FA Homework 10

Q 1.

a)

→ Subset $A \equiv \{y_1, \_\_, y_{n_1}\}$
y_s - source

To find $dist(y_s, y_i)$ $\forall i$, $y_i \in A$

→ Subset $B = \{x_1, \_\_\_, x_{n_2}\}$

→ Without loss of generality, we start with y_s, let y_3 be connected to $y_{u_1}, -, y_{u_I}$ and $x_{v_1}, -, x_{v_J}$ such that (y_s, y_{u_i}) is a directed edge and (y_s, x_{v_j}) is a directed edge.

→ For our transformed graph G',
(y_s, y_{u_i}) are added as they are,
(y_s, x_{v_j}) are added as they are,

for each x_{v_j}, the edges from x_{v_j} to set A vertices are added to the graph by adding a new corresponding y vertex to our graph. If (x_{v_j}, y_k) is an edge, we add (x_{v_j}, y_k^2) in our Graph G'.

If we call the y_k arising from $A \to A$

connections as y'_k, y'_k outgoing connections for set A vertices are equivalent to y^2_k.

We repeat this process for other y vertices. We are esentially excluding the (x_i, x_j) edges and repeating the y verties for the conditions of our problem.

let there be 2 sets of vertices A_1, A_2 such that $A_1 \equiv A_2 \equiv A$

Our G' includes the following connections,

$$A_1 \to A_1$$

$$A_1 \to B$$

$$B \to A_2$$

$$A_2 \to A_2$$

$$|V'| = |A_1| + |A_2| + |B|$$
$$= 2|A| + |B| \qquad \# |V| = |A| + |B|$$
$$|V'| \leq 2|V| \;//$$

$$|E'| = |A_1 \to A_2| + |A_2 \to A_1| + |A_1 \to B| + |B \to A_2|$$

$$= 2|A \to A| + |A \to B| + |B \to A|$$

$$|E'| \leq 2(|A \to A| + |B \to B| + |A \to B| + |B \to A|)$$

$$|E'| \leq 2|E| \quad //$$

Q1. (b)

Algorithm to be invoked - BFS on G'.

Invocation call - $BFS(G', s)$

$\#$ vertices $G' \leq 2V$ - 1.a

$\#$ edges $G' \leq 2E$ - 1.a

V - # of vertices in G
E - # of edges in G

For BFS on $G(V, E)$,

$$T(G) \equiv T(V, E) = O(V + E)$$

$$T(G') = T(V', \mathcal{E}') = O(V + E)$$

$$T(V', \mathcal{E}') \equiv O(V' + \mathcal{E}')$$

for all sufficiently large $V', \mathcal{E}'$ sizes.

$$V' \leq 2V$$
$$\mathcal{E}' \leq 2\mathcal{E}$$

Using the definition of Big Oh notation,

$$T(V', \mathcal{E}') = O(V + \mathcal{E})$$

Inversion :

For $s \in A$, $y_i \neq s$,

$$y_1^i \in A_1 , \quad y_2^i \in A_2$$

$y_i.dist = \min(y_i^1 dist, y_i^2.dist)$

Justification:

→ Assuming the proof of correctness of BFS, the valid shortest path as exists in G', will be found by BFS on G'.

→ The path from s to y_i will either be exclusively A vertices, or an $A \to B \to A$ path with one B vertex. The first kind of paths will be reaching to y_i^1 and the second kind of paths will be reaching to y_i^2. We will take the min of the path lengths for finding the shortest valid path as from s to y_i, $y_i \in A$.

Q1. c.

For $s \in A$, $\in A$,

Statement P:

P- Path (s, y_i) is the path such that

Path (s, y_k) is the path such that the conditions are satisfied. (shortest path)

Statement q: The BFS on G' finds P.

Proof:
lemma 1: ~~If~~ P is a valid shortest path, then BFS on G' will find P.
$(p \to q)$

Case 1:
Path (s, y_k) consists of nodes in set A only.

$\therefore$ Path $(s, y_k) \in$ sub graph $(A \to A)$

We have $A_1 \to A_1$ connections in graph G. Therefore such a path exists in G'.
BFS (G') will find the shortest path distance between two vertices s, y_k

Case 2:
Path (s, y_k) consists of a series of nodes in A starting of s, a node x_j in B, another series of nodes in A ending with y_k.

