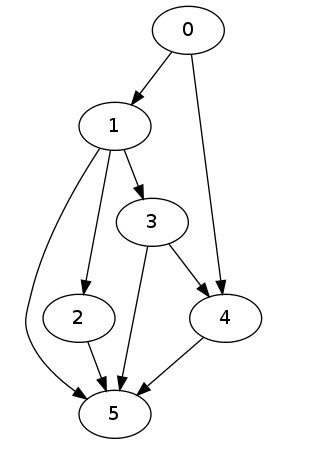
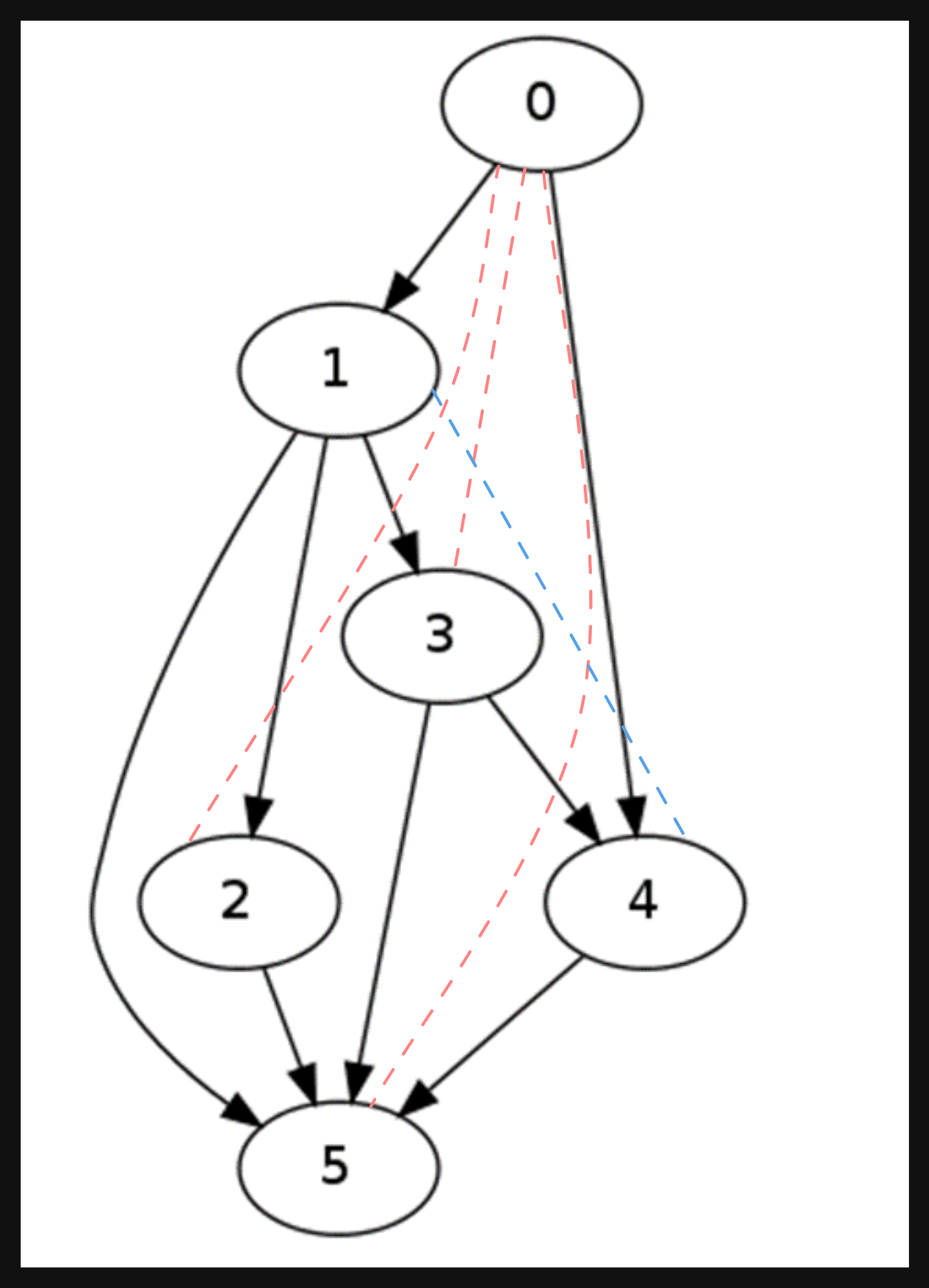
**Unit 3 Algorithmics**

