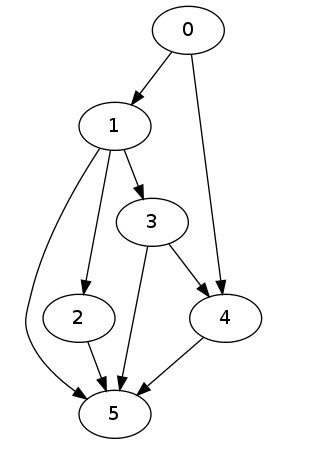
**3.8 Homework Task Submission**

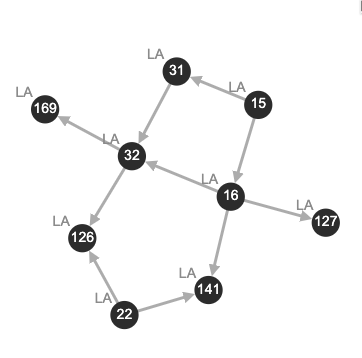
|  |  |
| --- | --- |
|  |  |
| **Answer all parts of this Homework task and submit your work into VSV Online as PDF and SnapApps/Edgy .xml or Python3/Trinket code files. This homework is required to demonstrate learning outcomes to a satisfactory standard.**  **For this Homework Submission 2 Files are expected:**  **• a PDF file with text responses to Problem 1 and Problem 2**  **• an exported coding file in .xml from SnapApps/Edgy or using Python3/Trinket addressing Problem 2, part e** |

**Problem 1: Transitive Closure** *(Outcome 1- Criteria 2)*



1. Consider the graph shown above, by adding extra “dotted” edges to the directed graph above show the Transitive Closure for this graph.
2. 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.
2. 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?

The transitive closure of a graph shows if a node is reachable from a given source node. In the context of the graph, the transitive closure shows what course you would be able to start with if you wanted to end up doing a particular course, showing if it “reachable” or not with the prerequisites. For example, if I wanted to do LA169, it would make sense for me to start with LA15, as LA169 is not “reachable” from LA22, which the transitive closure would show.

**Problem 2: Best First Search** *(Outcome 2 - Criteria 4)*

1. Describe in your own words how the “Best First Search” algorithm works to determine a path from one state to another state.

The “Best First Search” algorithm is a way of finding a path from a source node to a target node, which uses a heuristic function to evaluate possible nodes and decide which one to go to next. This greedy algorithm starts from a source node, and then uses the heuristic function to estimate the distance between the current node and goal state for each node. As such, it uses a min priority queue, with priorities assigned by the heuristic function to decide which node to go to next, which is then dequeued. As such, it will always find a path from the source state to the target state which may not always be the shortest, but it can provide a good approximation of such based on the heuristic function.

1. Write an algorithm in pseudocode that matches your English description from part **a).**

**While sourceState != targetState:**

**queue = createPriorityQueue(sorting: min)**

**For each neighbour of sourceState that has not been visited:**

**queue.enqueue(heuristicFunction(neighbour))**

**nextNode = queue.dequeue()**

**sourceState = sourceState + nextNode**

1. If you are given a map with several locations as shown below where there is a cost to visit each node and you were located at position “B”, describe how “Best First Search” would attempt to get to position “I”? Show all the edges traversed and nodes visited and the justification for each traversal decision.

Our heuristic function for this best first search could simply assign each node with the cost it takes to travel to it. As such, starting from node B, the best first search would first take us to node F, because it has a cost of 1. Below is how the algorithm would continue:

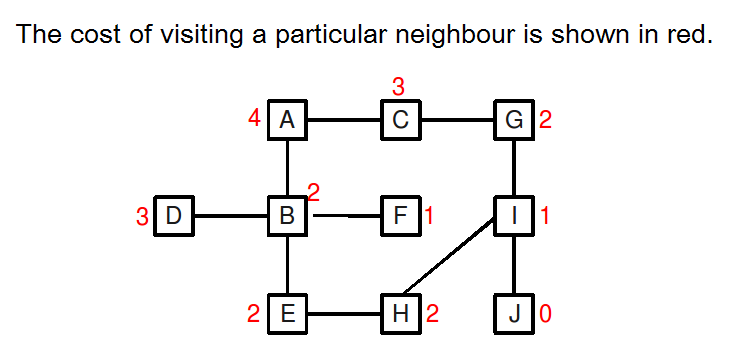
B – F, lowest cost of 1

F – B, lowest of 2

B – E, lowest cost of 2

E – H, lowest cost of 2

H – I, lowest cost of 1



1. What are the limitations of the “Best First Search” algorithm? In what cases may it not be the best algorithm to use to find the “Best” path from one location to another?

One limitation that was made evident by the previous example is that the algorithm is heavily dependant on the heuristic function and its ability to approximate the distance to the goal state. As we only took into consideration the cost to neighbours, we wasted time on suboptimal choices such as B to F. Also, if we want to find the shortest path, this algorithm may not be the most suitable as it not gaurenteed to do so.

1. Implement your “Best First Search” algorithm for this particular problem using the SnapApps/Edgy or Python3/Trinket coding environment.