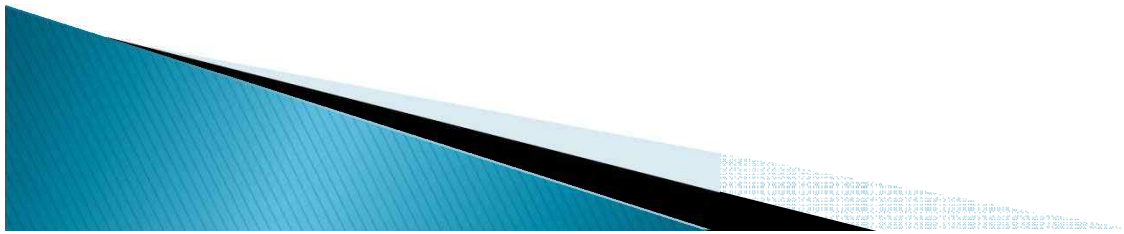
SCC

- Strongly Connected Component
- 단방향 그래프에서 A -> B 의 경로와 B -> A의 경로가 모두 존재하는 묶음을 말합니다.
- 이렇게 말하면 감이 잘 오지 않죠?



SCC

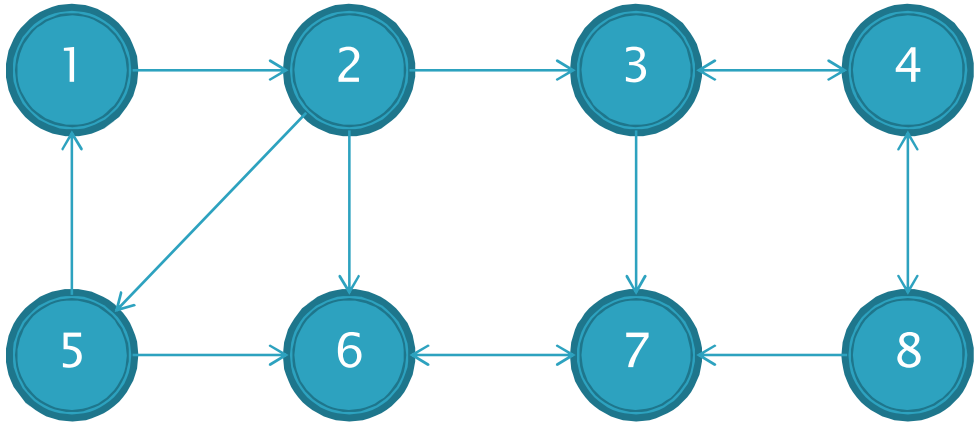
- 이와 같은 그래프에서

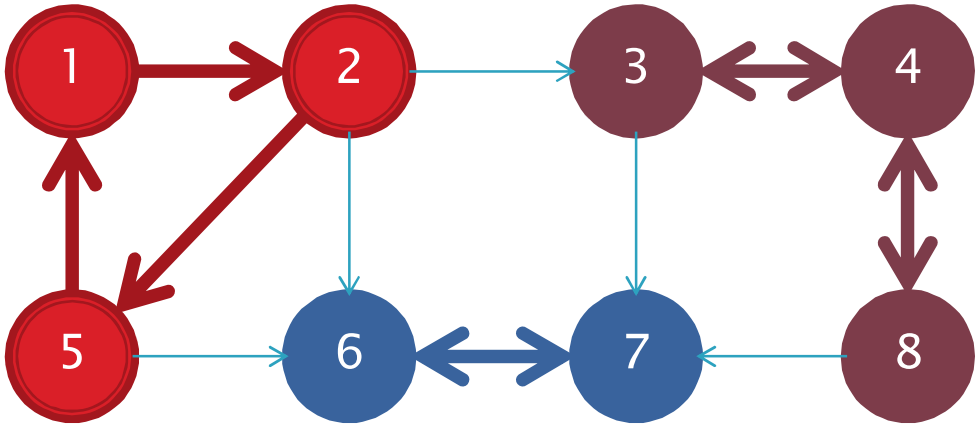
  아래와 같이 묶으면

  하나의 sub Graph

  에서 A->B와 B->A

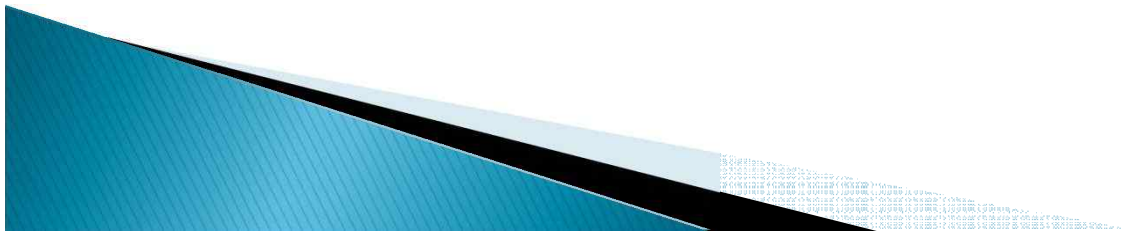
  경로 모두 존재하죠?





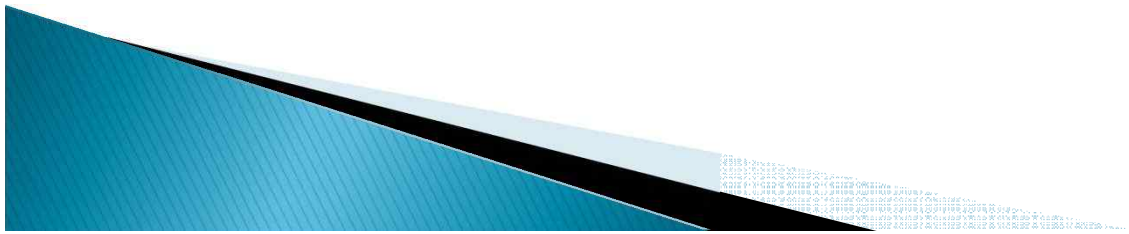
SCC

- 즉, 하나의 Sub Graph안에서 어디에서든 출발해도 다시 자기자신으로 돌아올 수 있습니다.
- 이를 Strongly Connected Component라고 합니다!
- 하나의 Graph에 있는 모든 SCC를 어떻게 찾을지?



SCC

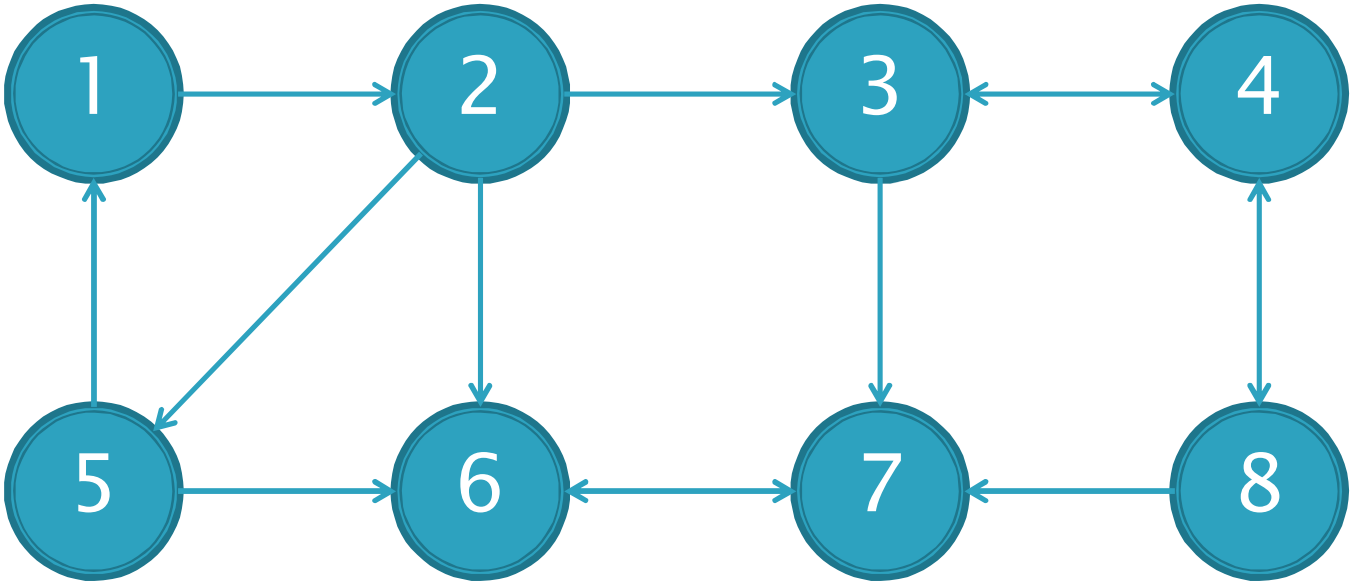
- …저도 알고리즘 이름은 잘 모르겠네요 ㅠ
- 방법은 꽤 간단한 편입니다!



