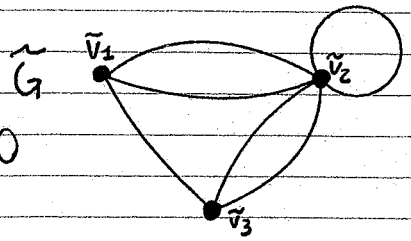
Michael Ilyin Worksheet 1-27

1.

a)




10/10

b)

$$A_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array}\begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 0 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \end{array}, \quad A_{\tilde{G}} = \begin{array}{c} \\ \tilde{v}_1 \\ \tilde{v}_2 \\ \tilde{v}_3 \end{array}\begin{array}{c} \begin{array}{ccc} \tilde{v}_1 & \tilde{v}_2 & \tilde{v}_3 \end{array} \\ \begin{bmatrix} 0 & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & 0 \end{bmatrix} \end{array}, \quad (A_G)^T = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array}\begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array}$$

$$A_G + (A_G)^T = \begin{bmatrix} 0 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & 0 \end{bmatrix} = A_{\tilde{G}}$$

This sum does hold for any general diagraph $G$ with supporting graph $\tilde{G}$. In the matrix $A_G$ the $A_G[i,j]$ entry represents the number of edges between the vertex $V_i$ and the vertex $V_j$ in which $V_i$ is the initial vertex. In the matrix $(A_G)^T$, the $(A_G)^T[i,j]$ entry becomes the representation for the number of edges between the vertex $V_i$ and the vertex $V_j$, in which $V_i$ is the terminal vertex. In the $(i,j)$ entry of the matrix $A_G + (A_G)^T$, the number will represent the sum of the number of edges between $V_i$ and $V_j$ in which $V_i$ is the initial vertex, and in which $V_j$ is the initial vertex. This sum is equal to just the number of undirected edges between the vertex $V_i$ and $V_j$ in the underlying supporting general graph, $\tilde{G}$. In the matrix $A_{\tilde{G}}$ the $(i,j)$-th entry represents exactly this number of edges between $V_i$ and $V_j$. This is why every $(i,j)$-th entry of the matrix $A_G + (A_G)^T$ will be equal to the $(i,j)$-th entry of the matrix $A_{\tilde{G}}$. ☐ good!

c) Number of walks of length 3 from $V_1$ to $V_3$ = $(1,3)$ entry of $A_G^3$

$$A_G = \begin{bmatrix} 0 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad A_G^2 = \begin{bmatrix} 0 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 2 \\ 0 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$A_G^3 = \begin{bmatrix} 0 & 3 & 2 \\ 0 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}\begin{bmatrix} 0 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 3 \\ 0 & 3 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

$A_G^3[1,3] = 3$ ✓ Therefore, there are 3 walks of length 3 from $V_1$ to $V_3$ in the graph $G$

d)

| G: d(u, v) | $V_1$ | $V_2$ | $V_3$ |
|---|---|---|---|
| $V_1$ | 0 | 1 | 1 |
| $V_2$ | +∞ | 0 | 1 |
| $V_3$ | +∞ | 1 | 0 |

✓

| $\tilde{G}$ d(u, v) | $V_1$ | $V_2$ | $V_3$ |
|---|---|---|---|
| $V_1$ | 0 | 1 | 1 |
| $V_2$ | 1 | 0 | 1 |
| $V_3$ | 1 | 1 | 0 |

✓

2. For a general graph G, the graph distance d, as defined, satisfies the properties:

i) $d(x,y) \geq 0$, with $d(x,y) = 0$ iff $x = y$

because if $d(x,y) = 0$ then the shortest path between the vertex $x$ and the vertex $y$ is of length 0. This means that the path starts at the vertex $x$, traverses no edges, and ends at the same vertex. Thus for a path to start at a vertex $x$ and end at a vertex $y$, and have a length of 0, the vertex $x$ and the vertex $y$ must be the same vertex. Hence, $x = y$. ⤻

If the vertex $x$ and the vertex $y$ are the same vertex, then the shortest $x$-$y$ path would start at the vertex $x$ and end at the same vertex, $y$, traversing 0 edges. Thus the shortest $x$-$y$ path has length 0. Hence, $d(x,y) = 0$.

Now, if the vertex $x$ and the vertex $y$ are NOT the same vertex, then the shortest $x$-$y$ path can be at least length 1. Thus $d(x,y) \geq 1$. This inequality along with $d(x,y) = 0$ iff $x = y$, means that $d(x,y) \geq 0$. ✓

ii) $d(x,y) = d(y,x)$, for any vertices $x, y$.

Since the graph G is not a digraph, the edges between vertices in G do not have direction. Thus, if $d(x,y) = k$, then the shortest $x$-$y$ path has length $k$. That path can be represented as $x = v_0, e_0, v_1, e_1, \ldots, v_{k-1}, e_{k-1}, v_k = y$. Now, a path containing those same edges and vertices will also be the shortest $y$-$x$ path. This path will now be $y = v_k, e_{k-1}, v_{k-1}, \ldots, e_1, v_1, e_0, v_0 = x$, having also length $k$. Thus, $d(y,x) = k$. Hence, $d(x,y) = k = d(y,x)$. Therefore, $d(x,y) = d(y,x)$ for any vertices $x, y$.

iii) $d(x,z) \leq d(x,y) + d(y,z)$, for any vertices $x, y, z$.

