COMP 3711 – Design and Analysis of Algorithms
2017 Fall Semester – Written Assignment # 3
Distributed: March 27, 2017 – Due: April 10, 2017
Revised (Q5 Diagram (C) fixed) April 8, 2017

Your solutions should contain (i) your name, (ii) your student ID #, and (iii) your email address
<u>Some Notes:</u>

- Please write clearly and briefly.
- Please follow the guidelines on doing your own work and avoiding plagiarism given on the class home page.
  In particular ***don't forget to acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.
- This assignment is due by 23:59 on April 10, 2017 in BOTH hard AND soft copy formats. A hard copy should be deposited in one of the two COMP3711 assignment collection boxes outside of room 4210. A soft copy for our records in PDF format should also be submitted via the online CASS system. See the Assignment 1 page in Canvas for information on how to submit online.
- The default base for logarithms will be 2, i.e., $\log n$ will mean $\log_2 n$. If another base is intended, it will be explicitly stated, e.g., $\log_3 n$.
- **In the original published version of the assignment, diagram (C) of Question 5 was missing one edge. That edge, $(0,5)-(1,5)$, has now been added in.**

1. **Optimal Binary Search Trees** (20 points)

   Consider the following input to the Optimal Binary Search Tree problem (it is presented in the same format as the example powerpoint file posted on the lecture page):

i	1	2	3	4	5	6	7	8
a_i	A	B	C	D	E	F	G	H
$f(a_i)$	5	15	5	5	10	10	20	5

   a) Fill in the two tables below. As in the example powerpoint only the entries with $i \leq j$ need to be filled in. We have started you off by filling in the $[i, i]$ entries.

i/j	1	2	3	4	5	6	7	8
1	5							
2		15						
3			5					
4				5				
5					10			
6						10		
7							20	
8								5

i/j	1	2	3	4	5	6	7	8
1	1							
2		2						
3			3					
4				4				
5					5			
6						6		
7							7	
8								8

Table 1: Left matrix is $e[i, j]$. Right matrix is $root[i, j]$.

   b) Draw the optimal Binary Search Tree (with 8 nodes) and give its cost.

2. **Edit Distance** (25 points)

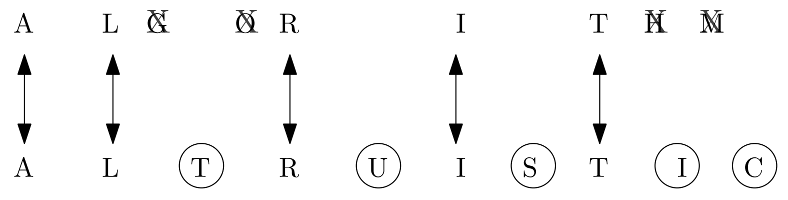
In this problem you must describe a dynamic programming algorithm for the *minimum edit distance* problem.

Background: The goal of the algorithm is to find a way to transform a "source" string $x[1 \ldots m]$ into a new "target" string $y[1 \ldots n]$ using any sequence of operations, *operating on the source string from left to right.* The *copy* operation copies the first remaining character in the source string to the target string, and deletes it from the source string. The *insert* operation adds one character to the end of the current target string. The *delete* operation deletes the first remaining character from the source string.

For example, one way to transform the source string `algorithm` to the target string `altruistic` is to use the following sequence of operations.

Operation	Target string	Source string
copy `a`	`a`	`lgorithm`
copy `l`	`al`	`gorithm`
delete `g`	`al`	`orithm`
insert `t`	`alt`	`orithm`
delete `o`	`alt`	`rithm`
copy `r`	`altr`	`ithm`
insert `u`	`altru`	`ithm`
copy `i`	`altrui`	`thm`
insert `s`	`altruis`	`thm`
copy `t`	`altruist`	`hm`
delete `h`	`altruist`	`m`
insert `i`	`altruisti`	`m`
delete `m`	`altruisti`	
insert `c`	`altruistic`	

The operations can be visualized as follows, with the bi-directional arrows signifying a copy, the circles an insert and the crosses a delete.