SCC

- 1. 임의의 점에서 DFS를 수행한다. 단, DFS를 수행하면서 그 노드로 들어간 Time과 빠져나오는 Time을 모두 기록한다.
- 2. DFS가 끝났다면 모든 간선의 방향을 바꾼다.
- 3. 빠져나오는 Time이 높은 순서대로 DFS 수행.
- ㅋㅋ뭔말인지 모르겠죠! @_@ !

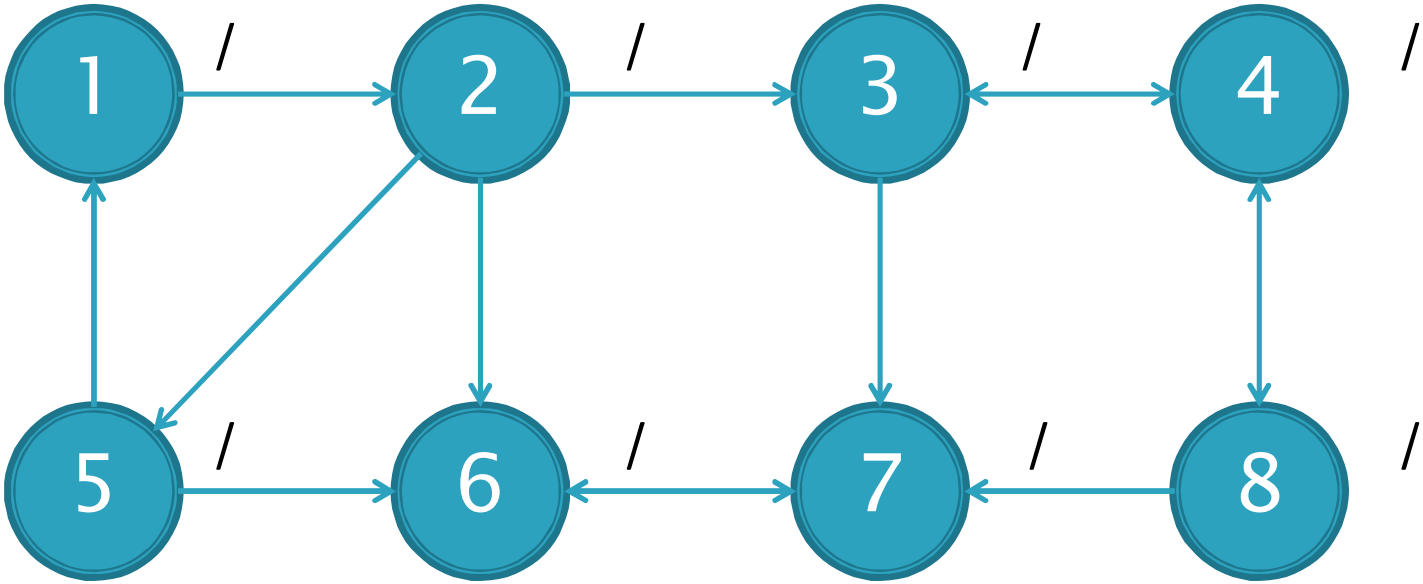
SCC



(1) 부터 DFS를 시작합시다. 들어가는 Time과

빠져나오는 Time을 모두 기록해볼까요?

