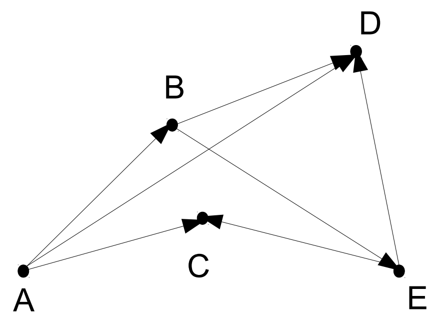
1) For the given graph: a) give it's adjacency matrix representation and adjacency list representation (using either Python's lists or Python's dictionaries).



b) in-degree of D =

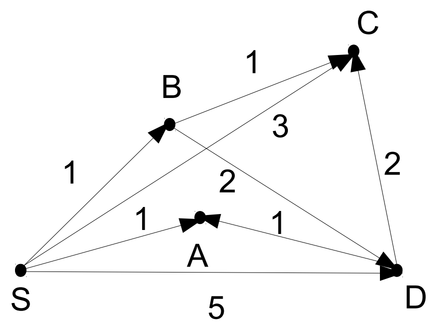
in-degree of A =

out-degree of B =

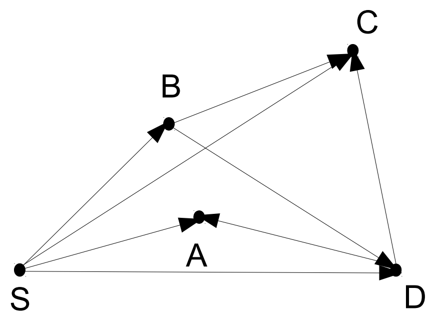
out-degree of C =

c) Does it have cycles? If yes, list them.

2) For the following weighted directed graph, give its adjacency list representation



3) For the given unweighted directed graph, use the unweighted shortest path algorithm (**BFS**).



queue: S
v:

	S	A	B	C	D
parent	None	None	None	None	None
distance	0				

queue:
v:

	S	A	B	C	D
parent	None				
distanc e	0				

queue:
v:

	S	A	B	C	D
parent	None				
distance	0				

queue:
v:

	S	A	B	C	D
parent					
distance					

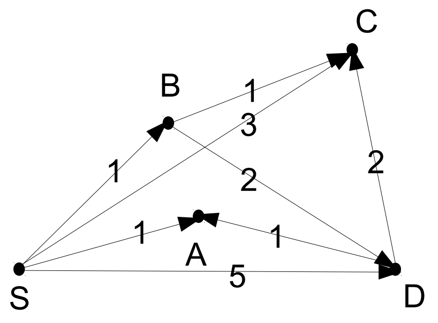
queue:
v:

	S	A	B	C	D
parent					
distance					

Give the path from S to C:

Give the path from S to D:

4) For the given weighted directed graph, use the weighted shortest path algorithm (**Dijkstra's**).



priority queue: S, A, B, C, D

	S	A	B	C	D
parent	None	None	None	None	None
distance	0	infty	infty	infty	infty

priority queue:

	S	A	B	C	D
parent					
distance					

priority queue:

	S	A	B	C	D
parent					
distance					