IT-309 - Assignment #8 (A8): Graph Modelling and Critical Path Analysis

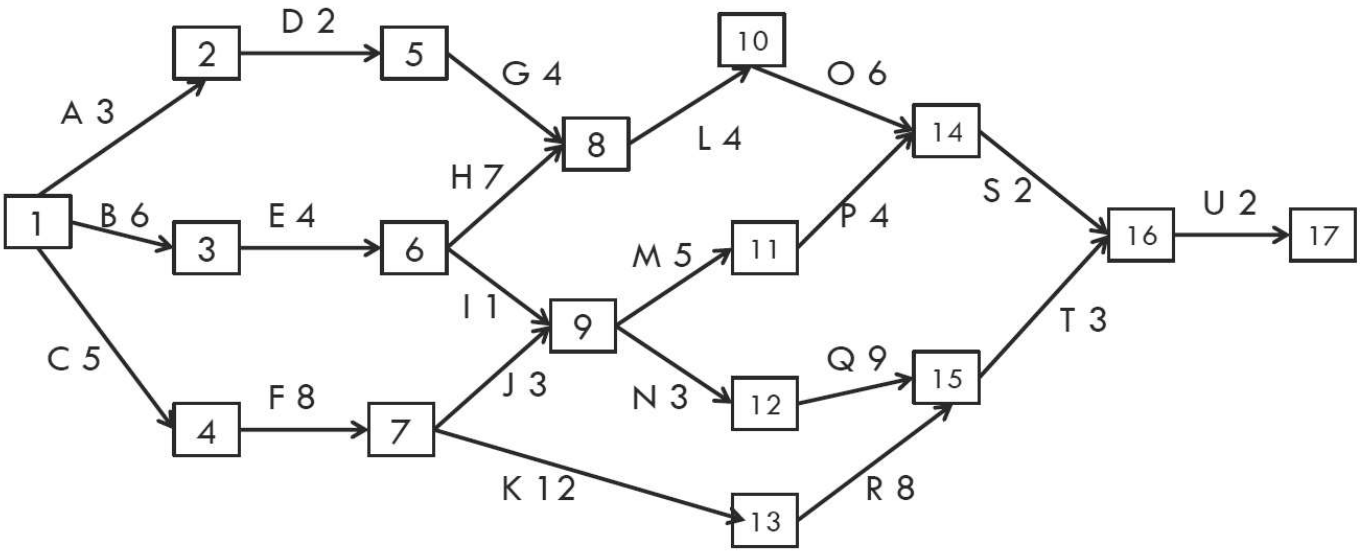
Assignment Given: 04/20/2022
Assignment Due: 05/06/2022, 11:59 pm

This assignment requires that you write code to process the schedule graph you loaded in A7 and has two different parts, each worth some number of points totally to 40 points. See the rubric table below.

(1) Using the graph model loaded in A7 (or an equivalent), calculate the longest path from the project start milestone (1) to the end milestone (17). I've coded this myself using a modification of the Breadth First Search algorithm presented in week #12. One modification would be to use the edge lengths (costs or weights) to calculate path lengths rather than just using 1 for each. Another is to be sure to calculate the **longest** path between two nodes. The result to be displayed is the critical path size recorded in milestone 17. The model's vertices or nodes are the project milestones. The edges are project activities or tasks. In the diagram below, milestones 2 and 5 form an edge with identifier "D" with length/cost/weight of 2. Using "MS" as the prefix for milestones that edge could be rendered as the triple (MS2, MS5, 2), or as the quadruple (MS2, MS5, "D", 2) if the activity identifier is to be included (which is the way I coded it).

(2) Continue with the above by modifying it to record the actual activities along the critical path. The result will be a list of ordered critical path activities through the graph (e.g., [A, B, E, …U]). You've already done this manually on paper, now you need to write code to do it. Defining edges using the quadruple approach should make this part easier.

For reference here is the schedule activity network graph from A7:



What and where to submit:

Submit the each of the above parts in a single Jupyter notebook. Start by using the Graph code from A7 to model the graph and add the traversal code. You may modify the Graph class if needed. For example, the nested Edge class would need to be modified to implement Edge objects as a quadruple as described above. You may also consult the textbook and other sources for ideas, but do as much of this as you can yourself. Label each part with a Markdown cell immediately preceding the part.

You may use simplified data structures if you want. For example, you could use the Queue ADT data structure presented earlier in the semester, but using a simple Python list would probably suffice. The 'enqueue' operation would be a simple

'append', which adds the item to the end of the list. 'Dequeue' would be implemented by removing the first element. Also, the priority queue could be modelled using a function like 'minPQ' that takes in a PQ as parameter and returns the minimum value in it. The PQ itself can be a Python list that you merely append to. The function sorts through and returns the min, but that can be done in about 12-15 lines of code. No need to use the full PQ ADT.

How the assignment will be assessed

The code will be visually inspected and executed. Each of the three tasks will be evaluated and graded with the maximum values listed in the table below.

Item	Assessment Description	Max Value
Calculate the longest path	Deliver Python code that calculates the longest path from milestone 1 to milestone 17 – that is the critical path (the longest through the graph and determines the minimum time needed to do a project) The result to be displayed is the longest path size from the starting milestones (#1) to the end milestone (#17).	30
Identify the critical path	Deliver Python code to identify the activities on the critical path. The display output is the list of activities on the critical path. Example: A, B, E, H,…U. Those activities would have to be managed closely in a project since any slippage in one of them would delay the project's delivery date. Determine and display the milestones along the path and from that determine the path's activities.	10
Total		**40**