L(i,j) - the length of the longest decreasing path ending at A(i,j).

Recursive solution to subproblems:

Base case: L(1,1) = 1

Recursive cases:

For $1<j\leq M$:

$$L(1,j) = \begin{cases} L(1, j-1)+1 & \text{if } A(1, j-1) > A(1, j) \\ 1 & \text{otherwise} \end{cases}$$

For $1<i\leq N$:

$$L(i,1) = \begin{cases} L(i-1, 1)+1 & \text{if } A(i-1, 1) > A(i, 1) \\ 1 & \text{otherwise} \end{cases}$$

For $1<i\leq N$ and $1<j\leq M$:

$$L(i,j) = \begin{cases} \max\{L(i-1, j), L(i, j-1)\}+1 & \text{if } A(i-1, j) > A(i, j) \text{ and } A(i, j-1) > A(i, j) \\ L(i-1, j)+1 & \text{if } A(i-1, j) > A(i, j) \text{ and } A(i, j-1) \leq A(i, j) \\ L(i, j-1)+1 & \text{if } A(i-1, j) \leq A(i, j) \text{ and } A(i, j-1) > A(i, j) \\ 1 & \text{if } A(i-1, j) \leq A(i, j) \text{ and } A(i, j-1) \leq A(i, j) \end{cases}$$

The answer to the problem is obtained as $\max\{L(i,j): 1<i\leq N, 1<j\leq M\}$

In order to get the actual sequence of numbers of the longest decreasing path, we need to keep track of how values L(i,j) were obtained. For instance, we can have another matrix called prev to indicate whether the previous element in the sequence is above the current one, to the left of the current one, or there is no previous element.

Running time of the solution is O(NM).