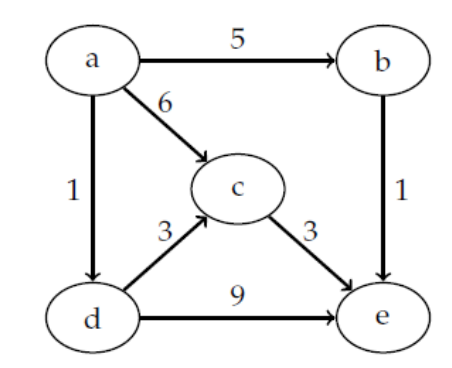
**CS566 Assignment 4**

Yiduo Feng

11/07/2022

**Tasks**

1. Graph Representation, Dijkstra’s Algorithm (3 points): Consider the following directed Graph.



• Represent the above graph in Adjacent Matrix format. (1 point)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e |
| a | 0 | 5 | 6 | 1 | 0 |
| b | 0 | 0 | 0 | 0 | 1 |
| c | 0 | 0 | 0 | 0 | 3 |
| d | 0 | 0 | 3 | 0 | 9 |
| e | 0 | 0 | 0 | 0 | 0 |

Run the Dijkstra’s algorithm on the above directed graph, using vertex “a” as the start source. Write down your steps and describe them briefly. You may use an array to show the value of vertices in each step (e.g., Step 0: {𝑎: 0, 𝑏: ∞, 𝑐: ∞, 𝑑: ∞, 𝑒: ∞}) (2 points)

Step 0: {𝑎: 0, 𝑏: ∞, 𝑐: ∞, 𝑑: ∞, 𝑒: ∞}

Set the initial distances as infinity

Step 1: {𝑎: 0, 𝑏: 5, 𝑐: 6, 𝑑: 1, 𝑒: ∞}

Visit “a” and count the distance of its neighbors, where a->b:1 a->c :6 a->b:5.

Step 2: {𝑎: 0, 𝑏: 5, 𝑐: 4, 𝑑: 1, 𝑒: 10}

Now d has the smallest distance which is 1, so visit “d” and it points to c and e, count the distance of e. a->d->c:4<6(so update c here). a->d->e:10.

Step 3: {𝑎: 0, 𝑏: 5, 𝑐: 4, 𝑑: 1, 𝑒: 7}

Now c has the smallest distance which is 4, so visit “c” and it points to e, count the distance of e. a->d->c->e:7<10(so update e here).

Step 4: {𝑎: 0, 𝑏: 5, 𝑐: 4, 𝑑: 1, 𝑒: 6}

Now b has the smallest distance which is 5, so visit “b” and it points to e, count the distance of e. a->b->e:6<7(so update e here).

Step 5: {𝑎: 0, 𝑏: 5, 𝑐: 4, 𝑑: 1, 𝑒: 6}

At the end, visit “e”, and it points to nothing. We read all of the nodes in the graph.

2. Depth-First Algorithm (4 points): Consider the above directed Graph from task 1.

• Start from the vertex “a” and apply the Depth-First Algorithm. Write your steps and describe

them briefly. You may use a→b→e; a→d to explain (2 points)

step1: a(gray)

read a.

step2: a(gray)->b(gray)

read b.

step3: a(gray)->b(gray)->e(gray)

read e.

step4: a(gray)->b(gray)->e(black)

no way to go, read e again.

step5: a(gray)->b(black)->e(black);

read b again.

step6: a(gray)->b(black)->e(black); a(gray)->c(gray)

read a’s new path.

step7: a(gray)->b(black)->e(black); a(gray)->c(gray)

read c, and {c, e}is a forward edge.

step8: a(gray)->b(black)->e(black); a(gray)->c(gray)

read c, and {c, e}is a cross edge.

step9: a(gray)->b(black)->e(black); a(gray)->c(black)

read c again and mark as black.

step10: a(gray)->b(black)->e(black); a(gray)->c(black); a(gray)->d(gray)

read a’s new path.

step11: a(gray)->b(black)->e(black); a(gray)->c(black); a(gray)->d(gray)

read d, and {d, c} and {d, e} are cross edge.

step12: a(gray)->b(black)->e(black); a(gray)->c(black); a(gray)->d(black)

read d again and mark as black.

step13: a(black)->b(black)->e(black); a(black)->c(black); a(black)->d(black)

read a again and mark as black.