$P = s - y' - x_j - y'' - y_k$

$P = s - y - x_j - y \quad y_k$

$y' \in y^*$, y', y'' can be empty sets
$y'' \in y^*$

$P \equiv A_1 \to b \to A_1$

let $y' \equiv [y_s -, y_p]$

$y'' \equiv [y_r, \text{—}, y_k]$

$y_s, \text{—}, y_p \in A_1$, exists in G'.

$y_p - x_i \in A_1 \to B$

$x_i - y_r \in B \to A_2$

Both edges exist in G'.

$y_r - y_k$ will be contained in the

$A_2 \to A_2$ connections and therefore will be present in G.

$\therefore$ If there is a valid shortest path, it will be found by BFS in G'.

Lemma 2 : A path that is found by BFS in G' is a valid, shortest path. i.e. If the BFS on G' finds path P, then, P is a valid shortest path. $(q \rightarrow p)$

G' does not contain $B \rightarrow B$ edges.
$\therefore$ There will not be any path in G' which will have two consecutive nodes from set B. i.e. $x_i - x_j$ or more x vertices consecutively will not appear. if a BFS search lands on x_i, it must next go on some $y \in A$ or terminate.

$\therefore$ Lemma 2 works.

From Lemma 1 and Lemma 2,

$$(p \rightarrow q) \wedge (q \rightarrow p) \equiv (p \leftrightarrow q)$$

$\therefore$ Our if and only if argument is valid. Path P is a valid shortest path if and only if the BFS on G' finds P.

Q2. (a)

```
BFS ( G, s)

{
   for each vertex u ∈ G.V - {s}
         u.d = ∞
         u.n = 0

   s.d = 0
   s.n = 1

   Q = ∅

   Enqueue (Q, s)

   while  Q = ∅  do
     {

        u = Dequeue (Q)

        for each  v ∈  G.Adj[u] :

            if  v.d == ∞ :

                  v.d = u.d + 1

                  v.n = u.n
```

```
            else if v.d == u.d + 1 :

                        v.n = v.n + u.n

        }


}
```

→ If v is visited the first time, we store in $v.n$, the number of paths in $u.n$, the path to v is through u. For every path p' from s to u, there will be a path from s to v, $P = [P' (u,v)]$

→ Based on the properties of BFS, if u is explored after v, v will have to be at a distance greater than u, else v would have been dequeued first.

→ If v is visited during BFS through being adjecent to some other u; the paths in that u, let's say u', are added to the paths in v.

Implicit claim:

When we dequeue u; the number of paths to u are known.

Proof:

Base Case: for dist = 1, the shortest path is ~~of~~ that vetrex will be of length one which is essentially the edge from s to that vertex. It will be dequeued after v.d is adjusted as 1 (s.d {0} + 1) and there will be no other equivalent shortest paths. { s.n initialized to 1 }

for dist = i, assume validity of u.n.

for dist = i+1, let there be some vertex v, the dequeue-ing of v will take place after the dequeue-ing of u as based on the property of the DFS queue.

∴ The updates to v from vertices of dist = i will be valid. ∴ v.n will be valid.

Q2. b.

Given:
For an undirected graph, there is more than one shortest path from v to s.

To prove:
The graph G contains a cycle.

Soln:
Consider two paths from s to v,

$P1 : s - u_1^1 - u_1^2 - u_1^{n-1} - v$

$P2 : s - u_2^1 - u_2^2 - - - u_2^{n-1} - v$

$dist.v = n$

For paths to be distinct,

$\min(dist.v) = \min(n) = 2 \quad , u_1^1 \neq u_1^2$

for $n \geq 2$.

Let u_i, u_j be the first vertices for which

$$u_i^1 = u_i^2$$

$$u_j, u_i \in \{ u_2, \_, v \}$$

Since we are talking about paths from u to v, u_i must be v in the maximum.

let $u_i^1 = u_j^2 = x$

If the indices for P1, P2 were different, the lengths won't come come to

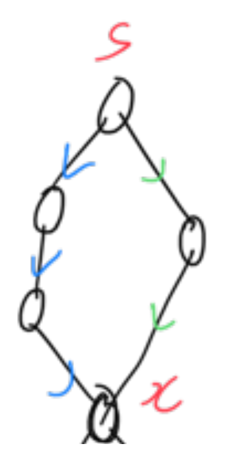
$\therefore$ p', p'' present in Graph G such that,

