Warm-Up Homework #0

CS 4120-5120 — Due: Wed, Jan 17, 2024
Spring 2024 — 30 points

1. [4 pts.] Suppose your programming language does not support multi-dimensional arrays, and so you have decided to simulate a 2-dimensional array. Let us assume all array indexes normally start at 1, not 0. For example, suppose you have an array A of 2 rows and 4 columns, which you think of as

A[1,1] A[1,2] A[1,3] A[1,4]
A[2,1] A[2,2] A[2,3] A[2,4]

This two-dimensional (2D) array is actually in a linear array B, so the elements of the first row are stored first, and the elements of the second row are stored next, and so on. [This is called "row-major order".] In other words,

A[1,1] A[1,2] A[1,3] A[1,4] A[2,1] A[2,2] A[2,3] A[2,4] are stored in:
B[1] B[2] B[3] B[4] B[5] B[6] B[7] B[8] respectively.

Thus, you simulate the 2D array by mapping the pair of indexes into the single index of the linear array B, which actually stores the data. For example, A[1,2] is stored in B[2] and A[2,3] is stored at B[7]. Assuming that A has R rows and C columns, devise a formula for k such that B[k] stores A[i,j], given the row index i and column index j. [**Hint**: Think about it - your formula for k would obviously depend on i and j, but could *possibly* depend on R and C, as well! It is ALWAYS important to get a feel for the "parameters" of a formula, i.e., variables that comprise the formula.]

2. [3 pts.] Do the same as in #1, i.e., assuming that A has R rows and C columns, devise a formula for k such that B[k] stores A[i,j], given the row index i and column index j, but this time assuming that the elements of A are stored in B in "column major" order. As an example, for the 2x4 matrix A in the above example, the order will be as follows:

A[1,1] A[2,1] A[1,2] A[2,2] A[1,3] A[2,3] A[1,4] A[2,4] are stored in:
B[1] B[2] B[3] B[4] B[5] B[6] B[7] B[8] respectively.

3. [4 pts.] Suppose we have a distance chart as in a map. Such a chart shows the distance for all pairs of cities (assuming the distance is the same in either direction between a given pair of cities). If we have a total of n cities, we just need to show (n-1) distances from the first city (city #1), but only (n-2) distances from the second city (city #2) to all but the first city (i.e., city #3 to city #n)! (That is because the distance from the first city to the second has already been recorded for the first city, and we have already assumed that the distance is the same in either direction.) Then, there will be one fewer distance to record for each of the remaining cities, with no distance left to be recorded for the last city! Also, in this specially-shaped distance array A, due the symmetry of distances discussed above, the distance A[i,j] from city i to city j and the distance A[j,i] from city j to city i will be the same, and will actually be stored corresponding to A[i,j] if i < j, or corresponding to A[j,i] if j < i.

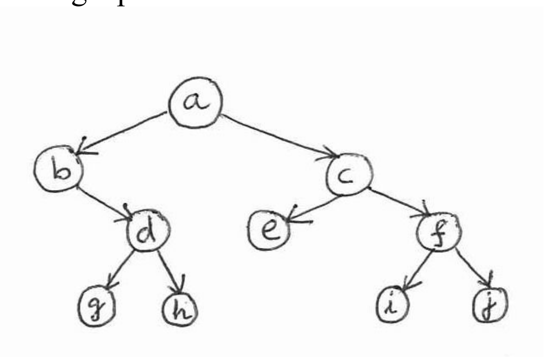
a. Suppose the distance chart A for n cities were to be stored in a single-dimensional array B. Clearly, A[1,2] through A[1,n] will be stored in B[1], B[2], … all the way through B[n-1], then A[2,3] through A[2,n] will be stored next in B[n], B[n+1],… all the way through B[2n-3], and so on. Assuming there are n cities, devise a formula for k such that the distance A[i,j] is stored in B[k].
b. Certain values or combinations of values are invalid for i and j. Obviously, any combination with either i or j greater than n is invalid! Explain why a combination where i and j are equal is also not valid.
c. For a total of n cities, give a formula for how many distances need to be stored.

4. [3 pts.] Suppose you perform the following operations on a stack. What will be the output?

Push 1
Push 2
Push 3
Push 4
Pop and output the popped item
Pop and output the popped item
Push 5
Push 6
Pop and output the popped item
Pop and output the popped item
Pop and output the popped item

At this point is there an overflow or underflow on the stack? Are there any items left over on the stack?

5. [3 pts.] Perform a post-order traversal on the following binary tree, i.e., show the order in which the nodes are visited during a post-order traversal:



6. [4 pts.] Keeping in mind that the following traversal strings both correspond to the same tree, construct the single binary tree that gives rise to both the following pre-order and in-order traversal strings. Note that each letter corresponds to a node. You may approach this problem in a “brute-force” or trial-and-error manner, but an algorithmic approach might be more rewarding!

Pre-order traversal: a b d c e g h f i
In-order traversal: d b a g e h c i f

7. [4 pts.] For n>=0, prove the following, preferably using mathematical induction:

$$\sum_{i=1}^{n} i = n(n+1)/2$$

8. Can you devise an algorithm for the task mentioned in problem #6? (A yes/no answer is acceptable this time, although normally it would be considered a “smart-aleck” response and therefore unacceptable. A good answer might possibly make up for little slips in the rest of the homework.)