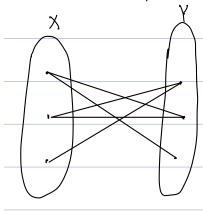
Discuss the hw with Siddha kilaru.

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



For every vertices in X:

For every vertices j in Y:

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 allored, it is bipartite.

If not, it is not.

Q 2	a. Prove by contradiction. Assume that the non-empty DA
	has no sources. Sources is a vertex without incoming edge. We
	will fall in a cycle if the non-empty DAG has no source.
	This is the contradiction with DAG. For example: we have 4
	A) modes and there is no sources.
	From the blue line, if the DAG h
	D no sources, it is no lunger a DAG
	b. In the adjacency matrix representation, if there
	sources exist, the corresponding matrix [:] [j] will be 1.
	And we need to walk through the location to check the
	value whether v or 1.
	In this case, it acts like 2D array. Hence, the time complexity is $O(V ^2)$. V is the number of nucle
	- Sur s

C. In the adjacency list representation, we can
construct the graph by doing: If nude 2 has a incoming edge
from node 1. Then I put node 1 in index 2. (As shown
Like: in the graph.
(1) (2) → 1 (3) (4) (4) (6) (6) (7) (7) (7) (7) (7) (7) (7) (7) (7) (7
6
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(IV). 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 verkex. I can make a hop to go over
the vertex in the graph, then visited its adjancy vertex and
marked them as visited. The the local variable 'time't.
Finally, we return time.
time = J
for all nude:
visit the adjancy verkex and mark visit.
time +t.

A. In order to compute a linear time algorithm to show the reachability weights for all vertices, we can reverse the graph into adjacy list representation. Each index stone the incoming edges from other vertices. Than for every adjacy vertices I reachable to u., we adolete the neight. Therefore, the total time complexity is O(1v1+1E1) which is linear algorithm. 5. 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(1v1+1E1). We can apply DFs technic to find the Preachability meight. For every unvisit sport, we do DFs to find the veachability. Just like we find the length from one node to others in a tree structure.	Q4.
reachability weights for all vertices, we can reverse the graph into adjacy list representation. Each index stone the incoming edges from other vertices. Then for every adjace, vertices I reachable to u, we calculate the neight. Therefore, the total time complexity is O(IVI+IEI) which is linear algorithm. 5. Assume the graph is a general direct graph with possible cycles the linear time algorithm to find the vecability weight for all vertices is O(IVI+IEI). We can apply DFS technic to find the Preachability weight. For every unvisit spot, we do DFS to DF	a. In order to compute a linear time algorithm to show the
reachable to u, we calculate the weight. Therefore, the total time complexity is O(101+1E1) which is linear algorithm. Linear algorithm to find the vecchability weight for all vertices is o(101+1E1). We can apply DFs technic to find the reachability weight for all vertices is the linear time algorithm. Preachability weight. For every universit sput, we do DFs to C, D find the reachability. Just like we find the length from one	
rechable to u, we calculate the weight. Therefore, the total time complexity is $O(V + E)$ which is linear algorithm. L. Assume the graph is a general direct graph with possible cycles the linear time algorithm to final the vecchobility weight for all vertices is $O(V + E)$. We can apply DFS technic to final the Preachability weight. For every unvisit spot, we do DFS to $O(V + E)$ that the vecchability of the use final the length from one	into adjancy List representation. Each index store the incoming edges from other
Therefore, the total time complexity is $O(V + E)$ which is linear algorithm. 5. Assume the graph is a general direct graph with possible cycles the linear time algorithm to find the vecability weight for all vertices is $O(V + E)$. We can apply DFS technic to find the Preachability weight. For every universit spot, we do DFS to $O(V + E)$ find the reachability. Just like we find the length from one	vertices. Then for every adjuncy vertices u
b. Assume the graph is a general direct graph with possible cycles the linear time algorithm to find the vecchability weight for all vertices is $O(V + E)$. We can apply DFs technic to find the P reachability weight. For every unvisit spot, me do DFS to $O(V + E)$ find the reachability. Just like we find the length from one	
b. Assume the graph is a general direct graph with possible cycles the linear time algorithm to find the vecchability weight for all vertices is $O(V + E)$. We can apply DFs technic to find the Prechability weight. For every unvisit spot, we do DFS to DFS t	Therefore, the total time complexity is O(v + E)
the linear time algorithm to find the vechability weight for all vertices is O(IVI+IEI). We can apply DFS technic to find the Preachability weight. For every unvisit spot, we do DFS to C), D find the reachability. Just like we find the length from one	which is linear algorithm.
	O(V + E). We can apply DFs technic to find the reachability weight. For every unvisit spot, we do DFS to