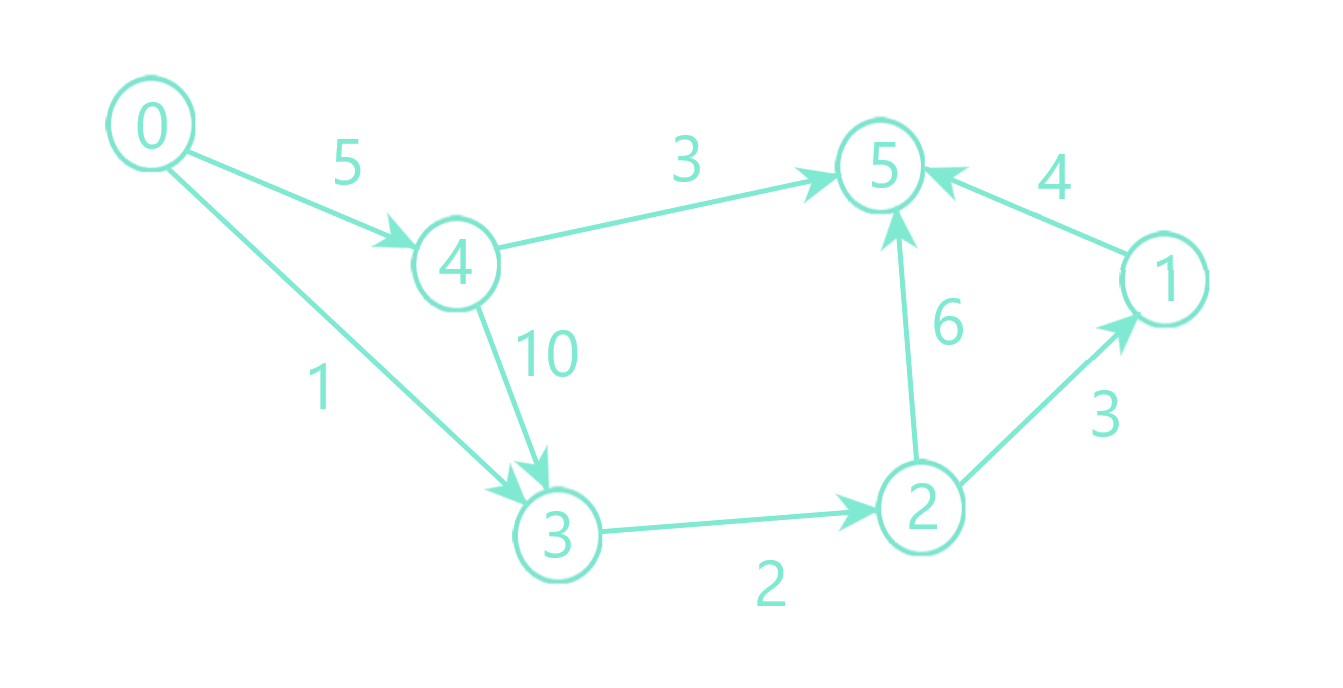
The source code for topological\_sort\_predecessor\_counting(graph) can be found in DAG.py.



For this graph the output is: 0 4 3 2 1 5 and the highest cost path between 0 and 5 is: 0-> 4-> 3-> 2-> 1->5 Cost: 24

