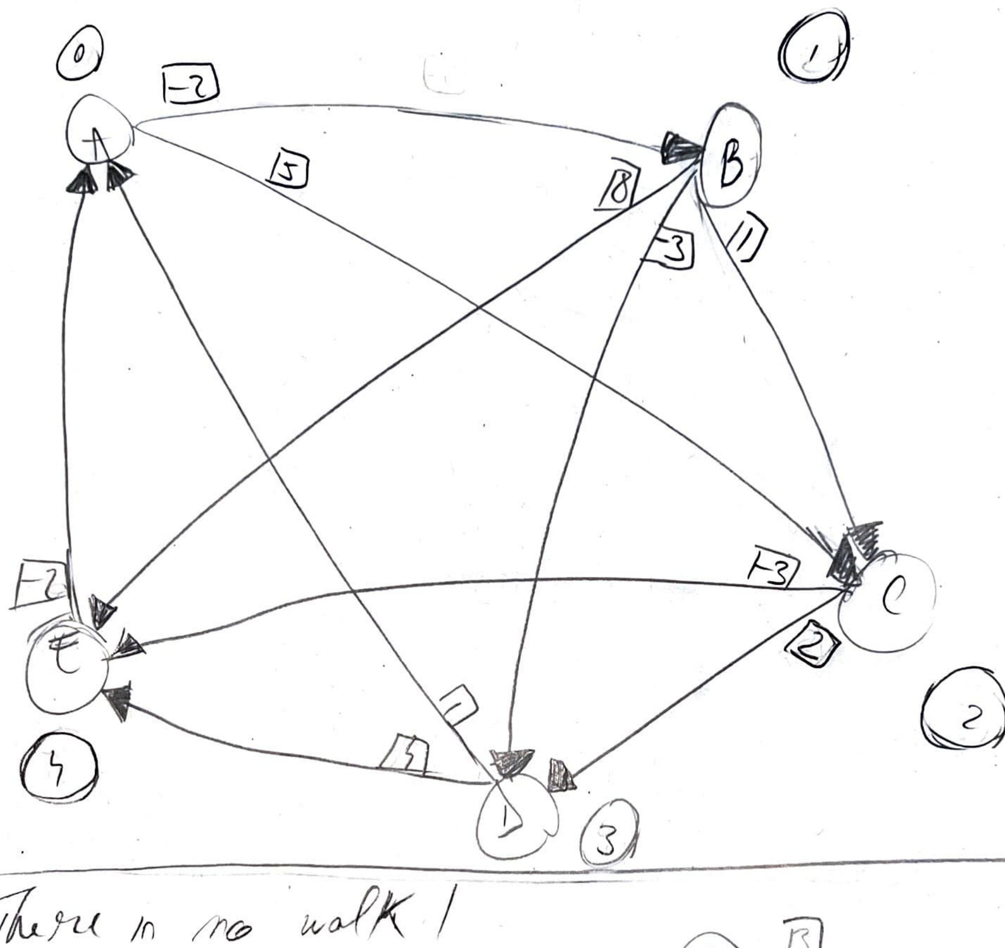
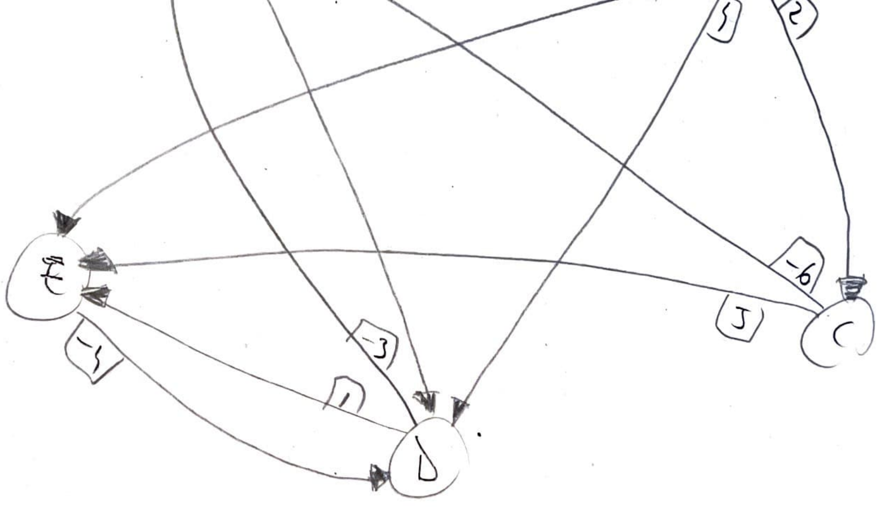
Practical Work nr 3.