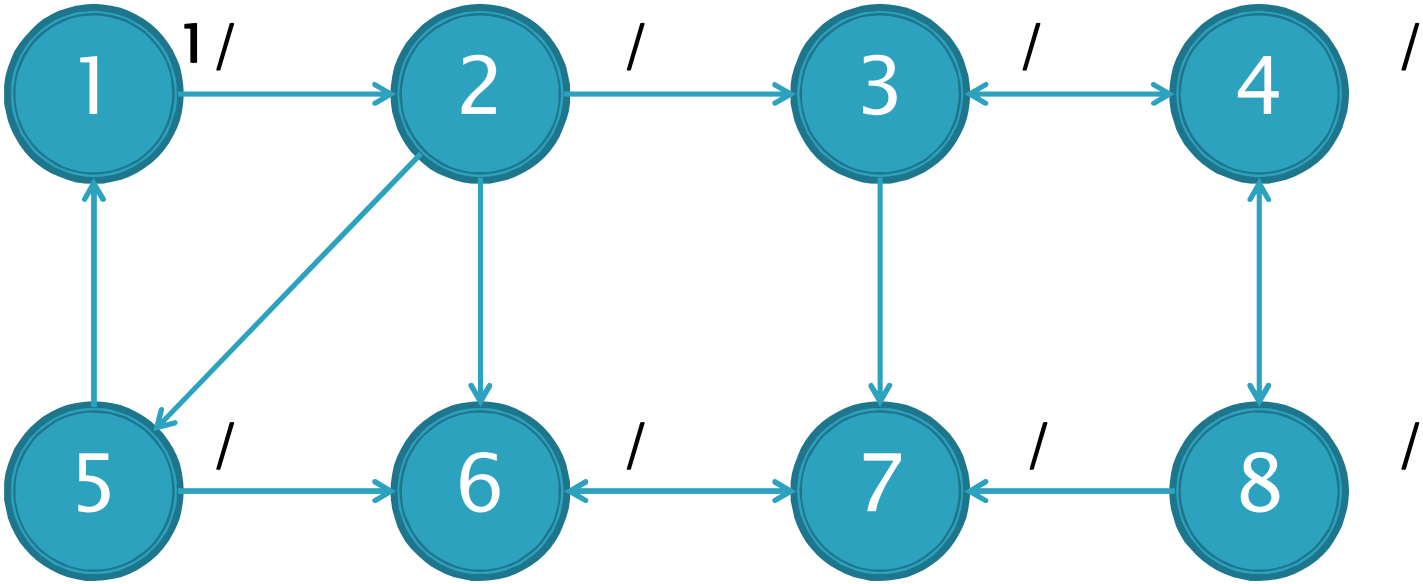
SCC



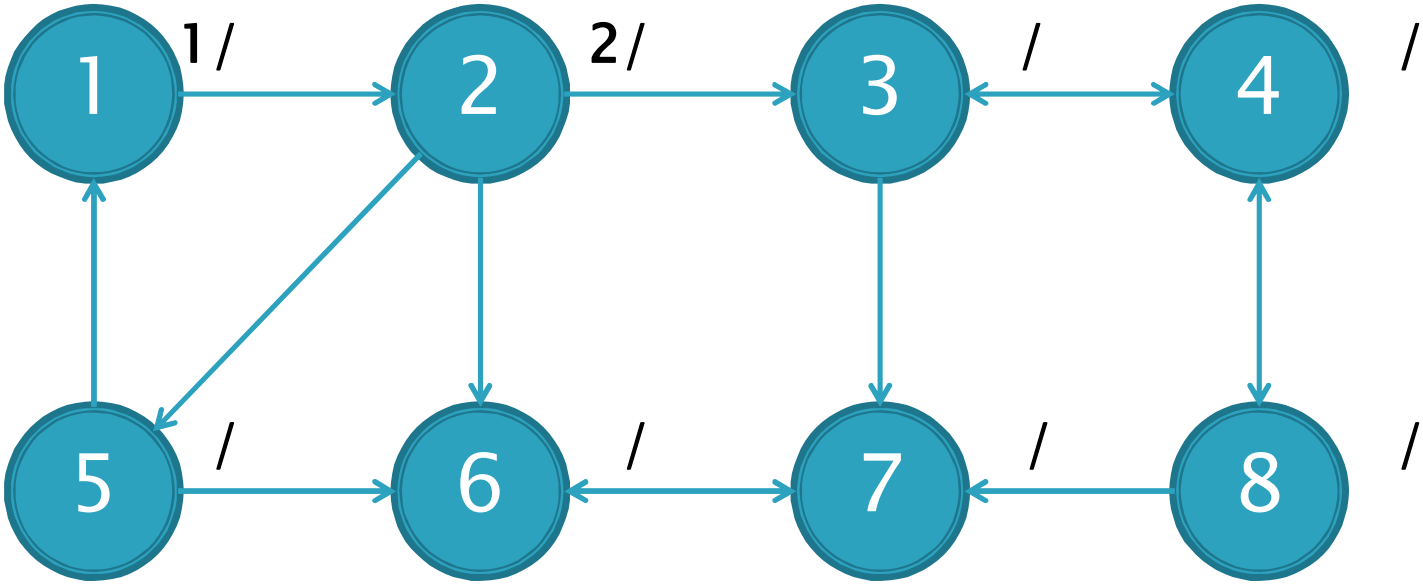
(1) 부터 DFS를 시작합시다. 들어가는 Time과

빠져나오는 Time을 모두 기록해볼까요?

