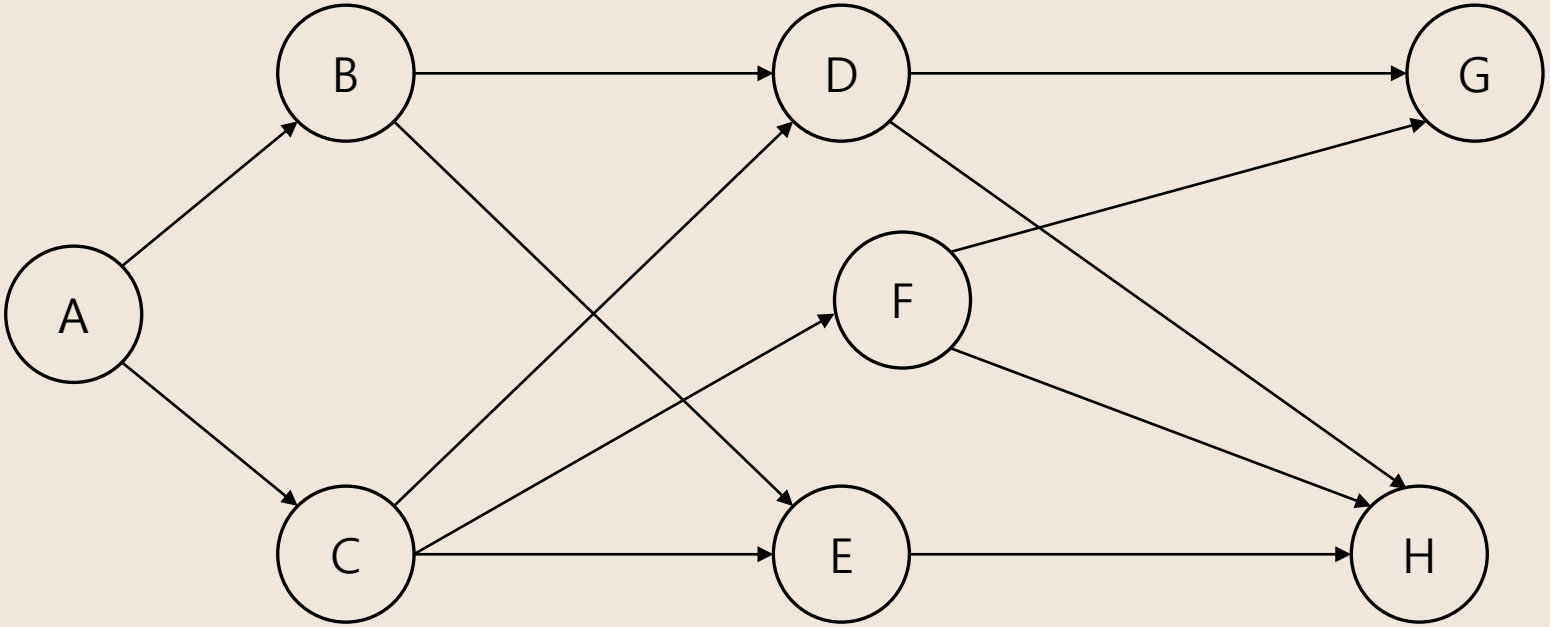
위상정렬(Topological Sorting)



- 주어진 DAG(Directed Acyclic Graph)에서
  - 모든 정점들을 일렬로 나열하되
  - u → v 이면 일렬로 나열된 순서에서 반드시 u 가 v 보다 먼저 나타나야 함

위상정렬(Topological Sorting)

