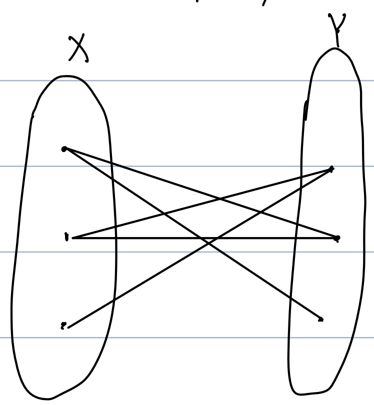
Intro to Algorithm — Homework 5

Discuss the hw with Siddha kilaru.

Q1. The algorithm can be do in DFS, which is in $O(|V|+|E|)$ time complexity. We are apply color to the vertices that have visited.



For every vertices i in X :
    do dfs(i)

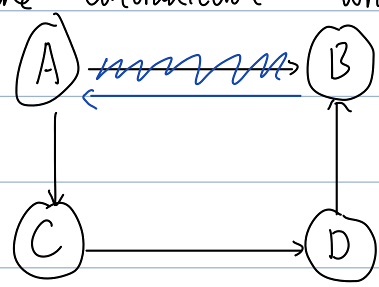
For every vertices j in Y :
    do dfs(j)

These steps will visit their adjacent vertices of i recursively and marked them.

Finally, we can run through the graph to check whether all the vertices are colored or not.
If all are colored, it is bipartite.
If not, it is not.

Q2

a. Prove by contradiction. Assume that the non-empty DAG has no sources. Sources is a vertex without incoming edge. We will fall in a cycle if the non-empty DAG has no sources. This is the contradiction with DAG. For example: we have 4 nodes and there is no sources.

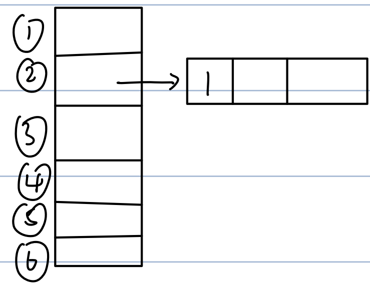


From the blue line, if the DAG has no sources, it is no longer a DAG.

b. In the adjacency matrix representation, if there is sources exist, the corresponding matrix[i][j] will be 1.
And we need to walk through the location to check the value whether 0 or 1.
In this case, it acts like 2D array. Hence, the time complexity is $O(|V|^2)$. V is the number of node.

C. In the adjacency list representation, we can construct the graph by doing: If node 2 has a incoming edge from node 1. Then I put node 1 in index 2. (As shown in the graph.

Like:



Then, the algorithm that need to reverse the graph is $O(|V|+|E|)$. However, if we are only looking at find a source, the time complexity is $O(|V|)$. Because, If we check on the index, if the index is empty, it has no source.

Q3. In the linear time algorithm to compute the neighbor degree for each vertex. I can make a loop to go over the vertex in the graph, then visited its adjancy vertex and marked them as visited. The the local variable 'time'++. Finally, we return time.

↓

time = 0

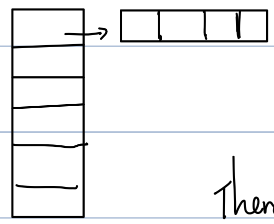
for all node:

visit the adjancy vertex and mark visit.

time ++.

Q4.

a. In order to compute a linear time algorithm to show the reachability weights for all vertices, we can reverse the graph into adjancy List representation. Each index store the incoming edges from other vertices. Then for every adjancy vertices v reachable to u, we calculate the weight.



Therefore, the total time complexity is $O(|V|+|E|)$ which is linear algorithm.

b. Assume the graph is a general direct graph with possible cycles, the linear time algorithm to find the reachability weight for all vertices is $O(|V|+|E|)$. We can apply DFS technic to find the reachability weight. For every unvisit spot, we do DFS to find the reachability. Just like we find the length from one node to others in a tree structure.