5) Given a graph with costs and two vertices finds a lowest cost walk between the given vertices, or prints a message if there are negative cycles accessible from the starting vertex. Use the Ford's algorithm.

There is a walk



There is no walk!



Note	(A->E) changed	edge (x,y)	distance dictionary A	B	C	D	E	predecessors dictionary A	B	C	D	E
initialization			0	∞	∞	∞	∞	-1	-1	-1	-1	-1
iteration 1	T	(A,B)	0	-2	∞	∞	∞		A			
	T	(A,C)	0	-2	5	∞	∞			A		
	T	(B,C)	0	-2	-1	∞	∞			B		
	T	(B,D)	0	-2	-1	-5	∞				B	
pană aici fără linii	T	(B,E)	0	-2	-1	-5	6					B
	F	(C,D)	0	-2	-1	-5	6					
	T	(C,E)	0	-2	-1	-5	-4					C
F -> contrar schimba dist la A, ținem cont, o [illegible]	F	(D,A)	0	-2	-1	-5	-4					
	F	(D,E)	0	-2	-1	-5	-4					
	F	(E,A)	0	-2	-1	-5	-4	-1	A	B	B	C
iteration 2	F	(A,B)	0	-2	-1	-5	-4					
	F	(A,C)	0	-2	-1	-5	-4					
	F	(B,C)	0	-2	-1	-5	-4					
	F	(B,D)	0	-2	-1	-5	-4					
	F	(B,E)	0	-2	-1	-5	-4					
	F	(C,D)	0	-2	-1	-5	-4					
	F	(C,E)	0	-2	-1	-5	-4					
	F	(D,A)	0	-2	-1	-5	-4					
	F	(D,E)	0	-2	-1	-5	-4					
	F	(E,A)	0	-2	-1	-5	-4	-1	A	B	B	C

!!

Nu s-a mai updatat => the lowest cost path between A and E is equal to distance[E] = -4.

The path leading to E from A, with the lowest cost is:

current_vertex = E => path = [E]

current_vertex = predecessor[E] = C => path = [E, C]

current_vertex = predecessor[C] = B => path = [E, C, B]

current_vertex = predecessor[B] = A => path = [E, C, B, A]

current_vertex = predecessor[A] = -1 => break

Node (A-S)	changed	edge(x,y)	distance dictionary A	B	C	D	E	predecessors dictionary A	B	C	D	E
Initialization			0	∞	∞	∞	∞	-1	-1	-1	-1	-1
iteration 1	T	(A,B)	0	3	∞	∞	∞		A			
	T	(A,D)	0	3	∞	2	∞				A	
	T	(B,C)	0	3	5	2	∞			B		
	F	(B,D)	0	3	5	2	∞					
	T	(B,E)	0	3	5	2	2					B
	F	(C,A)	0	3	5	2	2					
	F	(C,E)	0	3	5	2	2					
	F	(D,A)	0	3	5	2	2					
	F	(D,E)	0	3	5	2	2					
	T	(E,D)	0	3	5	-2	2					D
									A	B	A	D
iteration 2	F	(A,B)	0	3	5	-2	2					
	F	(A,D)	0	3	5	-2	2					
	F	(B,C)	0	3	5	-2	2					
	F	(B,D)	0	3	5	-2	2					
	F	(B,E)	0	3	5	-2	2					
	F	(C,A)	0	3	5	-2	2					
	F	(C,E)	0	3	5	-2	2					
	F	(D,A)	0	3	5	-2	2					
	T	(D,E)	0	3	5	-2	-1					D
	T	(E,D)	0	3	5	-5	-1				E	
iteration 3			0									
			0									
			0									
			0									
			0									
			0									
			0									
			0									
	T	(D,E)	0	3	5	-5	-4					D
	T	(E,D)	0	3	5	-8	-4				E	

note	changed	edge(x,y)	distance dictionary					predecessor dictionary				
			A	B	C	D	E	A	B	C	D	E
iterations ↓ last iteration, because we have 5 vertices, we iterate nr-of-vertices −1 times												
	T	(D, E)	0	3	5	−5	−7					
	T	(E, D)	0	3	5	−5	−11		A	B	E	D

We notice that the cost of D and E has constantly decreased over the iterations, creating a negative cost cycle (they will decrease infinitely). This is what the second loop will verify => we have a negative cost cycle, so we cannot find a path between A and E because the predecessor of E will always be D and the predecessor of D will always be E, so we have a cycle created by the E and D loop, so E isn't accessible from any other vertex