위상 정렬 알고리즘

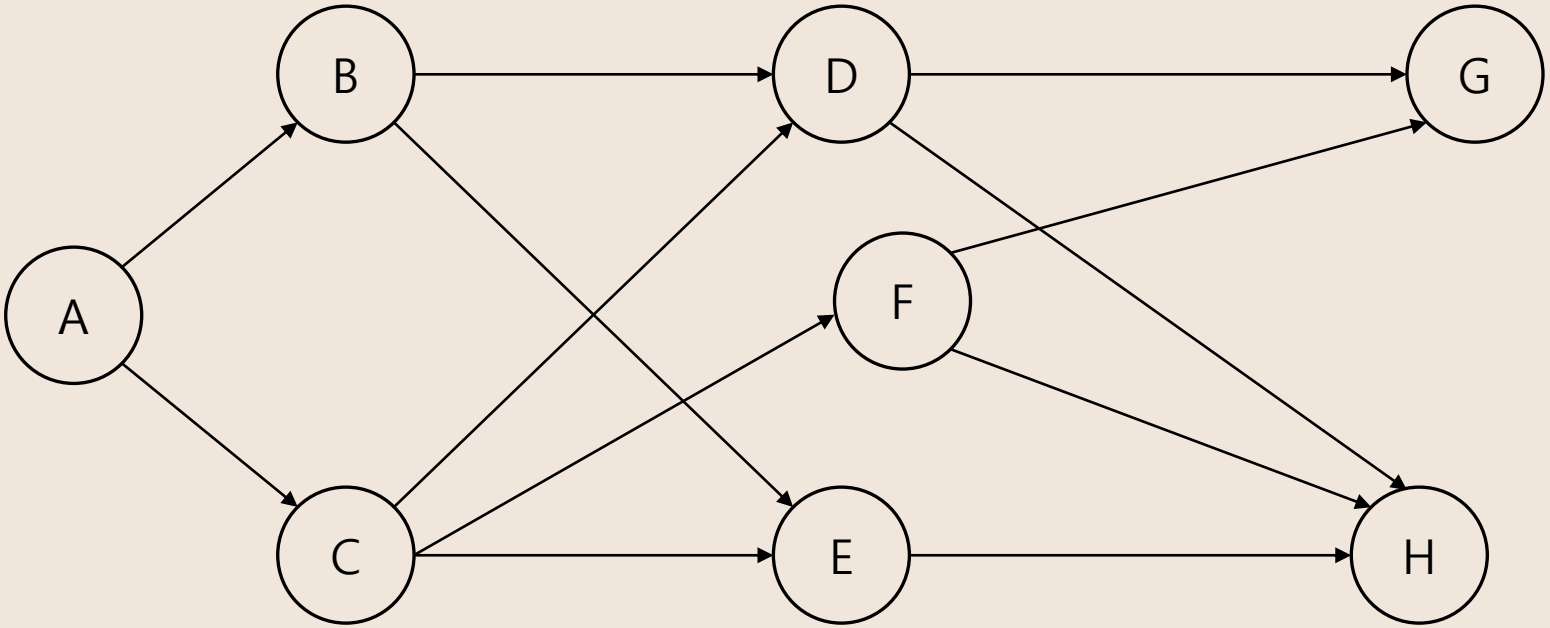
Input: G = (V, E)

V′ ← V; E′ G′ = ← E; G′ = (V′, E′)

