CS 566 HW4 Jixing(Jacey) Man

Question1:Consider the following directed Graph

(so I am assuming the adjacent matric format does not consider weight,
So it with be:

Answer: In adjacent Matric format:

	a	**b**	**c**	**d**	**e**
a	0	0	0	0	0
b	1	0	0	0	0
c	1	0	0	1	0
d	1	0	0	0	0
e	0	1	1	1	0

a. But in case I have also put the weight number in Adjacent Matric Format

	a	**b**	**c**	**d**	**e**
a	0	0	0	0	0
b	5	0	0	0	0
c	6	0	0	3	0
d	1	0	0	0	0
e	0	1	3	9	0

b. Run the Dijkstra's algorithm on the above-directed graph, using vertex "a" as the start source. Write down your steps and describe it briefly

Answer: I am assuming the question is asking me to find out the shortest path, "~" means infinite. Below are my steps:

Execute the dikkstra:

	a	b	c	d	e
step1	0	5	~	~	~
step2	0	5	1	~	~
step3	0	~	6	~	~
step4	0	~	6	~	3
step5	0	~	~	1	~
step6	0	~	3	1	~
step7	0	~	3	1	3
step8	0	~	~	1	~
step9	0	~	~	1	9

Getting the shortest path explanation:

Vertex	Node	weight	Path
a	a	0	
b	b	5	a
c	c	6	a,d
d	d	1	a
e	e	(5+1),(6+3),(1+9),(1+3+3), 6,9,10,7	(a+b),(a+c),(a+d),(a+d+c)

Cost per path: a-b-e = 6, a-c-e=9, a-d-e = 10, a-d-c-e = 7

The shortest path from a to e is through nods a-b - e, since the weight is the lowest 6.

Question2. Depth-First Algorithm (4 points):

Consider the directed Graph from task 1 with the weights.

• Start from the vertex ”a” and apply the Depth-First Algorithm. Write your steps and describe them briefly. (2 points)

Answer: So depth first would travel through all nodes, here are my steps:

So starting with a –b –e, then going back from e-b-a, then a-c-e, then e-c-a,then a-d-c-e, then back e-c-d-e

The step number in each nodes are:

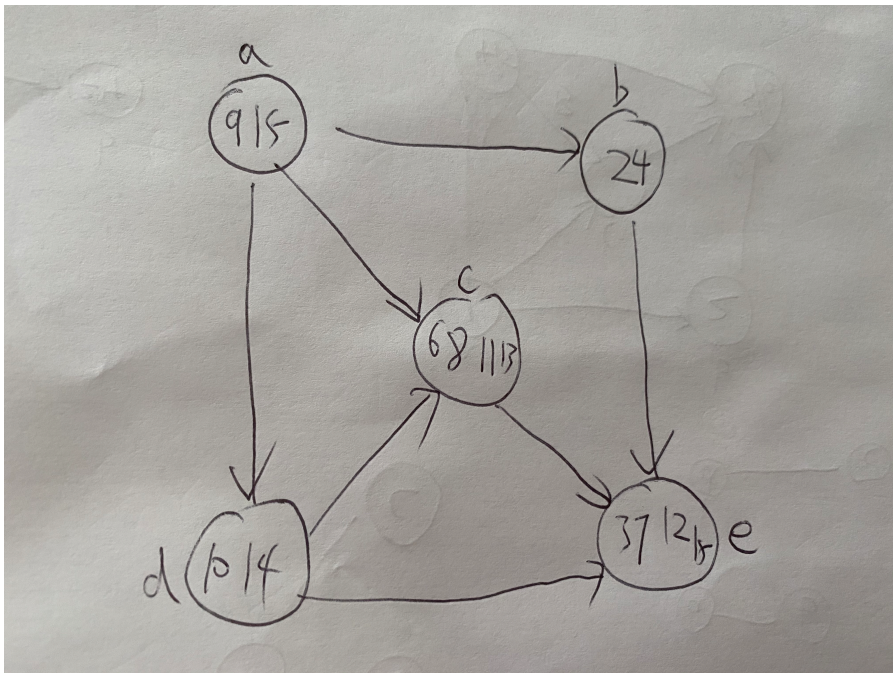
a: step 1,5,9

b: step 2,4

c: step 6,8,11,13

d: step 10,14

e: step 3,7,12,15



• Do you have any Back, Forward or Cross Edges? (2 points)
Back edge: a edge (u,v) that is v is ancestor of edge u, but not part of the dfs tree
Forward edge: edge (u.v) that v is descendant but not part of dfs tree
Cross edge: edge that connects two nodes, but do not have ancestor relationship
Backward edge: the line from d-c
Forward edge: there are no forward edge
Cross edge: the line from b-c

Question 3: Edge Classification (3 points): As we have learned in the lecture, when we apply Depth-First algorithm on graphs, it is possible to have Back, Forward or Cross Edges.
• Which kind of edges (Back, Forward or Cross Edges) is possible on undirected graphs? Describe your answer.
Answer: for undirected graphs, there no back and forward edges. Because without being directed, the back and forward edge on undirected graph can be the same thing, therefore forward edge can be also considered as backward edge. While cross edge would just be tree edges, as there are no restriction on which directions the vertex can go.

