CSCI-GA.1170-001 Homework 10

Ankit Sati

TOTAL POINTS

40.5 / 40

QUESTION 1

1 Avoiding Double Visits 12 / 11

Part a)

✓ **- 1 pts** *Missing or Incorrect:*

$$u \in A, v \in A:$$ Add$$ (u_1, v_1), (u_2,v_2)$$ to $$E'$$

✓ **- 1 pts** *Missing or Incorrect:*

$$u \in B, v \in A:$$ Add$$ (u, v_2)$$ to $$E'$$

Part b)

✓ **- 0 pts** *Correct*

Part c) Extra Credit

✓ **+ 3 pts** *Click here to replace this description.*

QUESTION 2

2 Counting Shortest Paths 9.5 / 10

✓ **- 0.5 pts** ***4-2) a)** Incorrect/Missing Correctness*

✓ **- 0 pts** ***4-2) b)** Correct*

✓ **- 0 pts** ***4-2) c)** Correct*

QUESTION 3

3 DAGs, Paths, and DFS 13 / 13

✓ **- 0 pts** *Correct*

QUESTION 4

4 Tricky DFS 6 / 6

✓ **- 0 pts** *10-4) all correct*

LO-1

a) G' will contain V' vertices and E' edges.

Now, here we will take 2 copies of each node in A and a single copy of every node in B.

So, V' will contain $2n_1 + n_2$.

$= n_1 + n_2$ vertices

Now $|V| = 2n_1 + n_2 = 2(n_1 + n_2) - n_2$.

$= |V'| < 2(n_1 + n_2)$.

$\Rightarrow |V'| < 2|V|$ ——— (1)

⇒ Now talking about E', we will include those edges such that all paths between two copy nodes of A contain at most one node of B.

⇒ Now in worst case all the edges which are in E will occur twice for both copies

So $|E'| \leq 2|E|$

This has 2 cases.

No B nodes are involved : $n_1(n_1-1)$

1 B node involved : $n_1(n_1-1) + 0 + n_1 \rightarrow u \in A, v \in B + n_2(n_2-1)$

And for they 4 cases

$|E| : n_1(n_1-1) + n_2(n_2-1) + n_1 n_2 + n_2 n_1$

$\therefore |E'| = 2n_1(n_1-1) + n_1 - n_2 + n_1 n_2$.

Q. $2|E| = 2n_1(n_1-1) + 2n_2(n_2-1) + 2n_1n_2 + 2n_1n_2.$

$$2|E| - |E'| = 2n_1(n_1-1) + 2n_2(n_2-1) + 4n_1n_2 - 2n_1(n_1-1) - n_1 + n_2 - n_1n_2.$$

$$= 2n_2^2 - n_2 + 3n_1n_2 - n_1 \geq 0$$

$$\Rightarrow |E'| \leq 2|E|$$

In the worst case every node edge in $|E|$ is there in E'

$|E'| \leq 2E$ still holds.

Hence sign of graph is indeed satisfies the required constraints ✓.

(b) The algorithm to be used will be very similar to the one used in BFS algorithm.

Now, there are two copies of each node belonging to set A.

⇒ Let us consider 2 partition of these nodes. Now if we start BFS from a source in Partition 1, then we will not take the vertex ~~from Partition~~

If source is in Part 1, then the corresponding vertices will be taken from part 2.

This is done in order to maintain the structure of Graph G'

As BFS is used on G'.

Runtime is $= O(|V'| + |E'|)$.

$= O(|V| + |E|)$. $\{ V' \leq 2V, E' \leq 2E \}$

The intension of our transformation is that at the end we are using source from part 1 and other nodes from part 2.

This ensures that set 2 loop occurs & we calculate the shortest distance from source to every other $v \in A$ which will pass through almost one from B.

In this manner we can compute shortest distance from s to v following the constraints of G from G'.

Vertex from another Partition

example $A = \{1, 2, 3\}$

$B = \{4, 5\}$

So a shortest path from 1 to 3 can be.