V′ ≠ Φ 이 참인 동안 다음 단계를 반복

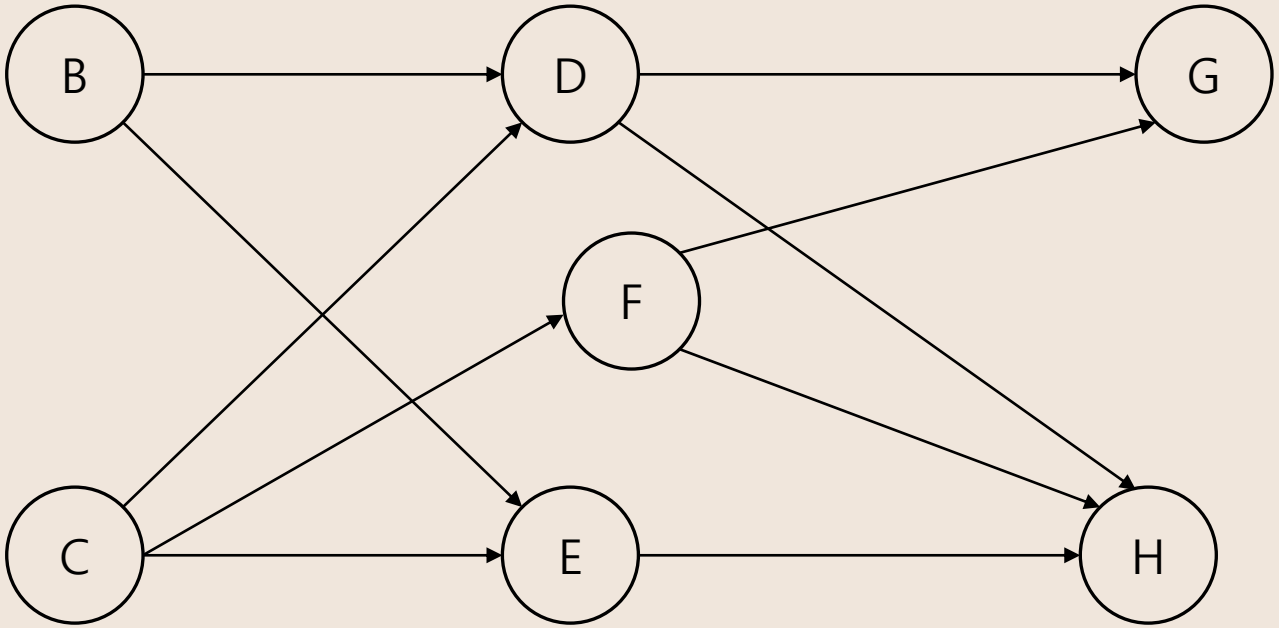
1. Indegree가 0인 정점 x를 선택한다. (DAG이므로 그런 정점이 반드시 존재)
2. x를 출력
3. V′ ← V′ – {x}
4. E′ ← E′ – {x에 연결된 outgoing edges}

위상정렬(Topological Sorting)



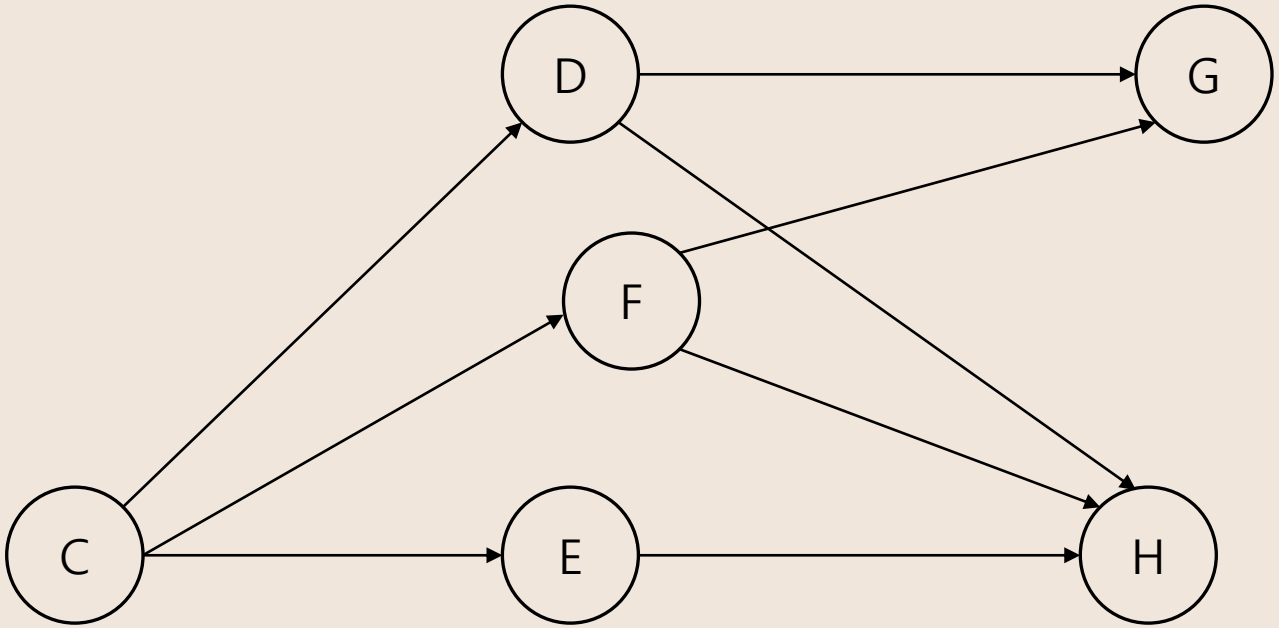
- 정점 A 가 유일하게 indegree가 0이므로 A를 출력하고 그래프 변경