• Do you have any Back, Forward or Cross Edges? (2 points)

Yes, {c, e}, {d, c} and {d, e} are cross edge.

3. Edge Classification (3 points): As we have learned in the lecture when we apply Depth-First the

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.

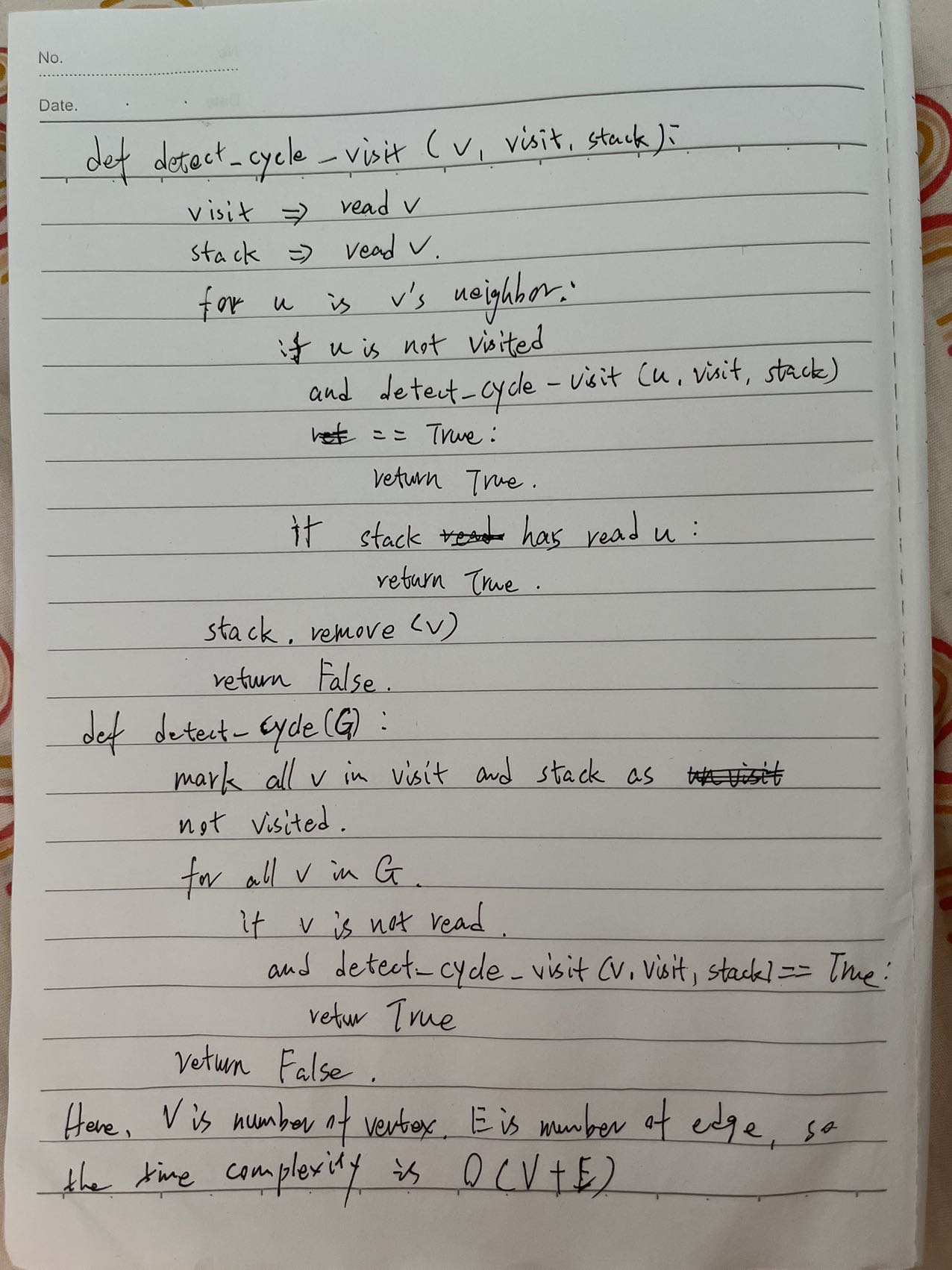
Only back edge can be in undirected graph. Since in undirected graph, edge(u, v) is the same as edge(v, u), but for forward edge and cross edge, when we read u and want to traverse v, we’ve already read (v,u).