Each of the operations has an associated cost, c for copy, i for insert, and d for delete. The cost of a given sequence of transformation operations is the sum of the costs of the individual operations in the sequence. For the example above, the cost of converting `algorithm` to `altruistic` using the given set of operations is $5c + 5i + 4d$. If $c = 1$, $i = 2$ and $d = 3$ the cost of the edit would be $5 + 10 + 12 = 27$.

Given two sequences $x[1 \ldots m]$ and $y[1 \ldots n]$ and a given set of operation costs c, i, and d, the *minimum edit distance* from x to y is the cost of the —em least expensive transformation sequence that converts x to y.

(a) Define a cost array that you will use for the dynamic programming solution. Give the recurrence equation and describe why this equation is correct.

(b) Give the pseudocode for an algorithm for calculating the cost array. Document your pseudocode so that it is clear what each section is doing.

(c) Analyze the running time of the algorithm in part (b). State your results using $O()$ notation.

Hint 1: The algorithm is very related to the algorithm for the longest common subsequence problem taught in class.

Hint 2: Consider the last operation in the transformation from $x_i = x[1 \ldots i]$ *to* $y_j = y[1 \ldots ..j]$. *How does the cost of the entire transformation depend on the cost of the last operation?*

Comment: The minimum edit distance between two genes is one measure used in bioinformatics to determine how "close" two genes are to each other.

3. **Adjacency Lists** (15 points)

The class notes introduce the adjacency list representation for directed graphs $G = (V, E)$. That representation maintains an array $A[...]$ indexed by V, in which $A[v]$ is a linked list. The linked list holds the names of all the nodes u to which v points, i.e., nodes u for which $(v, u) \in E$. (Technically, $A[v]$ contains a *pointer* to the first item in the linked list).

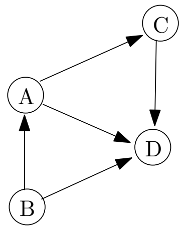
This is the default adjacency list format and can be thought of as an *out*-adjacency list representation

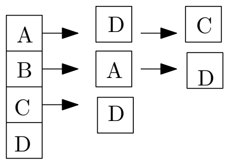
An *in*-adjacency list representation would be one in which $A[v]$ is the list of nodes that point *to* v.

The input to this problem is an *out*-adjacency list representation. Give pseudocode for an $O(|V|+|E|)$ algorithm that transforms the *out*-adjacency list representation into an *in*-adjacency list representation. Explain why your algorithm is correct and why it runs in $O(|V|+|E|)$ time.

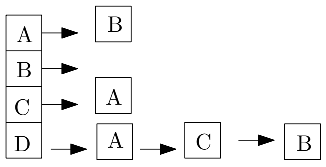
It is not necessary to write out all of the pointer details. You may assume that you have $O(1)$ *primitive operations* that allow creating a linked list and adding items to the head of the list.

The diagram below shows a directed graph and its associated out- and in-adjacency list representations.





Out-List



In-List

4. **Longest Paths in a DAG** (25 points)

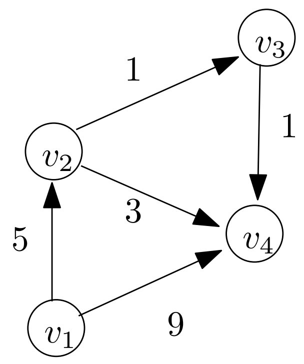
The input to this problem is a directed graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ satisfying the following property:

If $(v_i, v_j) \in E$, i.e., is an edge in the graph, then $i < j$.

Such a graph contains no cycles and is therefore known as a Directed Acyclic Graph (DAG). The edges in the graphs have associated lengths $\ell(i, j)$. The graph is inputted as a (in) adjacency list that has lengths stored with the edges.