위상정렬(Topological Sorting)



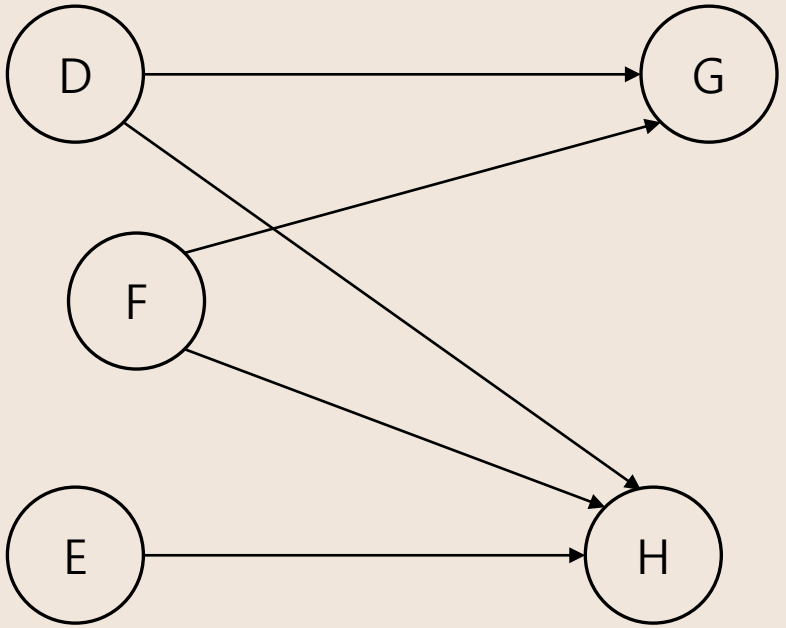
- 출력 상태: A
- 정점 B, C 의 indegree가 0이므로 둘 중 아무거나 출력 → B 선택

위상정렬(Topological Sorting)



- 출력 상태: A, B
- 정점 C 의 indegree가 0이므로 출력

위상정렬(Topological Sorting)



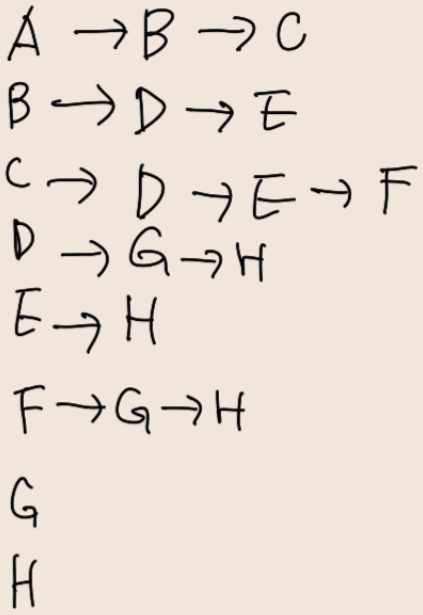
- 출력 상태: A, B, C
- 정점 D, E, F 의 indegree가 0이므로 셋 중 아무거나 출력 → F 선택

위상정렬(Topological Sorting)

