COMP 3711 – Design and Analysis of Algorithms
2021 Fall Semester – Written Assignment # 4
Distributed: November 4, 2021
Due: November 18, 2021, 11:59 PM
Updated: November 12, 2021

Your solution should contain
(i) your name, (ii) your student ID #, and (iii) your email address
at the top of its first page.

Some Notes:

- Please write clearly and briefly. Your solutions should follow the guidelines given at
  https://canvas.ust.hk/courses/38226/pages/assignment-submission-guidelines

  In particular, your solutions should be written or printed on *clean* white paper with no watermarks, i.e., student society paper is not allowed.

- Please also follow the guidelines on doing your own work and avoiding plagiarism as described on the class home page.
  You must acknowledge individuals who assisted you, or sources where you found solutions. Failure to do so will be considered plagiarism.

- The term *Documented Pseudocode* means that your pseudocode must contain documentation, i.e., comments, inside the pseudocode, briefly explaining what each part does.

- Many questions ask you to explain things, e.g., what an algorithm is doing, why it is correct, etc. To receive full points, the explanation must also be *understandable* as well as correct.

- Please make a *copy* of your assignment before submitting it. If we can't find your submission, we will ask you to resubmit the copy.

- Submit a SOFTCOPY of your assignment to CASS (not Canvas) by the deadline. The softcopy should be one PDF file (no word or jpegs permitted, nor multiple files).

  If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possibly denser.

- November 12, 2021. Note added at end of Problem 2, clarifying that $P(1,1)$ is always included in the value of the path.

Problem 1: [25pts] Longest Common Subsequence of Three Sequences.

In class you learned a DP algorithm for finding the longest common subsequence of two sequences. In this problem you need to extend that algorithm to three sequences.

More specifically, let

$$W = (w_1, \ldots, w_p), \quad X = (x_1, \ldots, x_m), \quad Y = (y_1, \ldots, y_n)$$

be three given sequences. $Z = (z_1, \ldots, z_q)$ is a *common subsequence* of W, X, Y if $z_t = w_{i_t} = x_{j_t} = y_{k_t}$ for all $t = 1, \ldots, q$ where

$$i_1 < i_2 \cdots < i_q, \quad j_1 < j_2 < \cdots < j_q, \quad k_1 < k_2 < \cdots < k_q. \tag{1}$$

The problem is to find the length of the longest common subsequence of the three sequences, i.e., the largest possible such q.

As an example, an LCS of $W = gahbczha$, $X = zargbrca$ and $Y = azbrcgar$, is $Z = abca$,

Define $c[i, j, k]$ to be the length of the longest common subsequence of $W[1 \ldots i]$, $X[1 \ldots j]$ and $Y[1 \ldots k]$.

Your goal is to calculate $c[p, m, n]$.

(a) Write down a recurrence relation for $c[i, j, k]$. It should be written in the form

$$c[i,j,k] = \begin{cases} ??????? & \text{if } ????????????? \\ ??????? & \text{if } ????????????? \\ \vdots & \vdots \end{cases}$$

Your recurrence should include the initial conditions.

(b) Derive the correctness of the recurrence relation for part (a).

(c) Give documented pseudocode for an algorithm for calculating $c[p, m, n]$, based on your recurrence relation from (a).

(d) Analyze the running time of your algorithm. For full marks, your algorithm should run in $O(pmn)$ time.

Problem 2 [25 pts] Prize Collecting Walks on a Grid

The input to this problem is an $m \times n$ grid. The top left entry is $(1,1)$ the bottom right entry is (m,n). Entry (i,j) in the grid contains a prize or penalty with known worth $P(i,j)$ (the input).

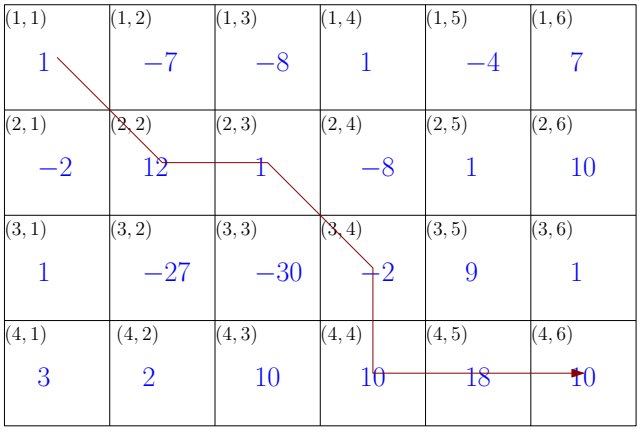
A *walk* starts at $(1,1)$ and terminates at (m,n). At each step you can either move one unit to the right or one unit down or one unit to the right and one unit down. That is, from (i,j) you may move to either $(i,j+1)$, $(i+1,j)$ or $(i+1,j+1)$ (if you are on the boundary and only one such neighbor exists, you must move to that existing neighbor).