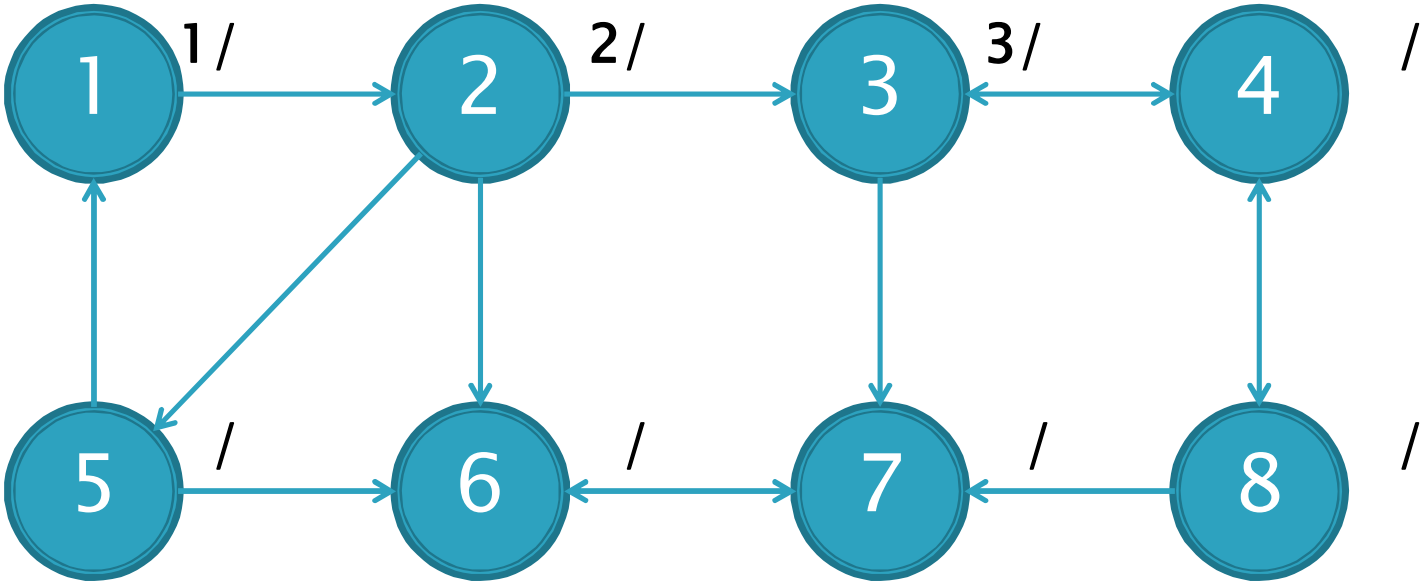
SCC



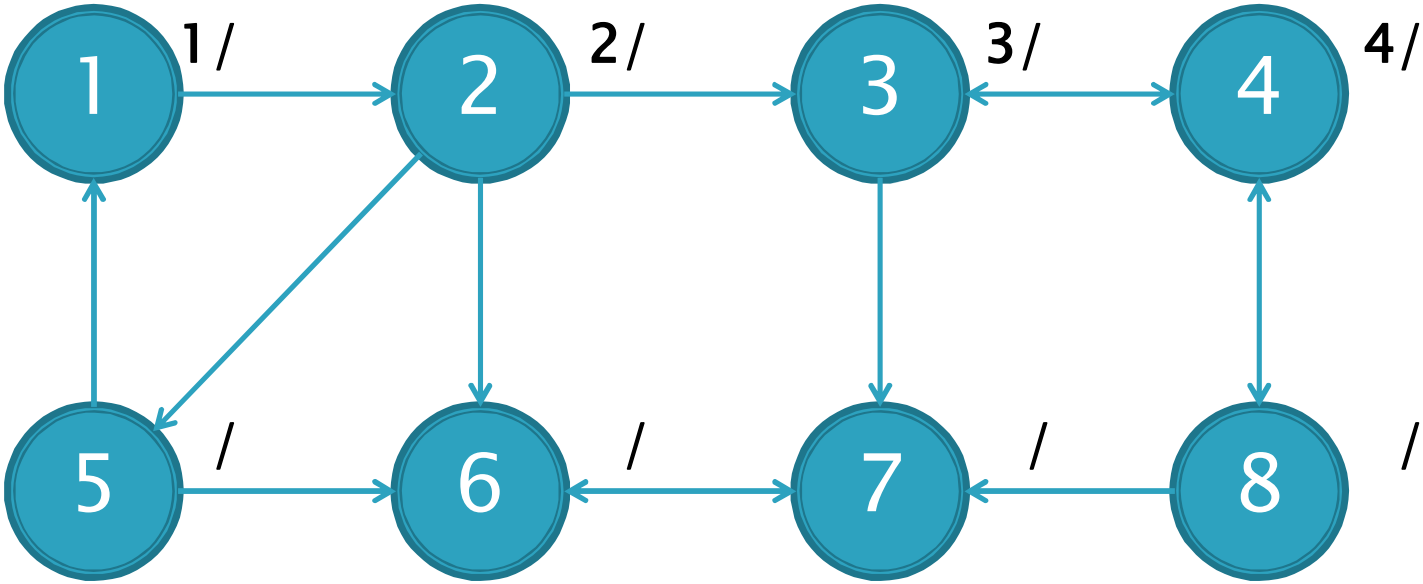
SCC



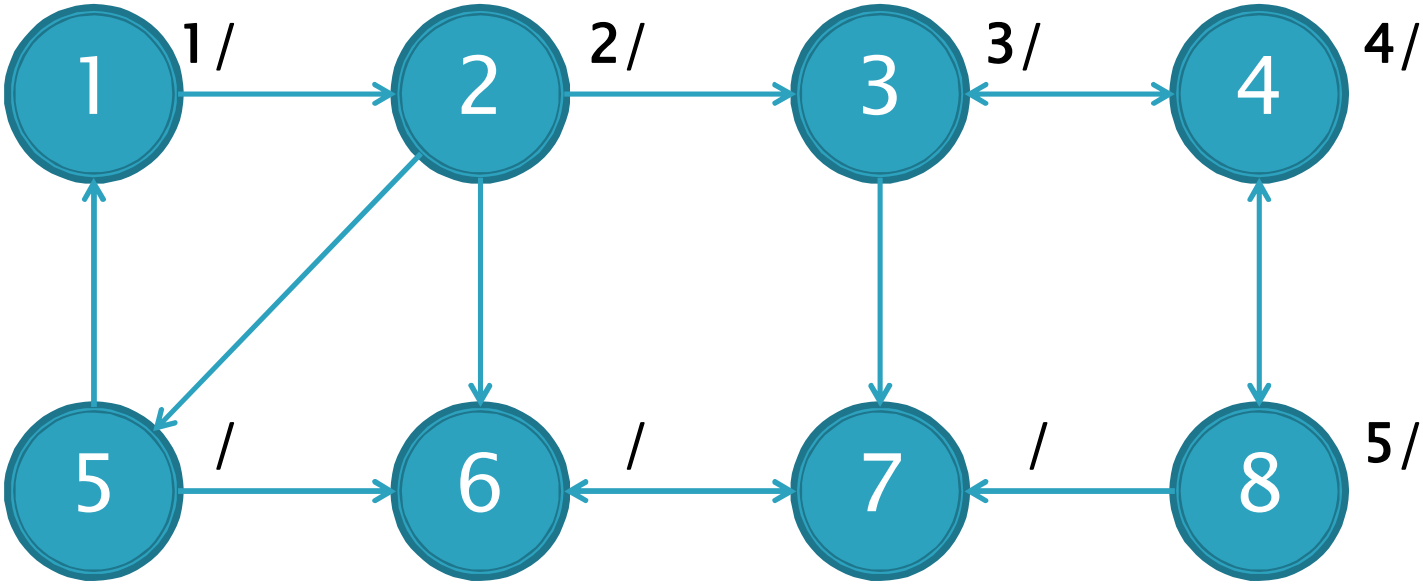
SCC



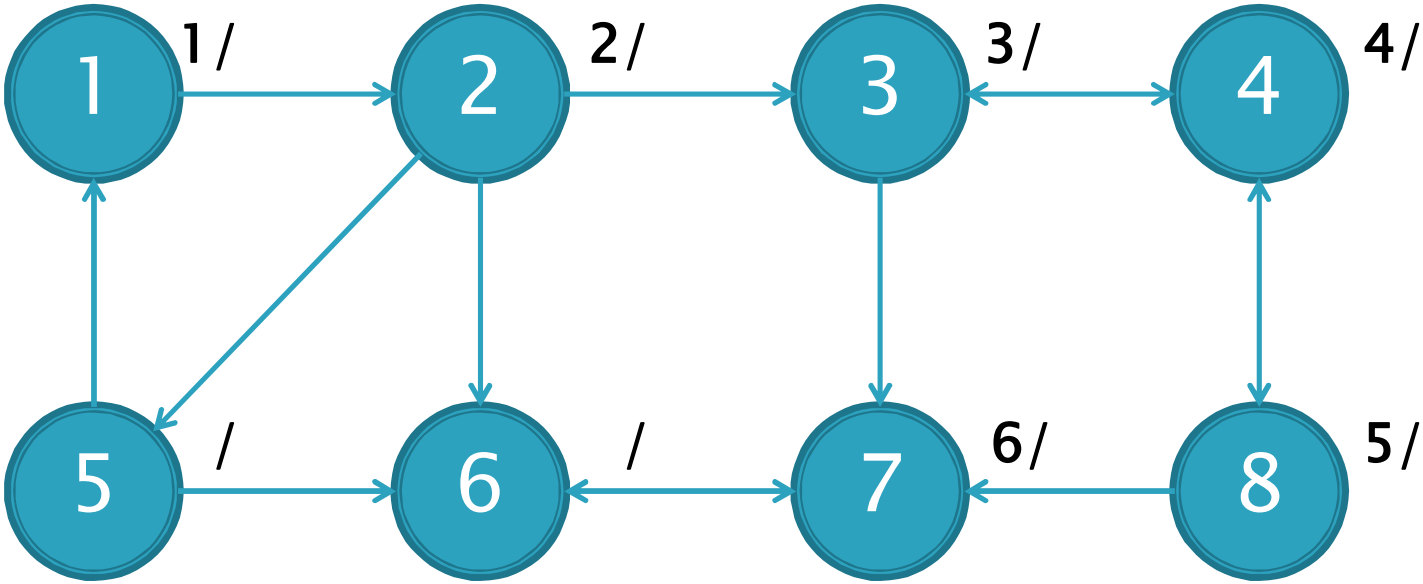
SCC