또 다른 알고리즘

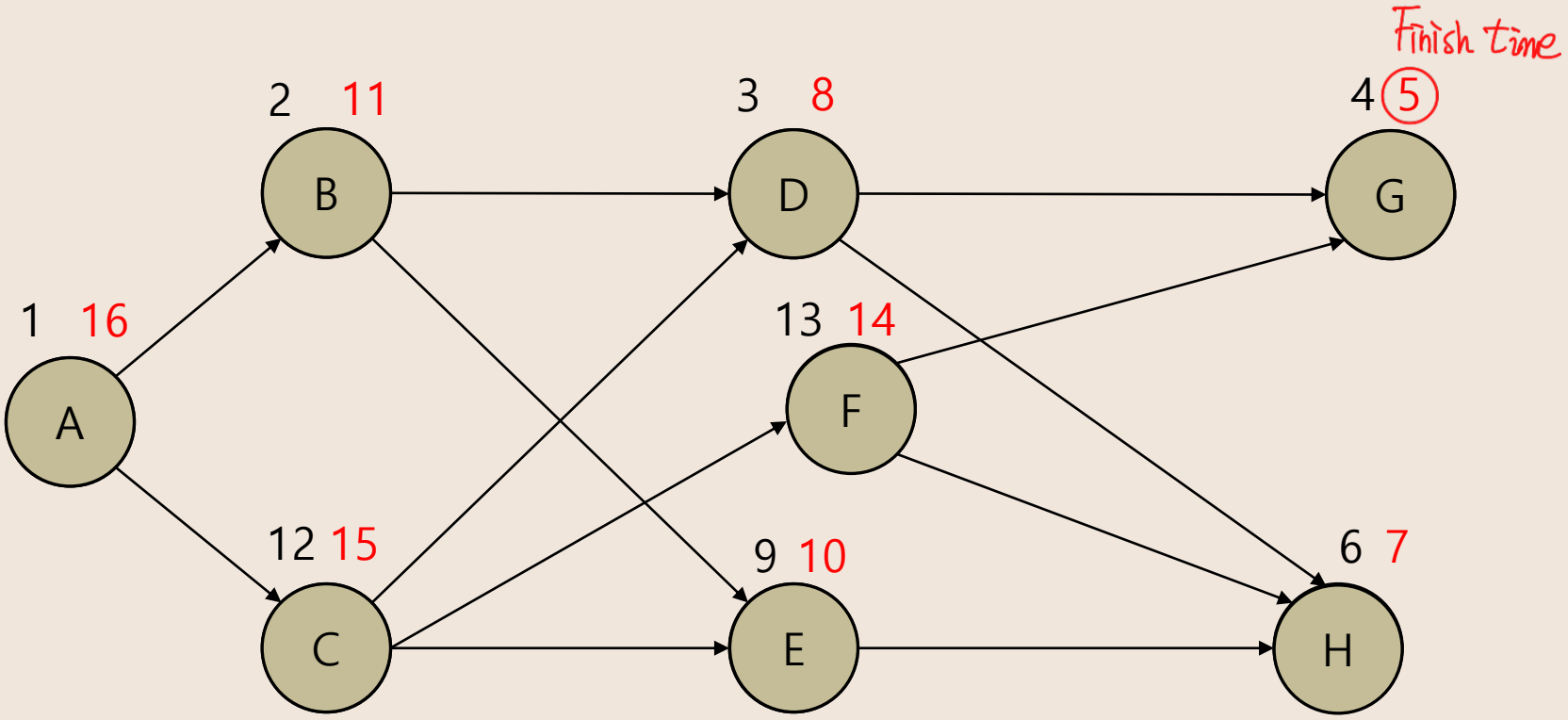
Input: G = (V, E)

1. DFS(S)를 호출, 각 정점 v의 종료시각 f[v]를 계산
2. 각 정점이 종료될 때, 그 정점을 list에 추가
3. list의 역순을 출력



위상정렬(Topological Sorting)

또 다른 알고리즘

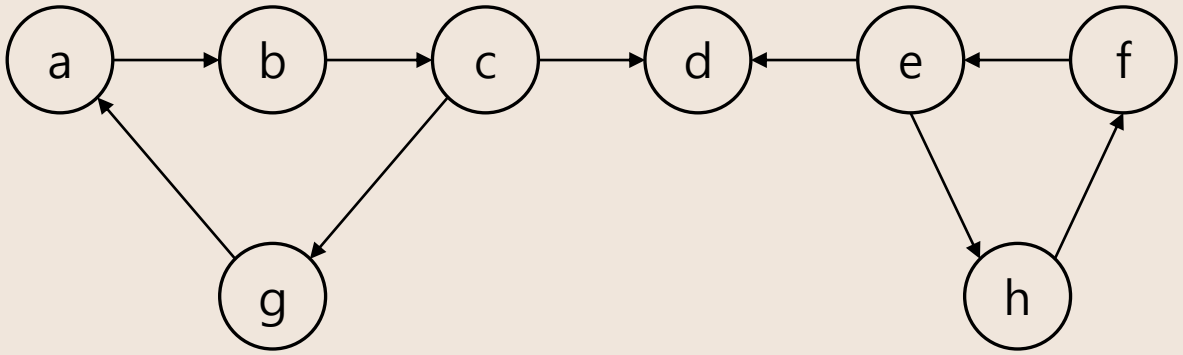


List: G H D E B F C A

위상정렬 결과: A C F B E D H G

SCC(Strongly Connected Component)

