Tarjan’s algorithm for finding strongly-connected components is given on the other side of this page. This question refers to the stack $S$ that the algorithm uses.

When the algorithm is applied to a specific graph, the stack $S$ evolves as shown below. The stack is shown after every case where a vertex or subset of vertices is added or removed from the stack.

Show a graph that causes the stack to evolve in this way.

Show the $d$ labels that the algorithm assigns.

$S = A$
$S = AB$
$S = ABC$
$S = AB$
$S = ABD$
$S = A$
$S = \phi$

Answer: From $S$, the strongly-connected components are $\{C\}$, $\{B,D\}$, $\{A\}$. 

![Graph Diagram]

**Diagram:**
- Nodes: A1, B2, C3, D4
- Edges: A1 to B2, B2 to C3, C3 to D4, D4 to A1, A1 to D4, B2 to D4
Tarjan’s algorithm:

(1) Mark all the edges "unused".
   For every \( v \in V \), let \( d[v] = 0 \) and \( \Pi(v) = NULL \).
   Empty \( S \).
   Set \( i = 0 \) and \( v = s \).

(2) Set \( i = i + 1 \), \( d[v] = i \), \( L[v] = i \), and put \( v \) on \( S \).

(3) If there are no unused incident edges from \( v \) then go to Step 7.

(4) Select an unused incident edge \( e = (v, u) \).
   Mark \( e \) "used".
   If \( d[u] = 0 \) then set \( \Pi[u] = v \), \( v = u \) and go to Step 2.

(5) If \( d[u] > d[v] \) (\( e \) is a forward edge), go to Step 3.
   Else, if \( u \) is not on \( S \) (\( u \) and \( v \) do not belong to the same component), go to Step 3.

(6) \((d[u] < d[v] \) and both vertices are in the same component)
   Set \( L[v] = \min \{ L[v], d[u] \} \) and go to Step 3.

(7) If \( L[v] = d[v] \) then delete all the vertices from \( S \) down to and including \( v \); these vertices form a component.

(8) If \( \Pi[v] \) is defined then set
   \( L[\Pi[v]] = \min \{ L[\Pi[v]], L[v] \} \),
   \( v = \Pi[v] \),
   and go to Step 3.

(9) \((\Pi[v] \) is undefined)
   If there is a vertex \( u \) for which \( d[u] = 0 \) then set \( v = u \) and go to Step 2.

(10) (all the vertices have been explored)
    Stop.
Tarjan’s algorithm for finding strongly-connected components is given on the other side of this page. This question refers to the stack $S$ that the algorithm uses. When the algorithm is applied to a specific graph, the stack $S$ evolves as shown below. The stack is shown after every case where a vertex or subset of vertices is added or removed from the stack.

Show a graph that causes the stack to evolve in this way.

Show the $d$ labels that the algorithm assigns.

$S = A$
$S = AB$
$S = ABC$
$S = AB$
$S = A$
$S = AD$
$S = \phi$

Answer: From $S$, the strongly-connected components are \{C\}, \{B\}, \{A,D\}. 

A 1  
B 2  
C 3  
D 4
Tarjan’s algorithm:

1. Mark all the edges "unused".
   For every \( v \in V \), let \( d[v] = 0 \) and \( \Pi(v) = NULL \).
   Empty \( S \).
   Set \( i = 0 \) and \( v = s \).

2. Set \( i = i + 1 \), \( d[v] = i \), \( L[v] = i \), and put \( v \) on \( S \)

3. If there are no unused incident edges from \( v \) then go to Step 7.

4. Select an unused incident edge \( e = (v, u) \).
   Mark \( e \) "used".
   If \( d[u] = 0 \) then set \( \Pi[u] = v \), \( v = u \) and go to Step 2.

5. If \( d[u] > d[v] \) (\( e \) is a forward edge), go to Step 3.
   Else, if \( u \) is not on \( S \) (\( u \) and \( v \) do not belong to the same component), go to Step 3.

6. \((d[u] < d[v] \) and both vertices are in the same component\)
   Set \( L[v] = \min \{ L[v], d[u] \} \) and go to Step 3.

7. If \( L[v] = d[v] \) then delete all the vertices from \( S \) down to and including \( v \); these vertices form a component.

8. If \( \Pi[v] \) is defined then set
   \( L[\Pi[v]] = \min \{ L[\Pi[v]], L[v] \} \),
   \( v = \Pi[v] \),
   and go to Step 3.

9. \((\Pi[v] \) is undefined\)
   If there is a vertex \( u \) for which \( d[u] = 0 \) then set \( v = u \) and go to Step 2.

10. \( \) (all the vertices have been explored)
    Stop.