$1 - 4 - 2' - 3'$.

1 is the source of Partition 1

$2'$ & $3'$ represent the vertices of A from partition 2.

4. is the vertex of B.

(C) Using an iff argument.

Claim 1. : Every desired path in G_1 exist in G_1'.

All paths in G_1 have a source $\in A$ and other vertices $v \in A$, passing through almost once from B.

So source will be taken from part 1 & other vertices will be taken from part 2 & B.

This way we ensure that path of G_1 always exist in G_1'.

Claim 2 ⇒ Every path in G_1' corresponds to a desired path in G_1.

The BFS in G_1' will take source & other vertices from separate parts. The path originate at A, Pass through B atmost once & stop at some other vertex $x \in A$. Thus these parts correspond to path desired in G_1.

Hence both the claims are justified ⇒

Correctness is justified ⇒.

1 Avoiding Double Visits 12 / 11

Part a)

✓ **- 1 pts** *Missing or Incorrect:*

$$u \in A, v \in A:$$ Add$$ (u_1, v_1), (u_2,v_2)$$ to $$E'$$

✓ **- 1 pts** *Missing or Incorrect:*

$$u \in B, v \in A:$$ Add$$ (u, v_2)$$ to $$E'$$

Part b)

✓ **- 0 pts** *Correct*

Part c) Extra Credit

✓ **+ 3 pts** *Click here to replace this description.*

10-2

(a) Calculation of number of shortest path from s to v, we need to calculate all possible ways to reach v from all of its parent.
This is easy to implement.
If a parent discovers a child, its number of shortest paths:
Now if another parent discovers same child, we merely need to add this parent number of shortest path to the child. This way we get the total number of shortest possible paths for a node v.

Algorithm

MBFS(G, s)

```
for each u ∈ G, V - {s}
        u.color = WHITE
        u.d = ∞
        u.π = Nil.
        u.M = Nil
s.color = GRAY
s.d = 0.
s.π = NIL.
s.M = 1.
Q = ϕ.
```

ENQUEUE(Q, s).

```
while Q ≠ ϕ
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1.
```

v. π = u

v.n = u.n.

ENQUEUE.(Q, v).

else if v.Color == GRAY.

if v.d == u.d + 1.

v.n = v.n + u.n.

u.Color = BLACK.

Runtime = O(m+n) m → edges

n → vertices

Just a modification of BFS so nothing much will change in its runtime.

All Graphs are undirected ✗ ✗ ✓

2.b

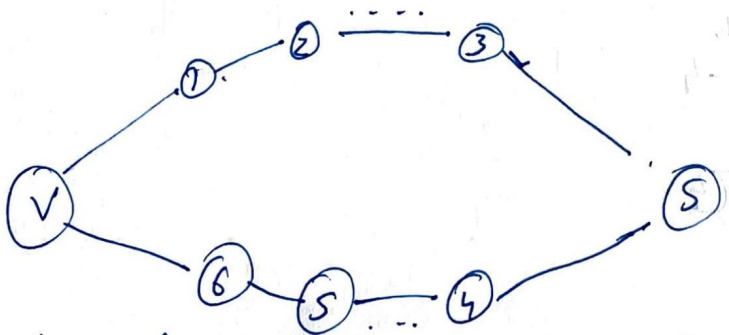
Simple Cycle → Cycle without repeated vertices.

Proof by Contradiction.

⇒ Assume that there are no simple cycles in a graph having more than 1 ~~cycle~~ shortest path from v to s.

In this case, there exists two such shortest paths from v to s, such that apart from v & s, no other nodes are common.

i.e.



→ There exist a disjoint pair of nodes in these two paths such that they only share v & s

→ Since graphs are undirected, the direction don't matter.

