**1. Answer:**

|  |
| --- |
| Algorithm initResult**(**G**)**  S**<-** **new** Empty Sequence  Algorithm preComponentVisit**(**G**,** v**)**  S**.**insertLast**(**v**)**  Algorithm result**(**G**)**  **return** S**;** |

**2. Answer:**

|  |  |  |
| --- | --- | --- |
| A | Algorsithm BFS**(**G**)**  Input graph G  Output labeling of the edges and partition of the vertices of G  initResult**(** G **)**  **for** all u of G**.**vertices**()**  setLabel**(**u**,** UNEXPLORED**)**  **for** all e of G**.**edges**()**  setLabel**(**e**,** UNEXPLORED**)**  **for** all v of G**.**vertices**()**  **if** getLabel**(**v**)** **=** UNEXPLORED  preComponentVisit**(**G**,** v**)**  BFS**(**G**,** v**)**  postComponentVisit**(**G**,** v**)**  result**(**G**)** | Algorithm BFS**(**G**,** s**)**  L **<-** **new** empty sequence  L**.**insertLast**(**s**)**  setLabel**(**s**,** VISITED**)**  startVertexVisit**(**v**)**  **while** **!**L**.**isEmpty**()** **do**  v **<-** L**.**remove **(**L**.**first**())**  **for** all e of G**.**incidentEdges**(**v**)** **do**  **if** getLabel**(**e**)** **=** UNEXPLORED then  w **<-** opposite**(**v**,**e**)**  **if** getLabel**(**w**)** **=** UNEXPLORED then  preDiscoveryTraversal**(**G**,** v**,** e**,** w**)**  setLabel**(**e**,** DISCOVERY**)**  setLabel**(**w**,** VISITED**)**  L**.**insertLast**(**w**)**  postDiscoveryTraversal**(**G**,** v**,** e**,** w**)**  **else**  setLabel**(**e**,** CROSS**)**  backTraversal**(**G**,** v**,** e**,** w**)**  finishVertexVisit**(**G**,** v**)** |
| B | Algorithm findPath**(**G**,** u**,** v**)**  S **<-new** empty stack **{**S is a subclass field**}**  z**<-**v **{**z is a subclass field **&** is the target vertex**}**  BFS**(**G**,**u**)**  **return(**path**)**    Algorithm startVertexVisit**(**v**)**  S**.**push**(**v**)**  Algorithm preDiscoveryTraversal**(**G**,**v**,**e**,**w**)**  **if** **!**pathFound then  S**.**push**(**e**)**  S**.**push**(**w**)** | Algorithm postDiscoveryTraversal**(**G**,**v**,**e**,**w**)**  **if** w **=** z then**{**z is a subclass varibale and target**}**  pathFound **=** **true**    Algorithm finishVertexVisit**(**G**,** v**)**  **if** **!**pathFound then  **return** no\_such\_path  **return** S**.**elements**()** |
| C | Algorithm startVertexVisit**(**v**)**  **if** **!**cycleFound then  S**.**push**(**v**)**  Algorithm preDiscoveryTraversal**(**G**,**v**,**e**,**w**)**  **if** **!**cycleFound then  S**.**push**(**e**)**  S**.**push**(**w**)**  Algorithm backTraversal**(**G**,** v**,** e**,** w**)**  **if** **!**cycleFound then  cycle**<-new** empty sequence  cycle**.**insertLast**(**w**)**  **while** o **!=** w **do**  o**<-**S**.**pop**()**  cycle**.**insertLast**(**o**)**  cycleFound **<-** **true**  Algorithm finishVertexVisit**(**G**,** v**)**  **while** **!**S**.**empty**()**  S**.**pop**()**  **if** **!**cycleFound then  **return** no\_cycle\_found  **else**  **return** cycle |  |
|  | No**.**  Reason**:** DFS goes into the deep of a node and there is no guarantee that that path will be the minimum number of edges and it is hard to find minimum number of edges during backtracking**.** | |

**3. Answer:**

|  |  |
| --- | --- |
| Algorithm DijkstraDistances**(**G**,** s**)**  Q **<-new** heap**-**based priority queue  initGraph**(**G**)**  **for** all v of G**.**vertices**()** **do**  **if** v **=** s  setDistance**(**v**,** 0**)**  **else**  setDistance**(**v**,** INF**)**  l**<-**Q**.**insert**(**getDistance**(**v**),** v**)**  setLocator**(**v**,**l**)**  startVertexVisit**(**G**,**s**)**  **while** **!**Q**.**isEmpty**()** **do**  u **<-** Q**.**removeMin**()**  **for** all e of G**.**incidentEdges**(**u**)**  **{** relax edge e **}**  z **<-** G**.**opposite**(**u**,**e**)**  r **<-** getDistance**(**u**)** **+** weight**(**e**)**  **if** r **<** getDistance**(**z**)** then  preUpdate**()**  setDistance**(**z**,**r**)**  Q**.**replaceKey**(**getLocator**(**z**),**r**)**  postUpdate**(**G**,**u**,**e**,**z**)**  finishVertexVisit**(**G**,**s**)** | Algorithm findPath**(**G**,** u**,** v**)**  S **<-new** empty stack **{**S is a subclass field**}**  z**<-**v **{**z is a subclass field **&** is the target vertex**}**  DijkstraDistances**(**G**,**u**)**  **return(**path**)**  Algorithm startVertexVisit**(**G**,**s**)**  setParent**(**s**,**NULL**)**  Algorithm postUpdate**(**G**,**u**,**e**,**z**)**  setParent**(**z**,**e**)**  Algorithm finishVertexVisit**(**G**,**s**)**  path**<-**empty sequence  u **<-** z**{**z is a subclass varibale and target**}**  path**.**insertLast**(**u**)**  **while** u **!=** s **do**  u**<-**getParent**(**u**)**  path**.**insertLast**(**u**)** |

**4.Answer:**

|  |
| --- |
| Algorithm initResult**(**G**)**  levelNo **<-** 1  Algorithm startVertexVisit**(**v**)**  setLevel**(**v**,**levelNo**){**levelNo is a subclass field**}**  Algorithm postComponentVisit**(**G**,**v**)**  levelNo **<-** levelNo **+** 1 |