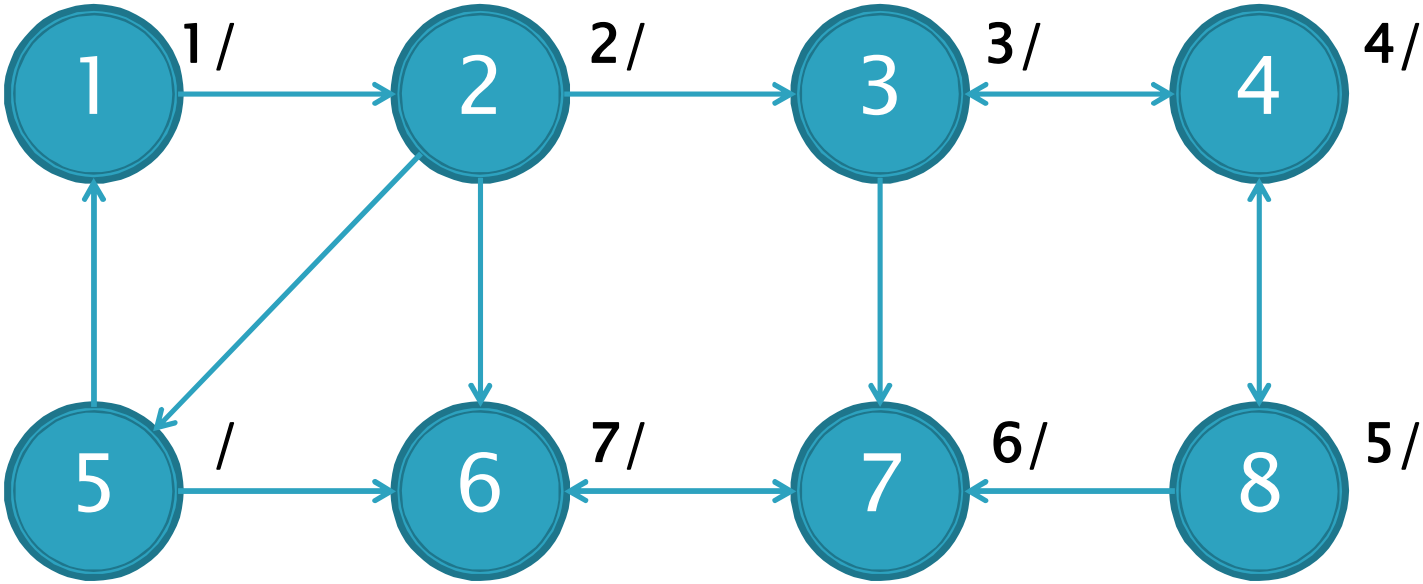
SCC



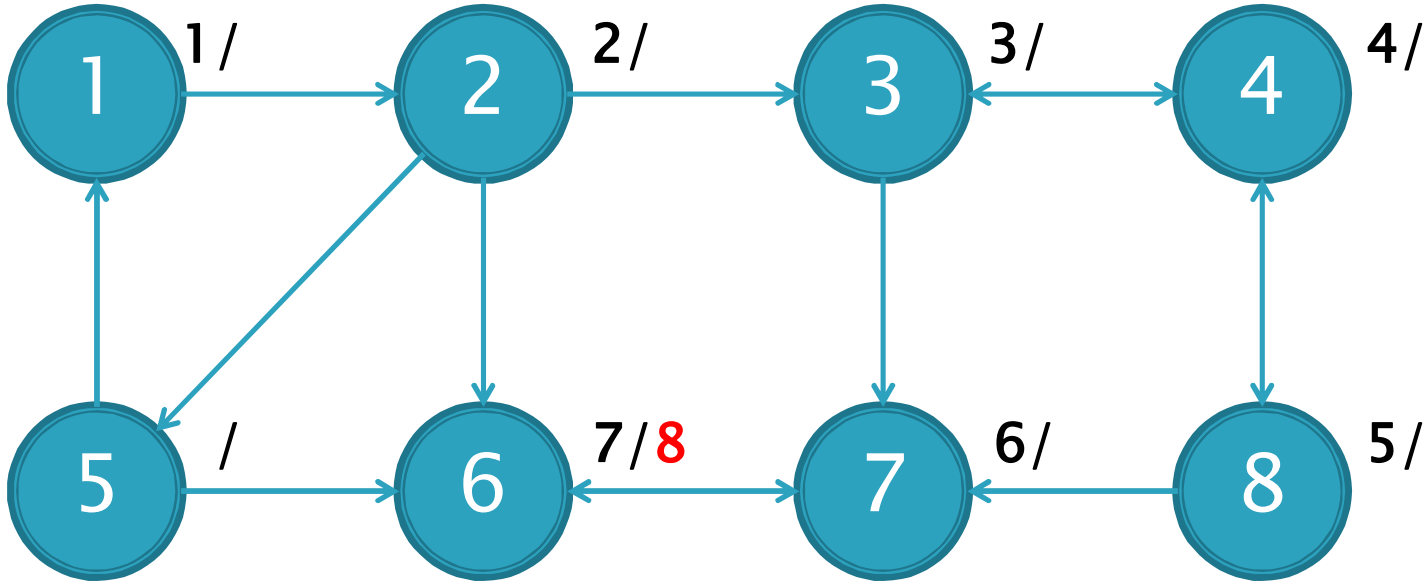
SCC



SCC



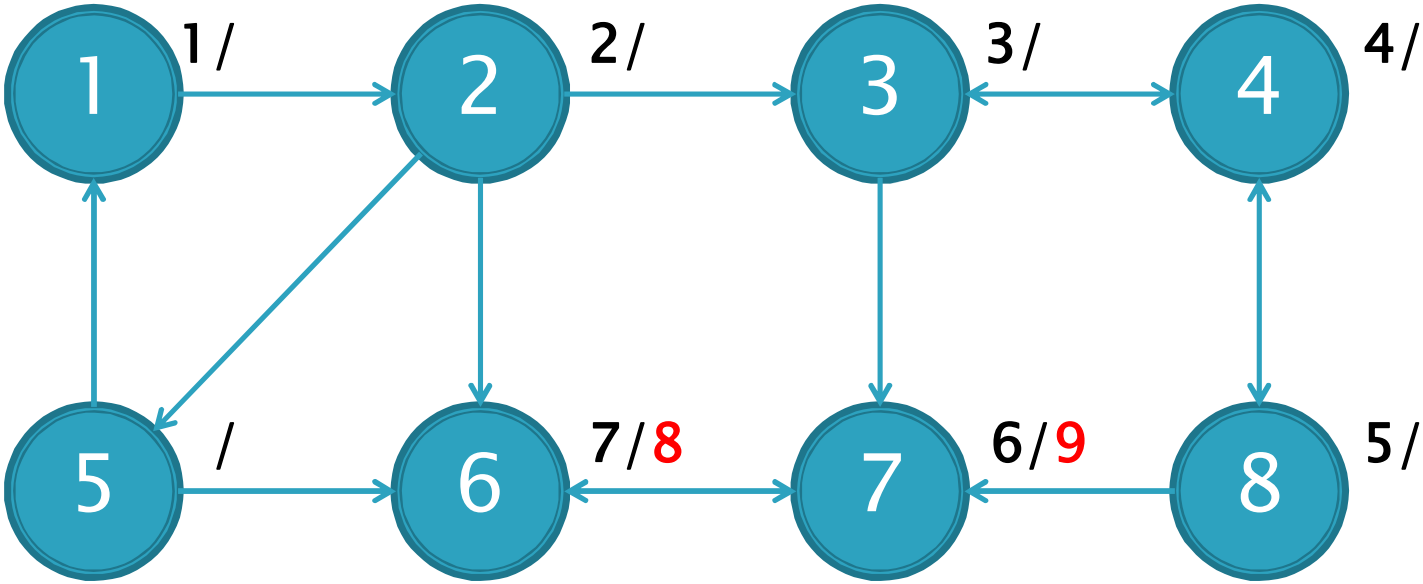
SCC



6번노드에서 더 이상 갈 곳이 없죠? 그러면 6번

노드에서 빠져나왔다고 생각하고 번호를 적습니다!