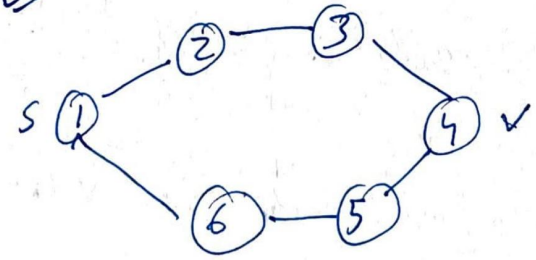
→ They form a simple cycle with no repeated vertices.

→ But our assumption states that there can be no simple cycles in this case.

Hence our assumption is wrong ✓✓.

⇒ If there is more than one shortest path from v to s, the graph must contain a simple cycle. ✓✓.

Example



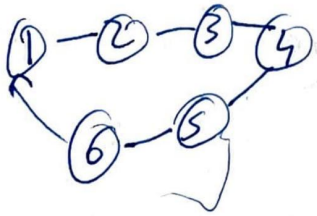
Hence proved.

s = 1
v = 4.

Path 1 = 1 – 2 – 3 – 4.

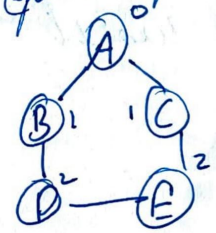
Path 2 = 1 – 6 – 5 – 4.

Cycle =



2.C. Gary is wrong

eg. example



A, m = 1.
B, m = 1.
C, m = 1.
D, m = 1.
E, m = 1.

All vertices have number of shortest path = 1, there exist a cycle.

A — B — D — E — C — A.

Therefore Gary is wrong.

2 Counting Shortest Paths 9.5 / 10

✓ **- 0.5 pts** ***4-2) a)** Incorrect/Missing Correctness*

✓ **- 0 pts** ***4-2) b)** Correct*

✓ **- 0 pts** ***4-2) c)** Correct*

Question 10-3

a) DFG (G)

```
for each v ∈ G.V
    u.color = WHITE
    u.dist = 0
time = 0
for each u ∈ G.V
    if u.color = WHITE
        DFS-VISIT(G,u)
```

```
DFS-VISIT(G,u)
    time = time + 1
    u.d = time
    u.color = GRAY
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.π = u
            DFS-VISIT(G,v)
            u.dist = max(u.dist, 1 + v.dist)
    u.color = BLACK
    time = time + 1
    u.f = time
```

Runtime = $O(m+n)$ where m = number of edges.
n = number of nodes.

This is similar to DFS where. ∀ node u, calls DFS-VISIT(G,u) exactly once.

(b). Proof by induction

Given

IH -

P_k : V.distance $\leq L[v]$, $\forall v \in V$ if $L[v] = k$.

Check Base Case. $k = 0$.

$L[v] = 0$.

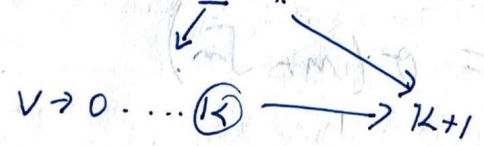
and V.dist = 0 as no node can reach any other side node.

$\Rightarrow$ V.dist = $L[v] = 0$.

$\Rightarrow$ Hypothesis holds true

Showing $k+1$ holds true, assuming the above.

if $k+1^{th}$ node is reachable by any node then the longest path from v with k nodes will be $k+1$.



All other paths of V.dist will be less than k or equal to it. Adding 1 to it will still make it $\leq k+1$.

Therefore

P_{k+1} : V.dist $\leq L[v]$, $\forall v \in V$ s.t.

$L[v] = k+1$.

Since Induction holds true

$\Rightarrow$ Our hypothesis is correct.

c) V.color = GRAY when DFS is processing the edge (v, u)

(d) Base Case $k=0$;

No node is connected to any other node.

$\Rightarrow$ $V.dist = 0 \quad \forall \; v \in V$

& $L[v] = 0$ or $L[v]$ is the true longest path.

$\rightarrow$ $V.dist = L[v] = k$.

$\rightarrow$ Induction Hypothesis is true

Induction Hypothesis:

P_{k+1} $V.dist = L[v], \forall \; v \in V, s.t. \; [v] = k-1$

Now, we will complete the induction step for 3 cases.