When you land on (k,r) you collect or pay money. How much you collect/pay depends on both $P(k,r)$ and how you got to (k,r).

If you move from (i,j) to $(i+1,j)$ or $(i,j+1)$ you will earn, respectively, $P(i+1,j)$ or $P(i,j+1)$. But, if you move to $(i+1,j+1)$, you will earn $\frac{3}{2}P(i+1,j+1)$.

The *Value* of a walk is the sum of all the prizes/penalties that you have collected/paid. Your goal is find the largest possible value walk. It is not necessary to find the walk itself; just the value.

As an example, consider the grid and walk shown below



This walk has value $1 + \frac{3}{2}12 + 1 + \frac{3}{2}(-2) + 10 + 18 + 10 = 55$ which is a max value walk.

Design an $O(mn)$ time dynamic programming algorithm for finding a maximum-value walk.

(a) Give the recurrence upon which your DP algorithm is based. Before giving the recurrence, define (using english and math symbols) the meaning of each entry.

Don't forget to completely specify the initial conditions of the recurrence relation.

(b) Explain how, after filling in your table, you can use the information in your table to solve the problem.
Note: This is meant to have a one line answer. Is the solution a particular table entry? If so, which entry? Or is it a function, e.g., the maximum, of a set of table entries?

(c) Prove the correctness of the recurrence relation from part (a).

(d) Give documented psuedocode for your algorithm. Your code only needs to find the *value* of a maximum-value walk, not the actual walk itself.

(e) Explain why your algorithm runs in $O(nm)$ time.

Added November 12, 2021. Although this was illustrated in the example, the original definition of the problem did not make it clear that the value of the path always includes $P(1,1)$.

To make this consistent you should assume that the walk really always starts at (imaginary) vertex $(1,0)$ *and the first step is always a move from* $(1,0)$ *to* $(1,1)$. *That will make it clear that the value* $P(1,1)$ *is always included in the value of the path.*

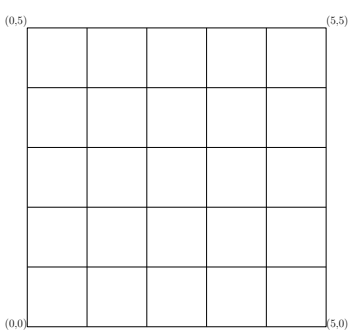
Problem 3 [20 pts] Depth and Breadth First Search

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

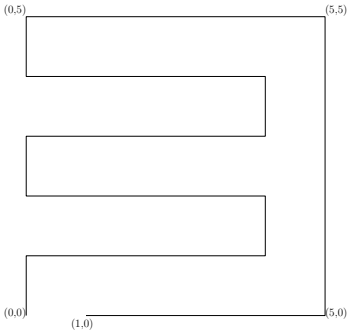
$$(i, j) :\rightarrow (i+1, j) \rightarrow (i, j+1) \rightarrow (i-1, j) \rightarrow (i, j-1).$$

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

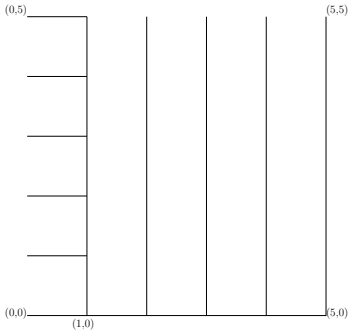
$$(1, 1) :\rightarrow (2, 1) \rightarrow (1, 2) \rightarrow (0, 1) \rightarrow (1, 0).$$



(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 $n = 5, 6, 7, 8$. You should see a pattern.

Problem 4 **[20 pts]** Implicit Graphs

Many graph problems are not actually presented as graph problems. Graphs are only introduced as part of the solution technique. Also, the graphs are often not given explicitly but are only implicitly defined.

Consider the following "3-jug" problem. You are given 3 jugs of known capacities. The first jug holds 8 liters, the second 5 liters, the third 3 liters.