SCC

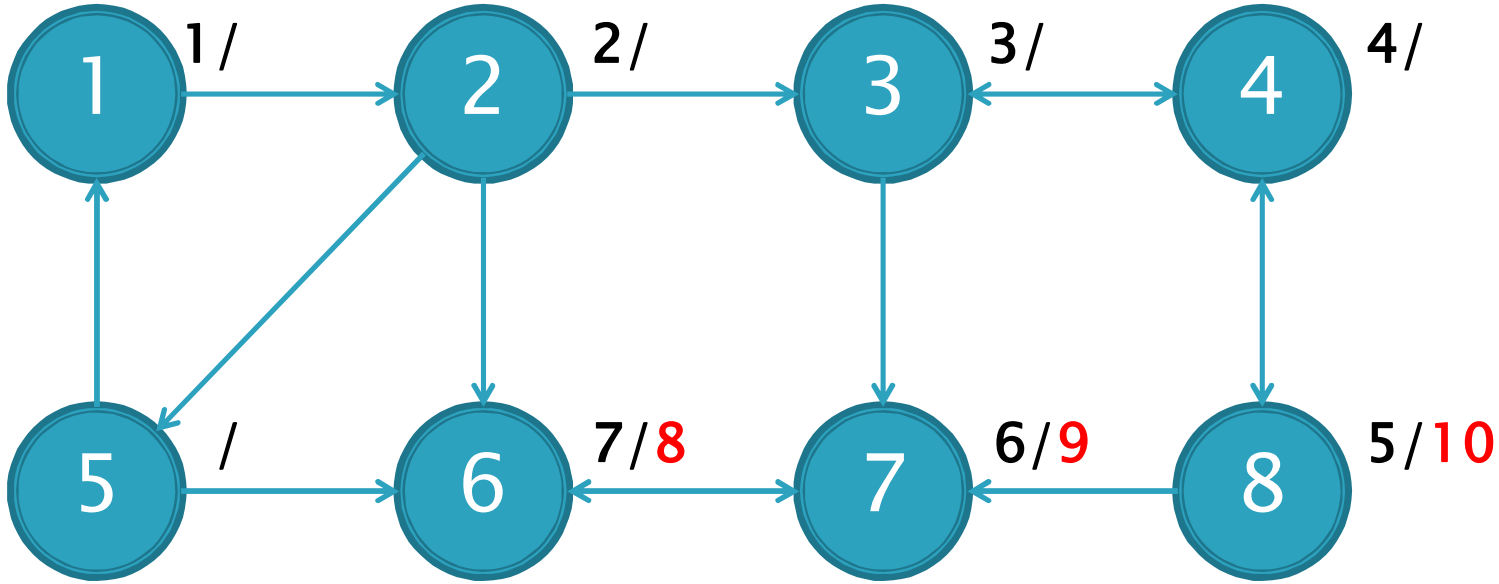


6번 노드에서의 DFS가 끝나면 7번이 6을 불렀으니

7번 노드로 가겠죠? 7번 노드 역시 갈 수 있는

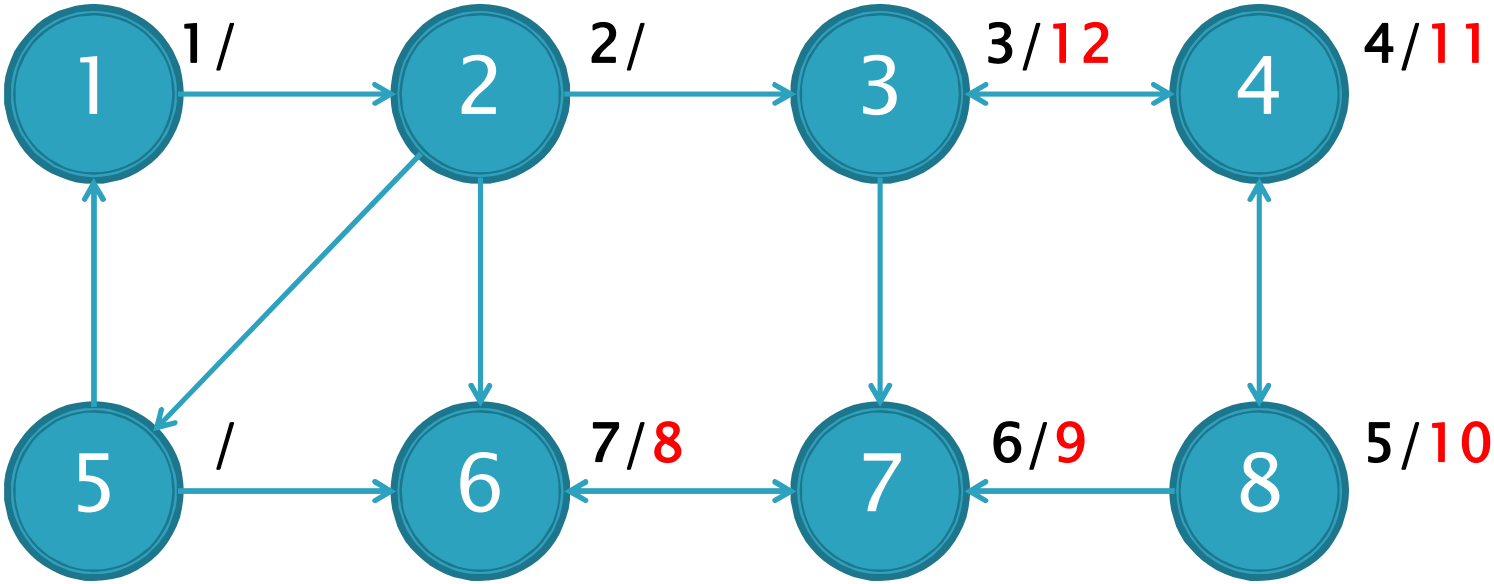
곳이 없습니다!

SCC



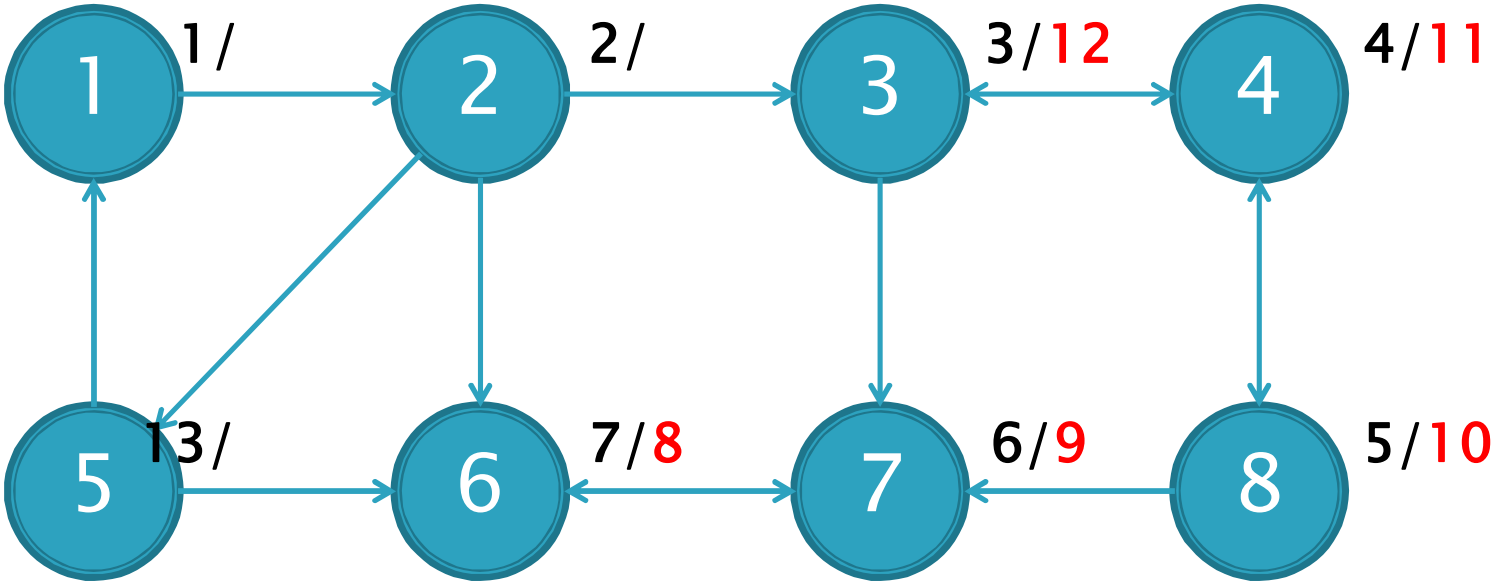
마찬가지로 8도 그렇고!

SCC



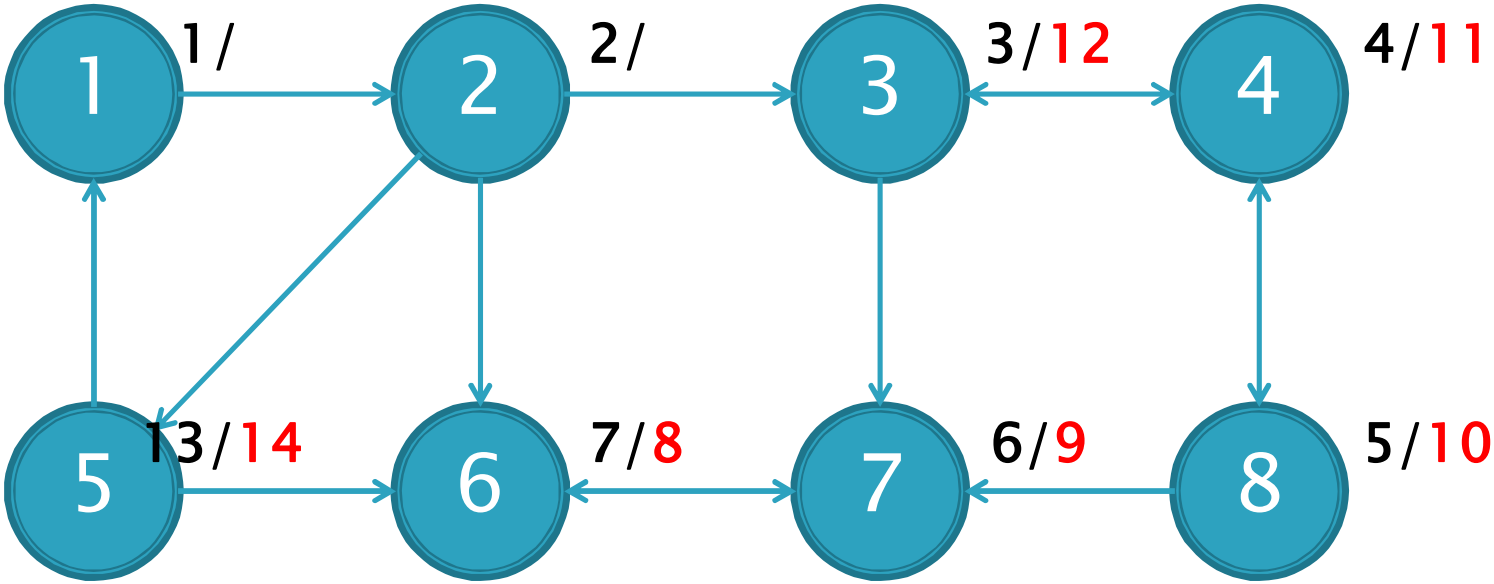
4, 3 모두 그렇습니다!