(1) $u.color = BLACK$. $\Rightarrow$ it is processed.

$v.dist = 1 + u.dist$.

$= 1 + L[u]$

$= 1 + k - 1$.

$v.dist = k$.

& $L[v] = k$.

$\Rightarrow$ Induction step is correct.

$\Rightarrow$ Hypothesis stands good.

(2) $u.color = WHITE$

u. is not discovered yet.

So, DFS-Visit (G, u) will be called & then

$v.dist = max(v.dist, 1 + u.dist)$.

{from hypothesis}, $u.dist = L[u]$

$= k-1$.

$\therefore$ $v.dist = max(v.dist, k)$.

Assuming $u.dist$ was longest all other children of v will return distance $< k$.

$\max(V.dist, k) = k$.

$\Rightarrow V.dist = k$.

and $L[v] = k$.

$\Rightarrow$ Induction step is correct.

$\Rightarrow$ Hypothesis is true.

3. $u.color = GRAY$.

Case is impossible.

Since we are processing edge $(v, u) \Rightarrow v$ is GRAY and parent to u, which can either be WHITE (waiting to be discovered) or BLACK (already processed).

And Inductive hold in both of these cases.

$\therefore$ Hypothesis holds true in all 3 cases.

3 DAGs, Paths, and DFS **13 / 13**

✓ **- 0 pts** *Correct*

Problem 10-4

(a)

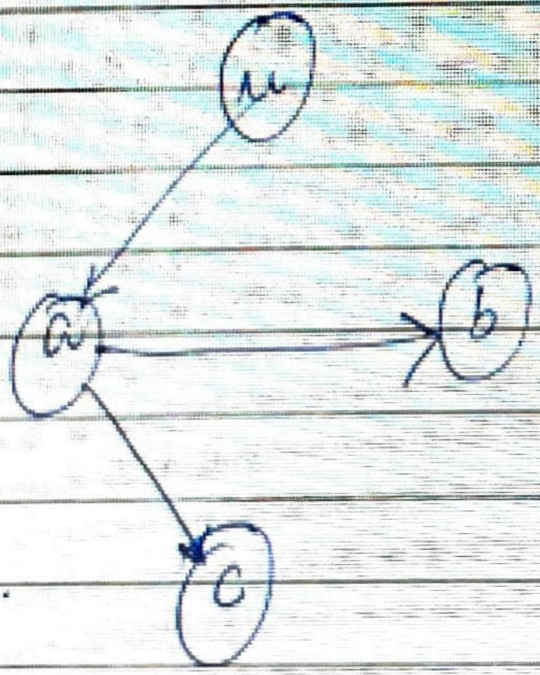
Since $e = (c, d)$ is the cross edge.

So, c and d have no parent-child relationship between them.

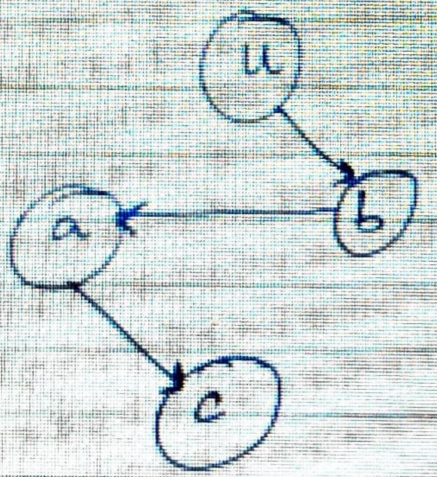
Now, for the conjecture to be true, we will say that (c, d) should not be a back edge because otherwise some directed cycle will be there in the graph.

The error in the proof is that we are considering the graph to be a DAG (Directed Acyclic graph). But it is not true as a directed graph may also contain some cycle.

(b) (c, b) is a cross edge in



(a, b) u → back edge in :



4 Tricky DFS **6 / 6**

✓ **- 0 pts** *10-4) all correct*