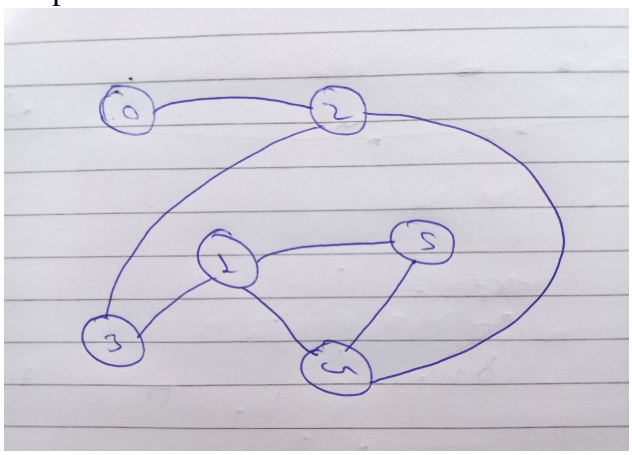
Problem Set 5

Name: Vivek Verma

Collaborators: None

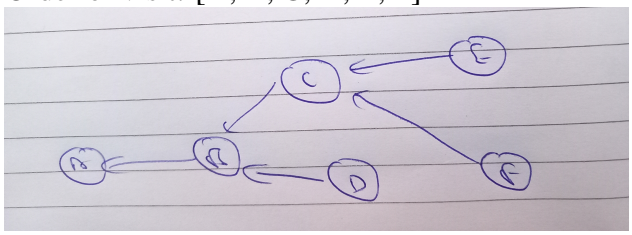
Problem 5-1.

(a) Graph

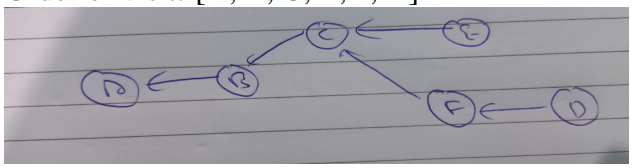


(b) { A: [B],
B: [C, D],
C: [E, F],
D: [E, F],
E: [],
F: [D, E] }

(c) BFS
Order of visit: [A, B, C, D, E, F]



DFS
Order of visit: [A, B, C, E, F, D]



(d) Topological order after removing (D, F): [A, B, C, F, D, E]
Topological order after removing (F, D): [A, B, C, D, F, E]

Problem 5-2.

Construct a Graph G where power plants and buildings are nodes of the graph and Wires are edges.

Since there are n power plants and there can be atmost one wire connected to each power plant. Also there are n^2 buildings and each building has to be powered only once. Therefore the number of wires are upper bounded by $O(n^4)$.

$$\because |E| \leq \left(\frac{|V|}{2}\right)$$

The power plant can be connected to one building and that building in turn may be recursively connected to zero or more buildings.

Maximum depth of recursion represents most number of buildings powered by the power plant.

Using Full DFS/BFS, connected components of the graph can be computed in $O(n^4)$ time.

Then size of each connected component can be computed in $O(n^2)$ time and The power plant present in maximum size connected component should be the one where emergency generator should be installed.

Problem 5-3.

Construct a graph G, where nodes are the friends and vertices represents that two nodes of the edge short-circuit each other.

A **bipartite graph** is a graph whose vertices can be divided into two sets such that in each set, there is not an edge between any two pair of vertices of that set.

If the graph G is a bipartite graph, then the nodes can be divided into two sets, such that in each set, there is not an edge between any pair of vertices.

To check whether G is bipartite or not, the two coloring algorithm can be used which is as follows:

Modifying BFS,

Initialize two colors c_1 and c_2.
Assign color c_1 to the source vertex.
Then assign c_2 to the neighbors of source.
Assign c_1 to the neighbors of neighbors of source vertex and so on.

If for any vertex, there is a neighbor which is colored same as the current vertex, then the graph is not bipartite since there is an odd cycle.

The algorithm runs in $O(n)$ time since there are n edges in the graph.

Problem 5-4. *— After looking at solution —*

Construct a graph G where vertices are squares of the grid, i.e each vertex is represented by (r, c) $r, c \in \{0, ..., n\}$.

Connect vertices $v_1 = (r, c-1)$ and $v_2 = (r, c)$ when v_2 is owned by some farmer and v_1 and v_2 are owned by different farmers.
And connect vertices $v_3 = (r-1, c)$ and $v_4 = (r, c)$ when v_4 is owned by some farmer and v_3 and v_4 are owned by different farmers.

The graph now has the property that a path between any two vertices is a route on the map that does not tramble crops.

For each $euphris$ vertex (r, c), mark vertices $\{(r, c), (r+1, c), (r, c+1), (r+1, c+1)\}$ as E. Mark in the same way for $tigrates$ vertices as T.

Combine vertices marked as E in a supernode S. Now run BFS from S to vertices in T in $O(n^2)$ time.

Return the shortest path from S to T by traversing the parents.

$\because$ there are $O(n^2)$ edges and running BFS takes $O(n^2)$ time. Traversing back to find shortest path takes $O(n^2)$ time. $\therefore$ Total running time is $O(n^2)$.

Problem 5-5.

Problem 5-6.

(a)

(b)

(c)

(d) Submit your implementation to `alg.mit.edu`.