If $d(x,z) = k$, then the shortest $x$-$z$ path has length $k$. If the vertex $y$ is already present in this $x$-$z$ path, then this $x$-$z$ path can be represented as: $x = v_0, e_0, v_1, e_1, \ldots, v_i, e_i, \ldots, v_{k-1}, e_{k-1}, v_k = z$, where $v_i = y$. $d(x,y)$ would then equal $i$ and $d(y,z)$ would equal $k - i$. Hence $d(x,y) + d(y,z) = i + (k-i) = k = d(x,z)$. Thus $d(x,z) = d(x,y) + d(y,z)$. Now, if the vertex $y$ is not already present in the shortest $x$-$z$ path, then $d(x,y) = d(x,v_i) + d(v_i,y)$ and $d(y,z) = d(y,v_i) + d(v_i,z)$. ← who is $v_i$ now?

Now $d(x,v_i) = i$ and $d(v_i,z) = k - i$, and let $d(v_i,y) = d(y,v_i) = l$. $l$ cannot be zero since $v_i$ cannot be $y$. Thus, $d(x,y) = i + l$ and $d(y,z) = l + k - i$.

Hence, $d(x,y) + d(y,z) = (i+l) + (l+k-i) = 2l + k$.

Since $2l + k > k = d(x,z)$, $d(x,y) + d(y,z) > d(x,z)$.

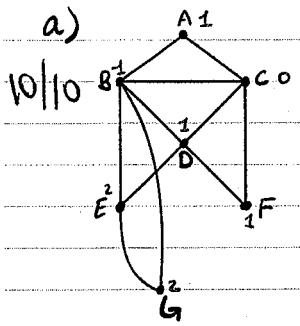
Therefore $d(x,z) \leq d(x,y) + d(y,z)$.

you should also consider the case $d(x,y) = +\infty$

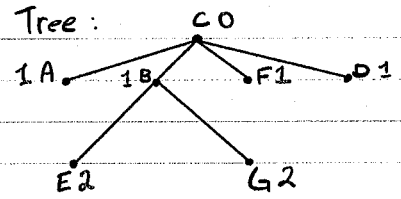
not accurate

3.

a) 10/10




Seach Tree:




b) Different (correct) answers are possible for part (a) because of the way that the BFS algorithm is structured. Each search tree, though not isomorphic will still show the distance from the initial vertex, in this case C, to any other vertex in the graph. This is because many vertices can be adjacent to a vertex $v_k$, but in the search tree that vertex $k$ must be adjacent to just one of those many vertices. The distance between the initial vertex and that vertex $v_k$ will have the same representation no matter the choice of adjacency in the tree and thus the distance will stay the same.

4. Modified BFS Algorithm:

Input: An unlabeled graph $G = (V, E)$ with distinguished vertex $x$.

Output: The length of the smallest cycle starting at the vertex $x$.

Method: Use a variable $i$ to measure the distance from $x$, and label vertices with $i$ as their distance is found, as well as check the adjacencies of labeled vertices to locate a cycle.

10+2

1. $i \leftarrow 0$
2. Label $x$ with "$i$"
3. Find all unlabeled vertices adjacent to at least one vertex with label $i$. If none are found, stop because we have reached all possible vertices and conclude that no cycle, starting at the vertex $x$, exists in the graph.
4. Label all vertices found in step 3 with $i+1$.
5. If any vertex labeled with $i+1$ is adjacent to two different vertices, labeled with $i$, Stop. Conclude that the smallest cycle, starting at the vertex $x$, has length $2(i+1)$.
6. If any vertices labeled with $i+1$ are adjacent, Stop. conclude that the smallest cycle, starting at the vertex $x$, has length $2(i+1)+1$.
7. $i \leftarrow i+1$
8. Repeat step 3.

GREAT!

In order to list the smallest cycle, starting with the vertex $x$, (if one even exists) there are two cases.

Case 1: If the algorithm Stops at 4. List the shortest path from the vertex $x$, to one of the two vertices, labeled $i+1$, then list the edge that is incident with both of the vertices labeled with $i+1$. Proceed by listing the shortest path from the other vertex labeled $i+1$, to the vertex $x$. This list will be the list that represents the shortest cycle, starting at the vertex $x$.

Case 2: If the algorithm concludes at step 5. List the shortest path from the vertex $x$ to one of the vertices, labeled $i$, that is adjacent to the vertex labeled $i+1$. Continue by listing the edge that is incident with both that chosen vertex, labeled $i$, and the vertex labeled $i+1$. Next list that vertex that is labeled $i+1$. Proceed by listing the edge that is incident with the vertex, labeled $i+1$, and the second vertex, labeled $i$. Finally, list the shortest path from that second vertex, labeled $i$, and the vertex $x$. This list will be the list that represents the smallest cycle, starting at the vertex $x$.

The procedure in either of the two cases, listed above, will create a list of the smallest cycle in the graph $G$, that starts at the vertex $x$. (If such a cycle even exists)

↳ Yes, but how do you actually implement this so that the computer can list those paths?