**Submit Task – Week 8**

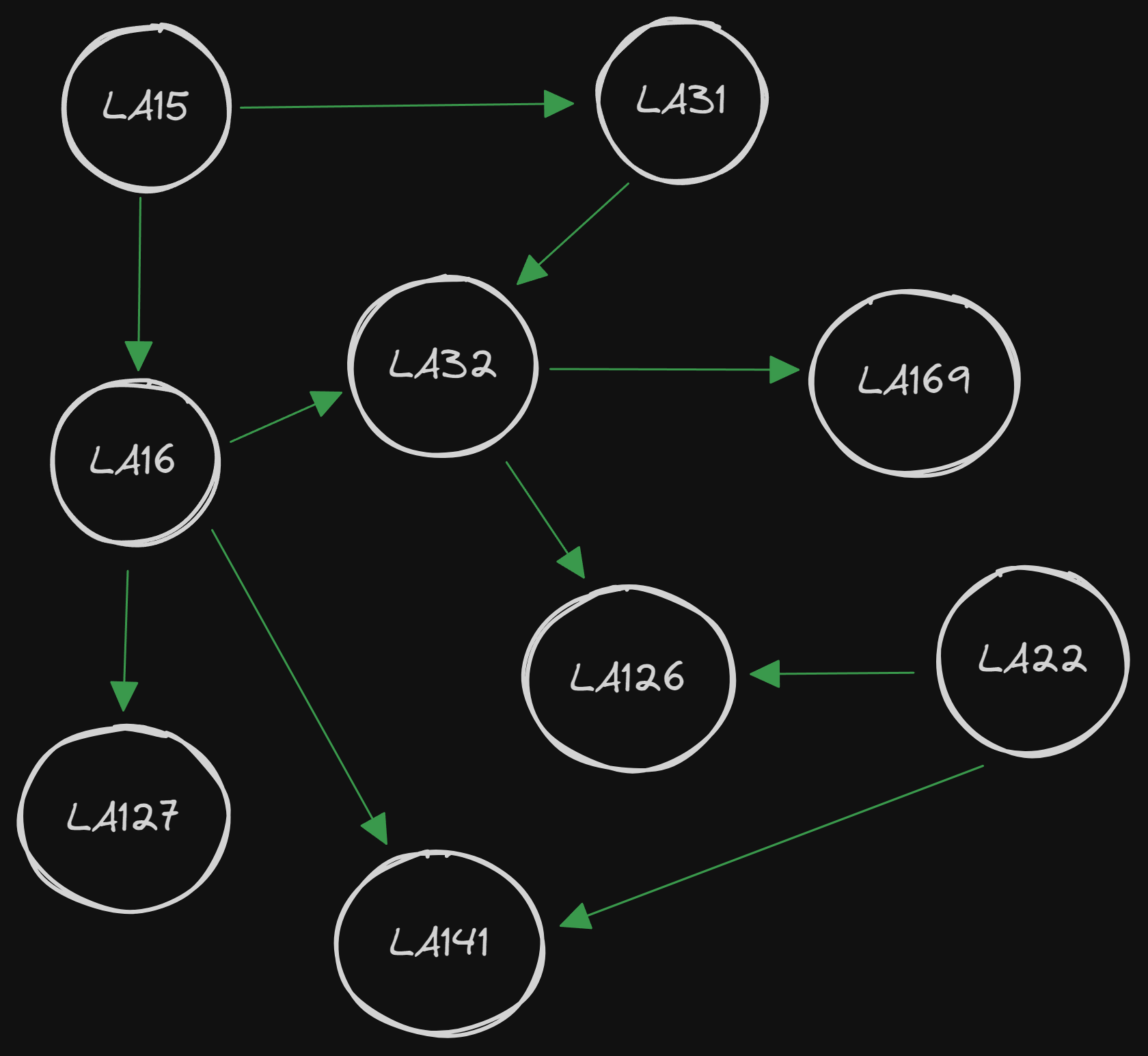
**Problem 1: Transitive Closure**



1. Consider the graph shown above, by adding extra “dotted” edges to the directed graph above show the Transitive Closure for this graph.
2. 
3. Bob loves foreign languages and wants to plan his course schedule for the following years. He is interested in nine language courses, with prerequisites as follows:

|  |  |
| --- | --- |
| **Language Course** | **Prerequisite course** |
| LA15 | none |
| LA16 | LA15 |
| LA22 | none |
| LA31 | LA15 |
| LA32 | LA16, LA31 |
| LA126 | LA22, LA32 |
| LA127 | LA16 |
| LA141 | LA22, LA16 |
| LA169 | LA32 |

1. Sketch a directed graph to find a sequence of courses that allows Bob to satisfy all the prerequisites.  Discuss any attributes or special features of the graph and give examples of how it would be used by Bob or other students to determine course prerequisites.



Bob can follow the directed edges to determine the order in which the courses should be taken. He can start with the courses with no incoming edges (LA15, LA 22) and then follow the arrows to the sink nodes (LA127, LA141, and LA169) which represent the final courses he can take after completing all pre-requisites. This graph is special as it is a directed acyclic graph.

1. If we use the Floyd-Warshall algorithm to compute the transitive closure of the directed graph from **part (b).** What information about the relationship between the courses does this transitive closure of the graph represent?

It shows all reachable courses from any given start node. The transitive closure tell you if the node you want to reach is reachable from a current node, and also gives you the shortest path to it (through this is not directly reflected in the transitive closure table and you have to program that in).

**Problem 2: Pseudocode Error**

Identify the problem(s) in this pseudocode for Floyd Warshall.

define FloydWarshall(graph):

n = number of vertices in the graph

distance = 2D array of size n x n

initialize distance to infinity for all pairs of vertices

for each vertex v:

distance[v][v] = infinity

for each edge (u, v) in graph:

distance[u][v] = weight of edge (u, v)

for i from 1 to n:

for j from 1 to n:

for k from 1 to n:

if distance[i][j] > distance[i][k] + distance[k][j]:

distance[i][j] = distance[i][k] + distance[k][j]

return distance

* It doesn’t initialize first node as 0 and instead sets all nodes to inf which wont make the program run atall.