Homework # 4

3.15 (a) Construct a directed graph with intersections as nodes and streets as edges. Check whether the directed graph is strongly connected.

- If the answer is yes (strongly connected), then you can go from any intersection to any other intersection.

(b) From the above graph constructed, starting from town hall, Consider all the vertices X reachable from the town hall vertex (do a BFS on G, with town hall as start node).

Find the Strongly connected component of G. If the Strongly connected component containing town hall also includes X, then the answer is yes. Else the answer is NO.

For both problems: Find the Strongly connected component of G.

(a) G has to have only one strongly connected component.

(b) The strongly component containing town hall should include X.

3.17

(a) If u and v belonging to two different strongly connected componets and if u and v are in $f(o)$ implies that there is a path from u to v and v to u a contradiction.

(b) Check whether there is a strongly connected Componet of size greater than 1. Then it will have an infinite trace.

(c) Check if there is a strongly connected Componet of size greater than 1, containing the good vertex.

(d) If G has an infinite Trace containing some good vertex, (There is a path from Bad vertex to good vertex); it means that there is a strongly connected componet containing the good vertex.

3.22. Find the strongly connected components; and the corresponding Strongly Connected Component graph. That strongly connected component graph is a single path, then the source vertex is the vertex s, from which all vertices are reachable.

3.24

Do topological sort of the given DAG.

From the resulting sequence

$a_1, a_2 \cdots a_n$, check whether there is an

edge from a_i to a_{i+1}.

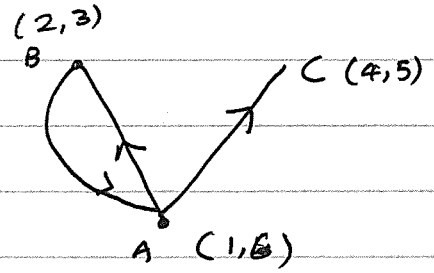
~~The reason why it works is~~

Or calculate longest path from the source vertex to the sink vertex [There has to be one source vertex and one sink vertex - otherwise no such path exists].

If the path length is $n-1$, such a path exists,

otherwise no such path exists.

5 22.3.8



A.pre = 1 B.pre = 2 C.pre = 4

A.post = 6 B.post = 3 C.post = 5

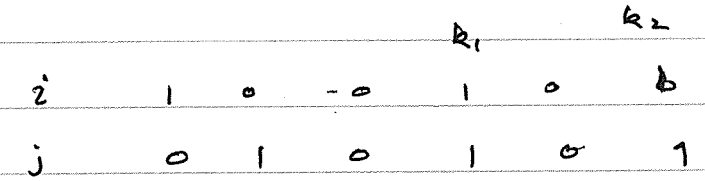
B.pre < C.pre Edge (B, A) is the back edge

But C is not a descent of B in the The DFS traversal.

4.3.

Let A be the adjacency matrix. (n × n)

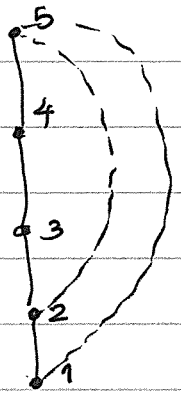
```
for i = 1 to n
    for j = i+1 to n
        sum = 0
        for k = 1 to n
            if (A[i][k] == A[j][k]) and (A[i][k] == 1)
                sum = sum + 1
                if (sum == 2), { there is
                    a cycle of length 4.
                    exit }
there is no cycle of length 4.
exit.
```



$A[i, k_1] = A[j, k_1] = 1$ $\quad (i, k_1, j, k_2)$

$A[i, k_2] = A[j, k_2] = 1$ $\quad$ form a four cycle.

4.4.



Shortest cycle is of length 3, containing edge (1,2), (1, 5), (2, 5).

The reason ~~it~~ the method suggested does not work is the shortest cycle may contain more than one back edge.

4.7. Let T be the shortest path tree.

Compute $D(s,v)$ in T

$D(s,v)$ is the shortest distance from s to v.

For each (u,v) in G,

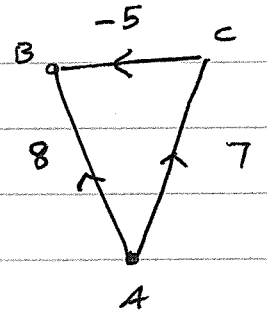
Compute $D(s,u) + w(u,v)$

If this is greater than or equal

$D(s,v)$, the given tree T is the

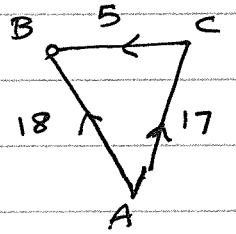
shortest path tree.

4.8.



~~If~~ The shortest path from A to B is 2.

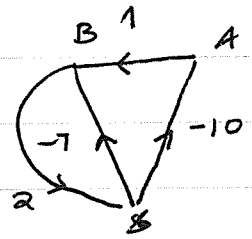
If we add 10 to each edge,



Compute the shortest path from A to B

$18 - 10 = 8$ which is not correct.

4.9.



There is a negative weight cycle involving S.

Hence Shortest path algorithm will not work.

If there is no negative weight cycles; then the algorithm will work - became a node when popped from the priority queue will never enter (as the distance of those vertices will be strictly non-decreasing).