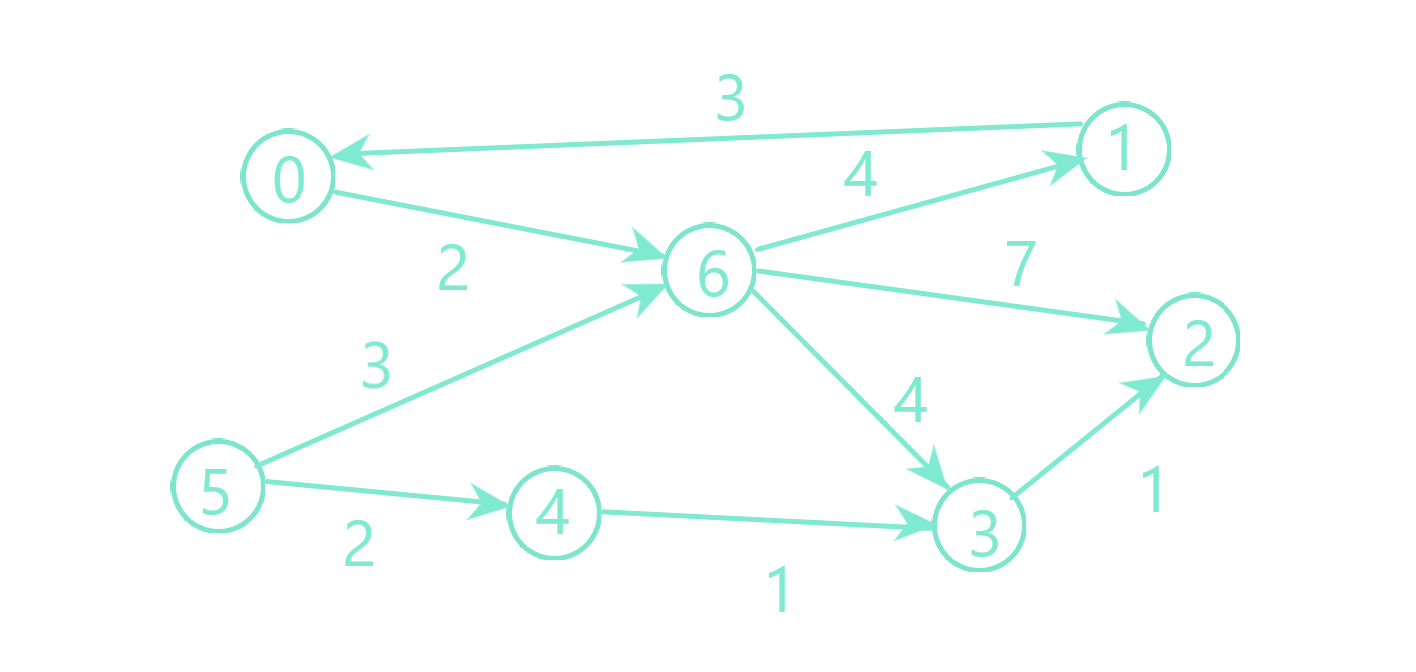
Topological order:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X | Queue | Predecessors | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 |
|  | 0 | 0 | 1 | 1 | 2 | 1 | 3 |
| 0 | 4 | 0 | 1 | 1 | 1 | 0 | 3 |
| 4 | 3 | 0 | 1 | 1 | 0 | 0 | 2 |
| 3 | 2 | 0 | 1 | 0 | 0 | 0 | 2 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 2 |
| 1 | 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | - | 0 | 0 | 0 | 0 | 0 | 0 |

Highest cost path:

Topological order: 0 4 3 2 1 5

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | Distance | | | | | | Predecessors | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 0 | 1 | 2 | 3 | 4 | 5 |
| Start | 0 | -inf | -inf | -inf | -inf | -inf | -1 | - | - | - | - | - |
| 0 | 0 | -inf | -inf | 1 | 5 | -inf | -1 | - | - | 0 | 0 |  |
| 4 | 0 | -inf | -inf | 15 | 5 | 8 | -1 | - | - | 4 | 0 | 4 |
| 3 | 0 | -inf | 17 | 15 | 5 | 8 | -1 | - | 3 | 4 | 0 | 4 |
| 2 | 0 | 20 | 17 | 15 | 5 | 23 | -1 | 2 | 3 | 4 | 0 | 2 |
| 1 | 0 | 20 | 17 | 15 | 5 | 24 | -1 | 2 | 3 | 4 | 0 | 1 |



This graph contains a cycle so the output of both functions will be None meaning a cycle has been found.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | Queue | Predecessors | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  | 5 | 1 | 1 | 2 | 2 | 1 | 0 | 2 |
| 5 | 4 | 1 | 1 | 2 | 2 | 0 | 0 | 1 |
| 4 | - | 1 | 1 | 2 | 1 | 0 | 0 | 1 |

The algorithm stops because the queue is empty, but not all the vertices were processed, therefore there is a cycle.