Question 4: Design an Algorithm (4 points): Design an optimal algorithm that can detect cycles in a given directed graph G = (V, E).
• Provide Pseudocode for your algorithm, I am assuming this is for DFS, directed graph

```
DFS(V)
for each vertex  u, element V,E do
        u.color = white
        u.pie = nil
end for
time = 0
for each vertex u, element V,E do
        if u.color == white then
                dfs-visit(V,E)
        end if
end for

DFS-Visit(V,u)
time = time + 1
u.d = time
u.color = gray
for each v element V.adj[u]
        if v.color == white
                v.pie = u
                dfs-visite(V.v)
u.color = Black
time = time + 1
u.f = time
```

• Describe the running time of your algorithm
Answer: the running time would O(V+E) because first we are sorting from the top, then contain each vertex's ancestors (nodes) with degree in 0. Then start compute these list of each vertexes in the order of the directions of sorting from the top.

Question 5: Run the Bellman-Ford algorithm on the above-directed graph, using vertex "a" as the start source. Write down your steps and describe it briefly.

1. So the steps would be first a–d–e–c–b–f–e–c–a–d–e–c–d–e–c–b–f in order to run through all of the nodes.
2. For weight see below,

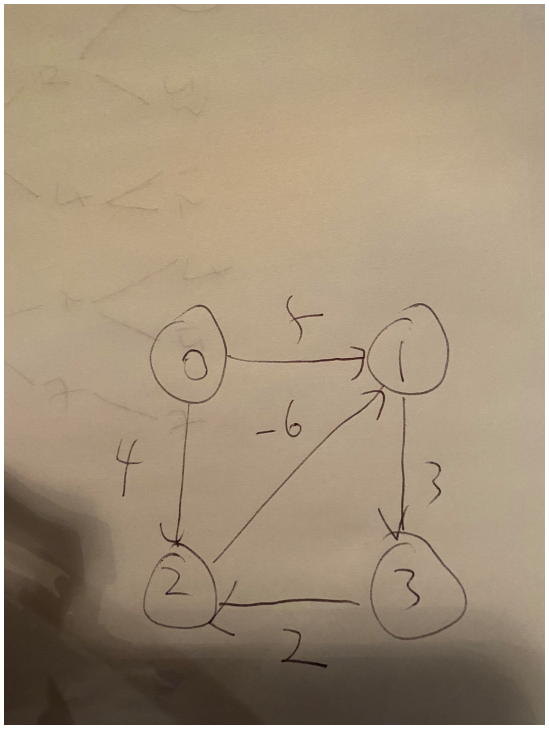
a-d	d-e	e-c	c-b	b-f	f-e	e-c	c-a	a-d	d-e	e-c	c-b	b-f
1	(-2)	3	5	(-1)	(-2)	3	2	1	(-2)	3	5	(-1)

a = 0
b=1-2+3+5-1-2+3+2+1-2+3+5 = 16
c=1-2+3+5-1-2+3+2+1-2+3 =11
d=1-2+3+5-1-2+3+2+1 = 10
e=1-2+3+5-1-2+3+2+1-2 = 8
f=1-2+3+5-1-2+3+2+1-2+3+5-1 = 15

a	b	c	d	e	f
0	16	11	10	8	15

Question 6:
• Describe why Bellman-Ford algorithm does not work when the given graph includes negative cycles.
Answer: Because when we try to use Bellman-ford to find the "shortest" path, the concept of "shortest" isn't clearly defined when negative weight cycles are present, because the possibility of negative infinity with negative weights

• Describe how the Bellman-Ford algorithm detects the negative cycles. Provide an example graph with negative cycles and show how it can be detected. See example graph and explaination below:



edge	weight
0--1	5
0--2	4
1--3	3
2--1	(-6)
3--2	2

If we tale all distance array with “a”, the vertex distance array d[0] = 0, the predecessor array p[0]=0, loop N-1 time, then perform the Nth loop to check for negative cycle

	0	1	2	3
d	0	(-3)	3	1
p	0	2	3	1

We got a change in the distance array, so it means a negative cycle exists, and we can’t find out the shortest path