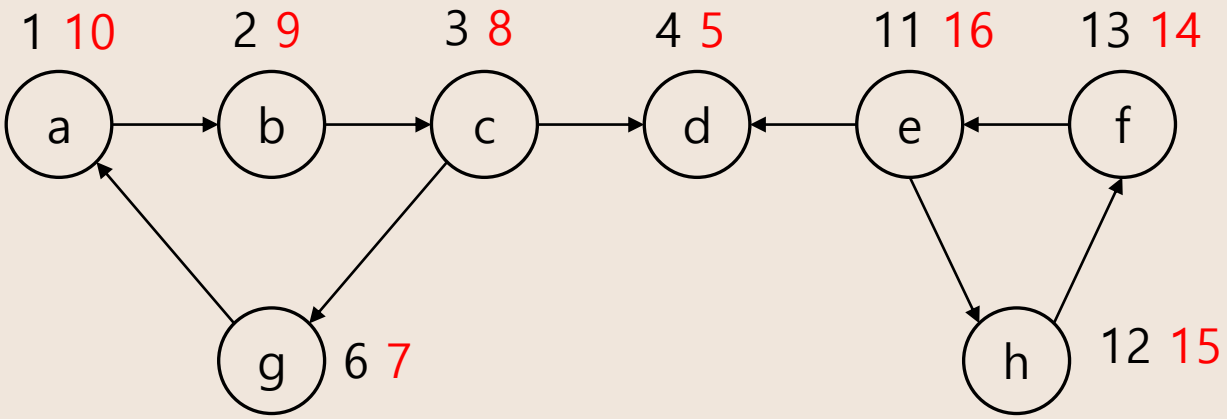
- {a, b, c, g}, {d}, {e, f, h} : 서로 연결되어 있음



- DFS-based linear time algorithm for SCC : O(n+m)

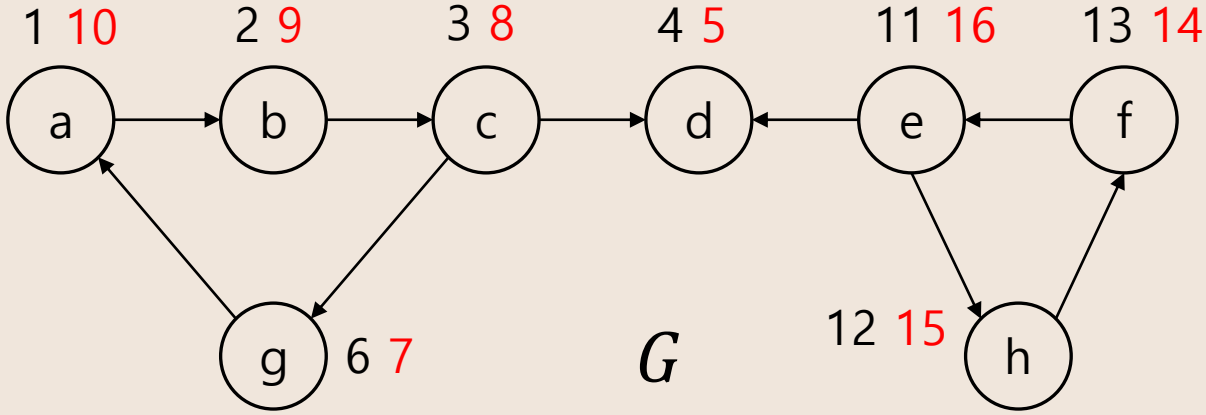
How to find SCCs?

1. G를 DFS로 순회하면서 finish time을 구한다



G

How to find SCCs?



2. G의 에지 방향을 반대로 한 G^T (transpose of G)에서 DFS를 이용해 순회
3. 단, 앞에서 구한 finish time의 역순을 고려하면서 DFS 적용
4. G^T 에서 도달가능한 것들의 집합은 SSC가 된다.