- The 8-liter jug is filled with water; the other two are empty.
- You can pour water from any jug to any other jug, stopping when the pouring jug is empty or the poured into jug is full (whichever comes first)
- You have no other way of measuring water

The problem is usually given as "Is there any sequence of water-pouring that would terminate with the water equally divided so that there are 4 liters each in the first and second jug?".

You will solve this and other problems using BFS and DFS.

Consider the directed graph $G_8 = (V, E)$ defined as follows

- Vertices represent possible configurations of 8 liters in the bottles (all numbers are nonnegative integers)

$$V = \big\{(a, b, c) \,:\, a + b + c = 8,\ 0 \le a \le 8,\ 0 \le b \le 5,\ 0 \le c \le 3\big\}.$$

- Let $v_1 = (a_1, b_1, c_1)$ and $v_2 = (a_2, b_2, c_2)$ where $v_1 \neq v_2$ and $v_1, v_2 \in V$. Edge $v_1 \to v_2$ *exists* if it is possible to legally get from configuration v_1 to configuration v_2 by pouring water from one jug to the other, i.e., one of the following 6 moves occurs:

(i) $(a_2, b_2, c_2) = (a_1 - p, b_1 + p, c_1)$ where $p = \min(a_1, 5 - b_1)$
(ii) $(a_2, b_2, c_2) = (a_1 - p, b_1, c_1 + p)$ where $p = \min(a_1, 3 - c_1)$
(iii) $(a_2, b_2, c_2) = (a_1 + p, b_1 - p, c_1)$ where $p = \min(b_1, 8 - a_1)$
(iv) $(a_2, b_2, c_2) = (a_1, b_1 - p, c_1 + p)$ where $p = \min(b_1, 3 - c_1)$
(v) $(a_2, b_2, c_2) = (a_1 + p, b_1, c_1 - p)$ where $p = \min(c_1, 8 - a_1)$
(vi) $(a_2, b_2, c_2) = (a_1, b_1 + p, c_1 - p)$ where $p = \min(c_1, 5 - b_1)$

- The adjacency list of v should be ordered as (i), (ii), (iii), (iv), (v), (vi), only including the edges that occur. For example, the adjacency list of $v = (2, 4, 2)$ is

$$(2, 4, 2) \,:\, (1, 5, 2),\ (1, 4, 3),\ (6, 0, 2),\ (2, 3, 3),\ (4, 4, 0),\ (2, 5, 1)$$

while the adjacency list of $v = (1, 5, 2)$ is

$$(1, 5, 2) \,:\, (0, 5, 3),\ (6, 0, 2),\ (1, 4, 3),\ (3, 5, 0).$$

Recall that "v' is reachable from v" if there is a path from v to v' in G_8.

Let $v_0 = (8, 0, 0)$ be the vertex corresponding to the configuration that has all of the water in one jug,

Now answer all of the following questions:

(A) Draw the BFS tree that results from running BFS starting at v_0.
Note that this tree is rooted at v_0.
You only need to draw the tree of all items that are reachable from v_0. This might or might not be the entire graph.

(B) Draw the DFS tree that results from running DFS starting at v_0.
Again, note that this tree is rooted at v_0 and you only need to draw the tree of all items that are reachable from v_0. This might or might not be the entire graph.

(C) Start with 8 liters of water in the jug of capacity 8.
Is there any sequence of water-pouring that would terminate with 4 liters each in the first and second jug?

If there is such a way, describe a method to achieve this that uses the least number of pouring steps.

(D) Justify the correctness of your answer to part (C) using the answers to parts (A) and/or (B).

(E) Start with 8 liters of water in the jug of capacity 8.
Is there any sequence of water-pouring that would terminate with 6 liters in the first jug and 1 liter each in the second and third jugs.

If there is such a way, describe a method that uses the least number of pouring steps.

(F) Justify the correctness of your answer to part (E) using the answers to parts (A) and/or (B).

(G) Prove or disprove the following statement:
For all $v \in V - \{v_0\}$,
if v is reachable from v_0 then v_0 is reachable from v.

(H) Prove or disprove the following statement:
For all $v \in V - \{v_0\}$,
if v_0 is reachable from v then v is reachable from v_0

Note: (C) and (E) are standard puzzle questions. For (D) and (F) you must either prove that your method uses the minimal number of pours or explain how you know that no method exists. (G) and (H) are to help you understand if the problem is "reversible" and how that translates into graph language.

Problem 5 [10 points] Optimal Binary Search Trees

Consider the following input to the Optimal Binary Search Tree problem (it is presented in the same format as in 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

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

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

Right matrix is $root[i, j]$.

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