$p' : s - u_1^1 - - u_1^r - x$

$p'' : s - u_2^1 - - - u_2^q - x$

$$C \equiv s - u_1^1 - - u_1^p - x - u_2^q \cdot - - u_1^2 - s$$

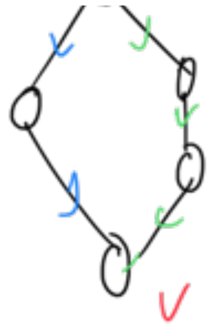
Eg:



P1

P2

$u_i = u_3^1$



$u_j = u_2^2$

$dist(s, v) = 5$

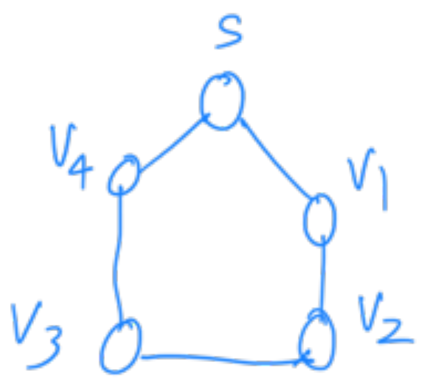
We can see that the path C is a cycle.

$$\# \; C \equiv s - u_1^1 - - u_1^p - x - u_2^q \cdots - u_1^2 - s$$

Q2. (c).

Gary is incorrect.

Counter example:



$\text{dist}(s, v_1) = 1$ # 1 path $\{(s,v_1)\}$

$\text{dist}(s, v_2) = 2$ # 1 $\{(s,v_1),(v_1,v_2)\}$

$\text{dist}(s, v_3) = 2$ # 1 $\{(s,v_4),(v_4,v_3)\}$

$\text{dist}(s, v_4) = 1$ # 1 $\{(s,v_4)\}$

The number of shortest paths is 1 for each v, but there exists cycle $C =$ $s - v_1 - v_2 - v_3 - v_4 - s$.

Q3.

(a)

DFS(G)

for each $u \in G.V$
    $u.color = white$
    $u.dist = 0$

for each $u \in G.V$

```
    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)

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

    u.color = Black

    time = time + 1

    u.f = time
```

→ The running time of the algorithm is the running time of DFS.

→ $$T(V, E) = O(V + E)$$

→ The number of DFS Visit Calls will be proportional to the number of vertices V. For each vertex, the for loop will run in time proportional to the number of outgoing edges.

$$T(V, E) = \left(\sum_{v_i} (c_1 + c_2\, e(v_i)) \right)$$

$$T(V, E) = O(V + E)$$

c_1, c_2 - constants

→ for some node v, for each of it's children compute u.dist in the recursive call. Now, v.dist will be 1 + the child which has the maximum u.dist

→ Therefore, after each recursive call to a child of v, we check if we are getting a max $v.dist$ value which is $u.dist + 1$.

Q3. (b)

Lemma :

→ for some v which is the parent of u. We claim,

$$L[v] \geq 1 + L[u]$$

→ ~~If~~ for contradiction, for some u, if $L[u] + 1 > L[v]$, then, we assign $L[v]$ as $L[u] + 1$, hence violating our assumption. Our lemma is this valid by contradiction.

→ To prove:

$$v.dist \leq L[v] \in V \text{ s.t. } L[v] = k$$

→ $L[v] = k$

$\forall u$ such that $u \in Child(v)$

$L[u] \leq k-1$ - from lemma.

Inductive hypothesis:

$$dist[u] \leq L[u] \quad \forall\, u \text{ such that } L[u] \leq k-1$$

→ Base case:

$L[leaf] = 0$
$dist.leaf = 0$

$$dist[leaf] \leq L[leaf]$$

Inductive Step:

→ for v, let $U \in \{u_1, \_, u_i\}$ be children of v.

for some u,

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

$$v.dist - 1 = u.dist$$

$v.dist - 1 = u.dist$

$L[u] \leq L[v] - 1$ — lemma

$u.dist \leq L[u]$ — inductive assumption

$\therefore \; v.dist - 1 \leq L[v] - 1$

$v.dist \leq L[v]$

Q3. (c)

v.color when DFS is processing the edge (v, u) is gray.

