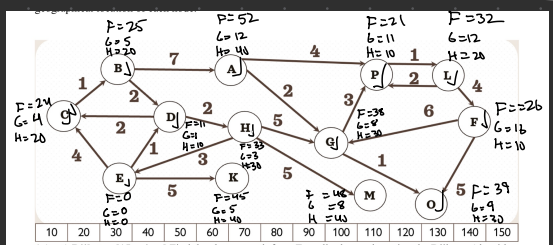
C: E, D, C = 3
D: E, D = 1
H: E, D, H = 3
B: E, D, G, B = 4
K: E, K = 5
M: E, D, H, M = 8
G: E, D, H, G = 8
O: E, D, H, G, O = 9
A: E, D, G, B, A = 11
P: E, D, H, G, P = 11
L: E, D, H, G, P, L = 12
F: E, D, H, G, P, L, F = 16

1.

Curr	Prev
E	null
D	E
H	D
C	D
B	C
E	K
M	H
G	H
O	G
A	B
P	A
L	P
F	L

Visited: E, D, H, C, B, K, M, G, O, A, P, L, F ✅ cust

2.



Visited: E, D, C, B, H, G, P, L, F, G, M, A

3. Both algorithms found H in 4 iterations therefore they are the same.

4. The time complexity of this code is O(n²) because we use 2 nested for loops, one to read the nxn matrix, and the other to check symmetry of matrix which is n^2 in worst case.
So; $O(n^2) + O(n^2) = O(n^2)$. where n is the size of the matrix.
The space complexity for this code is n^2 where n is size of the matrix because, the nxn matrix takes up O(n²) memory.

5. Time complexity is O(n) where n is the number of vertices, because I use for loops for this codes and none of them are nested so its all O(n).
The space complexity of this code is also O(n) where n is the # of vertices because I store O(n) elements in the graph I am printing out.

6. The time complexity is O(V+E) where V is the vertice, and E is the edge because in the worst case we go through every single vertice and edge in the graph using dfs.
The space complexity is also O(V+E) where V is the vertice and E is the edge because we store each of the edges at least once which = O(V+E).