SCC



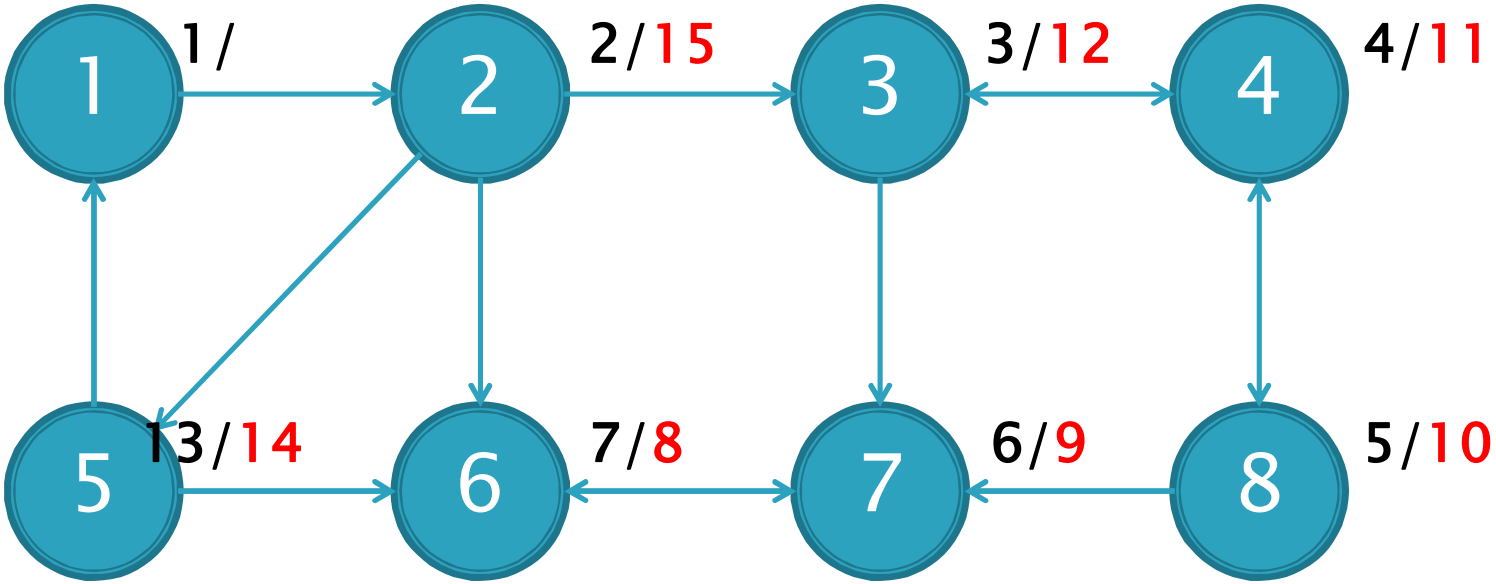
2의 경우는 아직 5로 갈 수 있죠?

SCC



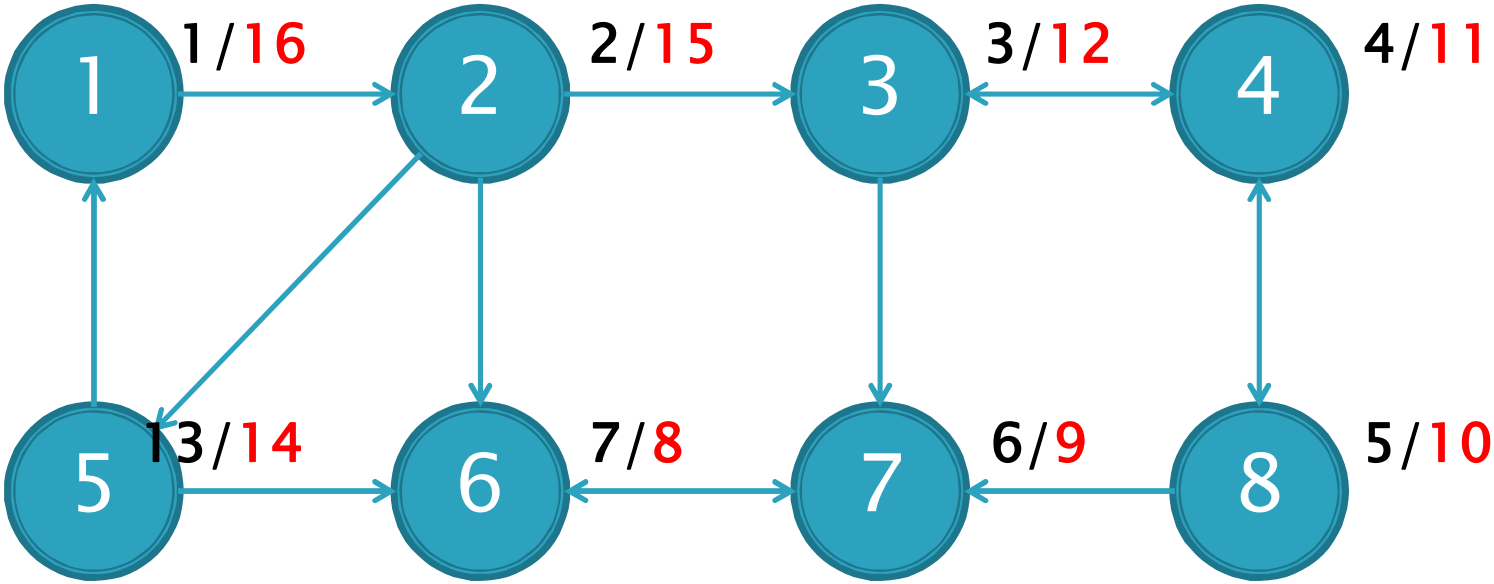
5에서 더 이상 갈 곳이 없네요!