4. Design an Algorithm (4 points): Design an optimal algorithm that can detect cycles in a given

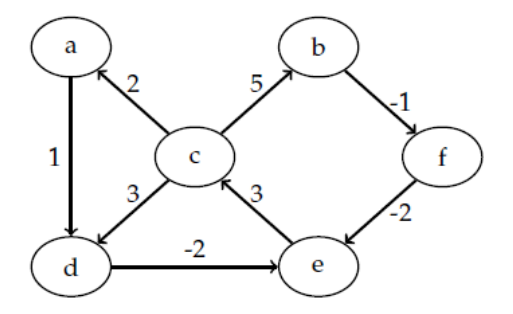
directed graph 𝐺 = (𝑉, 𝐸)

• Provide Pseudocode for your algorithm

• Describe the running time of your algorithm



5. Bellman-Ford Algorithm (4 points): Consider the following Graph



• Run the Bellman-Ford algorithm on the above directed graph, using vertex ”a” as the start

source. Write down your steps and describe them briefly. You may use an array to show the

value of vertices in each step (e.g., Step 0: {𝑎: 0, 𝑏: ∞, 𝑐: ∞, 𝑑: ∞, 𝑒: ∞, f: ∞})

Step 0: {𝑎: 0, 𝑏: ∞, 𝑐: ∞, 𝑑: ∞, 𝑒: ∞, f: ∞}

Set the initial distances as infinity

Step 1: {𝑎: 0, 𝑏: ∞, 𝑐: ∞, 𝑑: 1, 𝑒: ∞, f: ∞}

Read a and it’s path, {a, d} is 1;

Step 2: {𝑎: 0, 𝑏: ∞, 𝑐: ∞, 𝑑: 1, 𝑒: -1, f: ∞}

{d e}is -2, so a->d->e is -1+1 = -1;

Step 3: {𝑎: 0, 𝑏: ∞, 𝑐: 2, 𝑑: 1, 𝑒: -1, f: ∞}

{e, c} is 3, so a->d->e->c is -1+3 = 2;

Step 4: {𝑎: 0, 𝑏: 7, 𝑐: 2, 𝑑: 1, 𝑒: -1, f: ∞}

{c, b} is 5, so a->d->e->c->b is 2+5 = 7;

Step 5: {𝑎: 0, 𝑏: 7, 𝑐: 2, 𝑑: 1, 𝑒: -1, f: 6}

{b, f} is -1, so a->d->e->c->b->f is 7-1 = 6;

The traversal stops here because it is the smallest path we can find.

6. Bellman-Ford algorithm (2 points):

• Describe why the Bellman-Ford algorithm does not work when the given graph includes

negative cycles.

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

If there is a negative cycle in a graph, it means that each time we traverse the cycle, we can find a smaller path. In this way, the Bellman–Ford algorithm will report an error because it keeps repeating the negative loop.

According to the lecture slide