The *length* of a *path* is the sum of the lengths of the edges on that path. For all $j > 1$ define $V[j]$ to be the length of the *longest* path in the graph from v_1 to v_j.

Here is an example weighted DAG and its associated $V[j]$ values:



j	1	2	3	4
$V[j]$	0	5	6	9

Give an $O(|V| + |E|)$ time algorithm for calculating all of the values $V[j]$. You may assume that all of the $\ell[i, j]$ are positive and that at least one path exists from v_1 to every v_j.

(a) Write a documented pseudocode description of your algorithm

(b) Explain why your algorithm is correct

(c) Explain why your algorithm runs in $O(|V| + |E|)$ time

Hint: Use Dynamic Programming

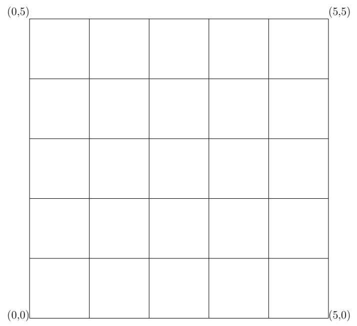
5. **Depth and Breadth First Search** (15 points)

In this problem you will have to describe the depth and breadth first search trees calculated for particular graphs. Let graph G_n be the $n \times n$ grid containing the n^2 points (i,j), $i = 0,\ldots,n-1$, $j = 0,\ldots,n-1$. Figure (A) illustrates G_6. Each point is connected to four neighbors: the one to its right, the one above it, the one to its left, and the one below it. Note that some points only have two or three neighbors, e.g., the four corner points only have 2 neighbors and that the edge points (aside from the corners) each have three neighbors. The adjacency list representation used is

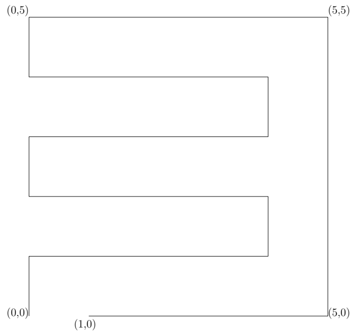
$$(i,j) :\rightarrow (i+1,j) \rightarrow (i,j+1) \rightarrow (i-1,j) \rightarrow (i,j-1).$$

For example, the adjacency list representation for $(1,1)$ in G_6 is

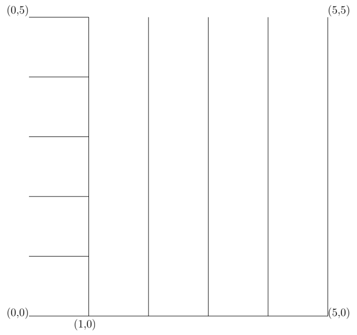
$$(1,1) :\rightarrow (2,1) \rightarrow (1,2) \rightarrow (0,1) \rightarrow (1,0).$$



(A)



(B)



(C)

Some nodes will only have two or three neighbors; their adjacency lists should be adjusted appropriately. For example

$$(0,0) :\rightarrow (1,0) \rightarrow (0,1) \quad \text{and} \quad (i,0) :\rightarrow (i+1,0) \rightarrow (i,1) \rightarrow (i-1,0)$$

for $i = 1,\ldots,n-2$.

In what follows *Describe the tree* means (i) list the edges in the tree and (ii) sketch a diagram that illustrates how the tree looks.

(a) Describe the tree produced by Depth First Search run on G_n, for all $n \geq 6$, starting at root $s = (1,0)$. Figure (B) illustrates the tree produced for G_6.

(b) Describe the tree produced by Breadth First Search run on G_n, for all $n \geq 6$, starting at root $s = (1,0)$. Figure (C) illustrates the tree produced for G_6.

Hint: experiment by drawing the BFS and DFS trees for $n = 5, 6, 7$. You should see a pattern.