Q3. (d)

To prove:

$$r.dist = L[r]$$

Base Case:

$L[leaf] = leaf.dist = 0$

$L[leaf] = leaf.dist = 0$

Inductive hypothesis:

$L[u] = u.dist$ such that
$L[u] \leq k-1$

Inductive Step:

For v, let the children be u and u'

$u' = \{ u'_1, u'_2 \_ \}$ if u' exists

1) If $u.color$ = black, $L[u]$ is computed
- from algorithm steps,
inductive assumption

for the loop consisting of nodes adjecent to v, we would have crossed $u \in Adj[u]$, DFS-Visit call on u will have been completed, we would have made the adjustment for $v.dist = 1 + u.dist$ outside the if statement, Since the longest path from v is through u, $v.dist$ as assigned by $u.dist$ is $L[v]$. For any $v.dist$ previously

1+ u.dist is L[v]. For any v.dist previously assigned, 1+ u.dist $\geq$ v.dist (old). The assignment will hold.

$$L[v] = v.dist$$

Also, any previous value of v.dist will be less than L[v] bast on our earlier proof.

$$\text{i.e. } L[v] \geq v.dist \quad - \text{ proof 3.b}$$

$\therefore$ Our max based assignment for v.dist as contingent from comparision with (u.dist + 1){L[v]} will be valid.

2) If u.color = white

→ if v.dist isn't updated,

$$v.dist = 0 \leq L[v]$$

→ for some $u' \neq u$ such that v.dist is updated wrt u'.

$$v.dist \leq L[v]$$

→ This is the case where DFS-VISIT to be called on u. L[u] is yet to be computed based on the recursive computations of the sub graph rooted at u.

3) If u.color = gray

→

\# if u is the first vertex visited from v.

→ if v.dist isn't updated,
$v.dist = 0 \leq L[v]$

→ for some $u' \neq u$ such that v.dist is updated wrt u'.
$v.dist \leq L[v]$

This is the case where DFS-Visit is called on u and L[u] is in the process of being computed.

Q4.

(a)

(a)

Let $e = (c, d)$ be a cross edge.

→ In our DFS traversal, we could either reach c first or d first.

For (c, d) to be a cross edge:

→ If we reach d first, from the DFS call at d, we do not reach c, i.e. vertex d and vertex c do not have a ancestor decesendant relationship.

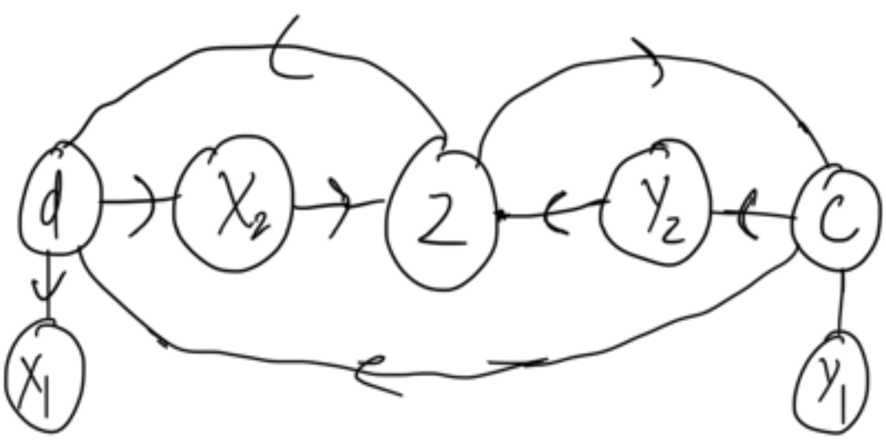
→ Vertex c is reached through another branch of the DFS after the recursion at d is terminated and the DFS control flow goes to the parents of d and their ancestors recursively. This follows from the Paranthesis theorem for DFS search.

→ This enables edge (c, d) to be a cross edge. Any version of the DFS which reaches d before c will be analogous to this DFS search in terms of edge (c, d) being the cross edge.

→ If in DFS, c is reached before d, c and d will have to have an ancestor descendent relationship for since the edge (c,d) exists, d will have to fall in c's subtree in the DFS search.

→ ∴ Since c is d's ancestor, (c,d) can't be a back edge.

Error :



Consider the above diagram,

→ let 2 be a node which has connections to both d and c and it is also to be reachable from both d and c.

If we start from d, we reach 2 and consequently, we reach c. Therefore, d and c have a ancestor descendant relationship and $e(c,d)$ is a back edge.

