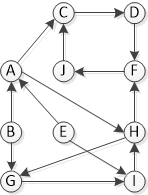
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| C:\Users\saif\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\final design.jpg | **Course:** | **Design & Analysis of Algorithms** | **Course Code:** | **CS2009** |
| **Program:** | **BS (Computer Science)** | **Semester:** | **Spring 2023** |
| **Duration:** | **15 Minutes** | **Total Marks:** | **10** |
| **Paper Date:** | **30-March-2023** | **Weight:** | **4** |
| **Section:** | **G** | **Page(s):** | **1** |
| **Exam:** | **Quiz 4** | **Reg. No.** |  |
| **Instruction/Notes:** |  | | | |

**Question 1: CLO , [10 marks]**

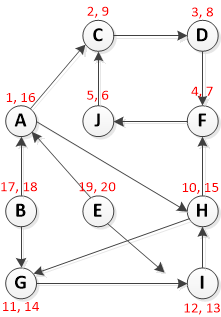
Find strongly connected components and the component graph.



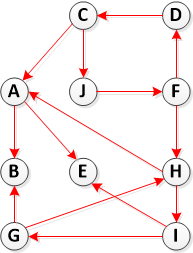
Solution:

**Step 1: Call DFS(G) to compute finishing times f[u] for each vertex u**

Running DFS starting on vertex A:



**Step 2: Compute Transpose(G)**



**Step 3. Call DFS(Transpose(G)), but in the main loop of DFS, consider the vertices in order of decreasing f[u] (as computed in step 1)**

vertices in order of decreasing finishing times values:

{E, B, A, H, G, I , C, D, F ,J}

So at this step, we run DFS on G^T but start with each vertex from above list:

* DFS(E): {E}
* DFS(B): {B}
* DFS(A): {A}
* DFS(H): {H, I, G}
* DFS(G): remove from list since it is already visited
* DFS(I): remove from list since it is already visited
* DFS(C): {C, J, F, D}
* DFS(J): remove from list since it is already visited
* DFS(F): remove from list since it is already visited
* DFS(D): remove from list since it is already visited

**Step 4: Output the vertices of each tree in the depth-first forest of step 3 as a separate strong connected component.**

So, we have five strongly connected components: {E}, {B}, {A}, {H, I, G}, {C, J, F, D}