If we start from z, we reach c and d in two sub branches of z such that c and d do not occur in each others respective sub trees.

→ $e(c,d)$ will therefore be a cross edge.

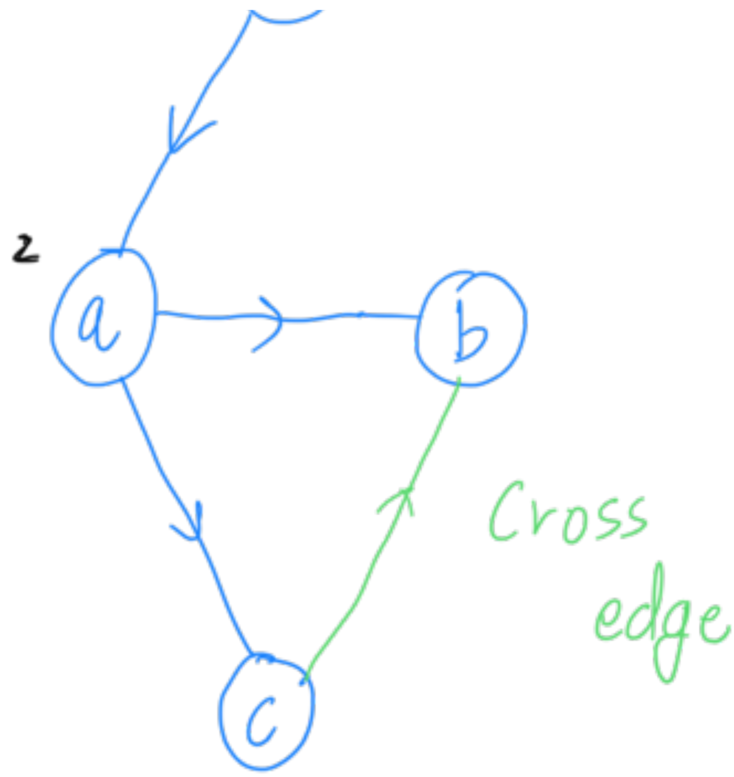
→ We can see that if there is a descendant of both c and d, such that, from that descendant, paths can be made to c and d such that, the path from z to c does not include d, then,

→ if we start from z, we reach c, d in two different branches of DFS traversal and edge c,d is a cross edge.

→ if we start from d, we get to z and the c, d and c have an ancestor descendant relationship. Edge c,d is a back edge.

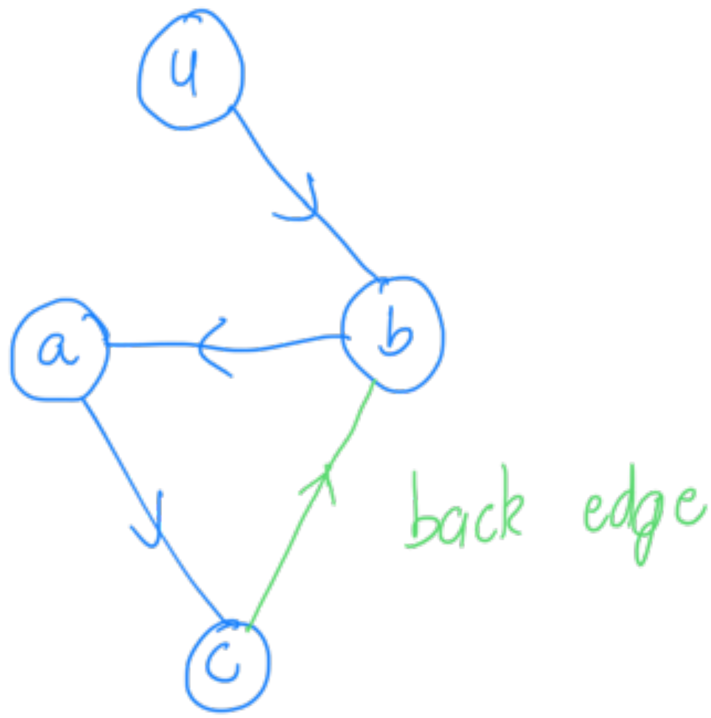
Q4.(b).

Case 1 : Cross edge



1. u to a
2. a to b
3. a to c

Case 2 : back edge



1. u to b

2. b to a
3. a to c