SCC



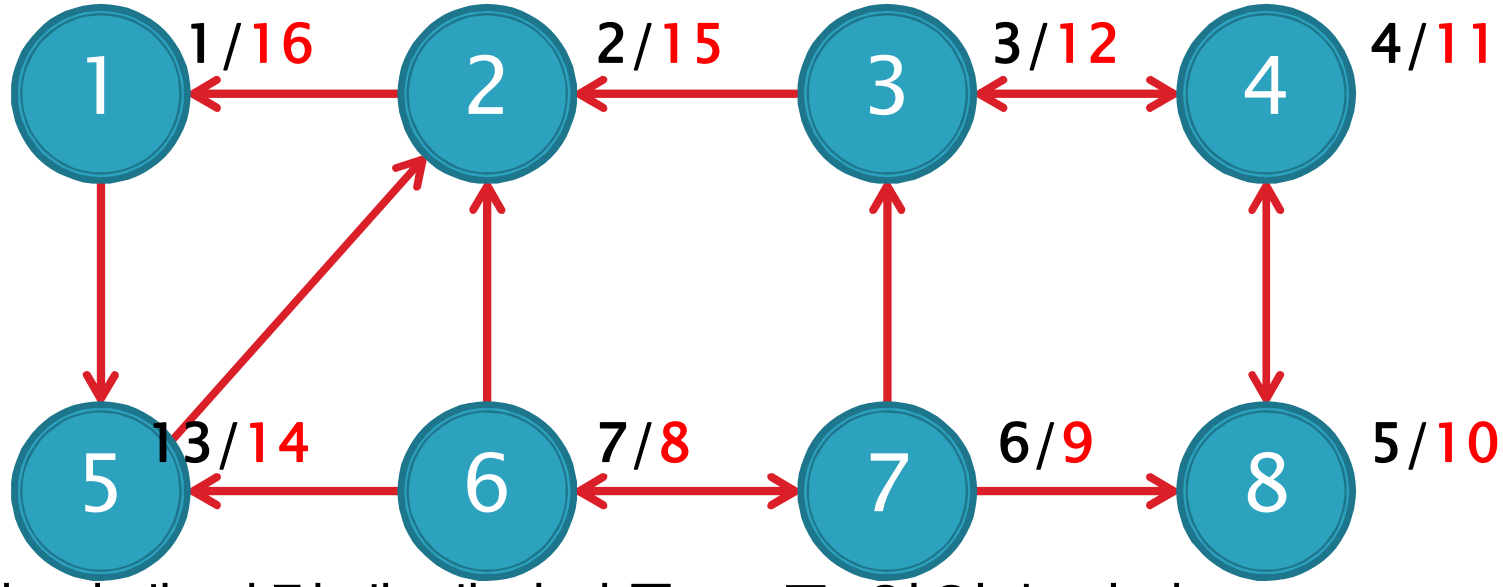
이젠 2에서도 갈 곳이 없습니다.

SCC



1에서도 갈 곳이 없네요. 이렇게 끝입니다!

SCC

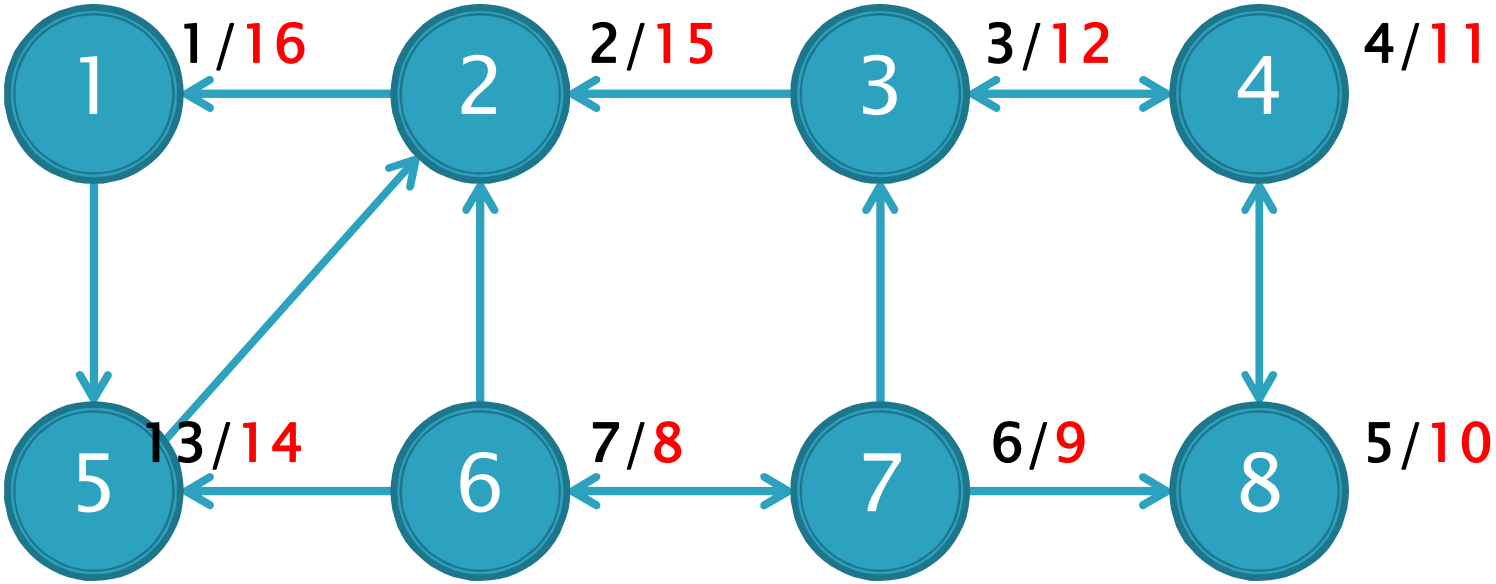


자 이제 이렇게 데이터를 모두 얻었습니다.

이제 간선의 방향을 모두 바꿔줍니다!

A -> B, B -> A 방향 모두 다 있을때는 바꿔도 같겠죠?

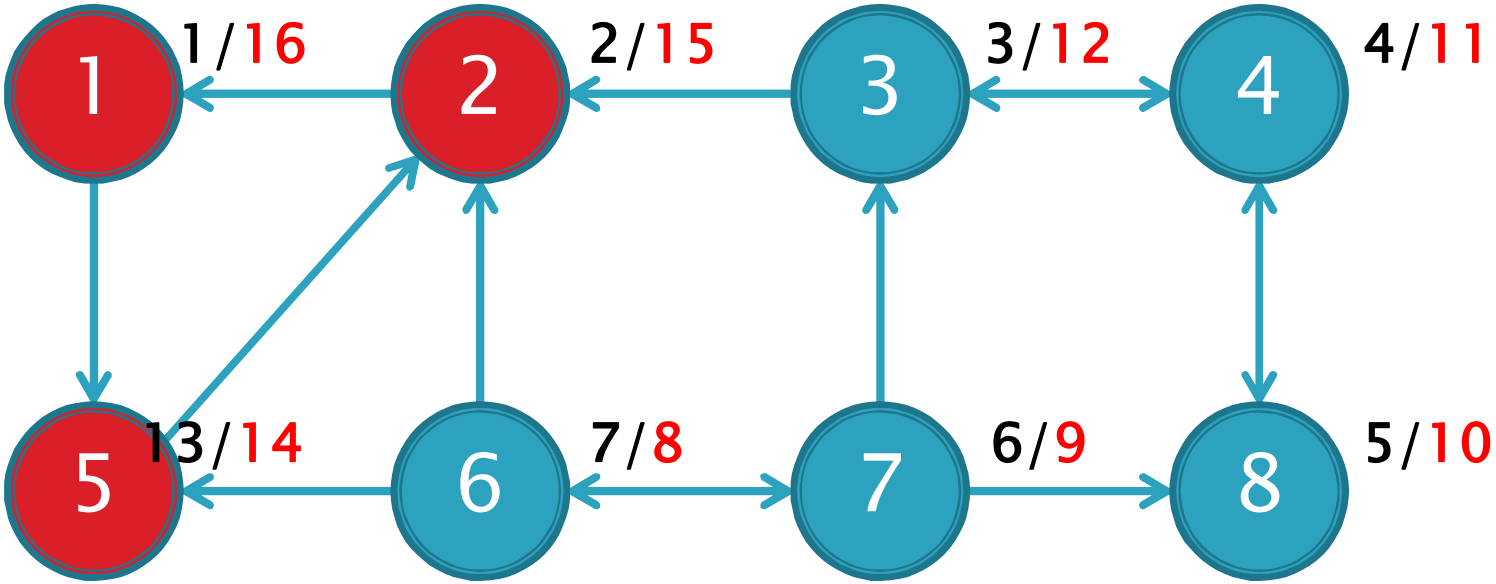
SCC



빠져나오는 숫자가 큰 노드부터 DFS를 돌립니다.

1. 에서 DFS를 돌려주면

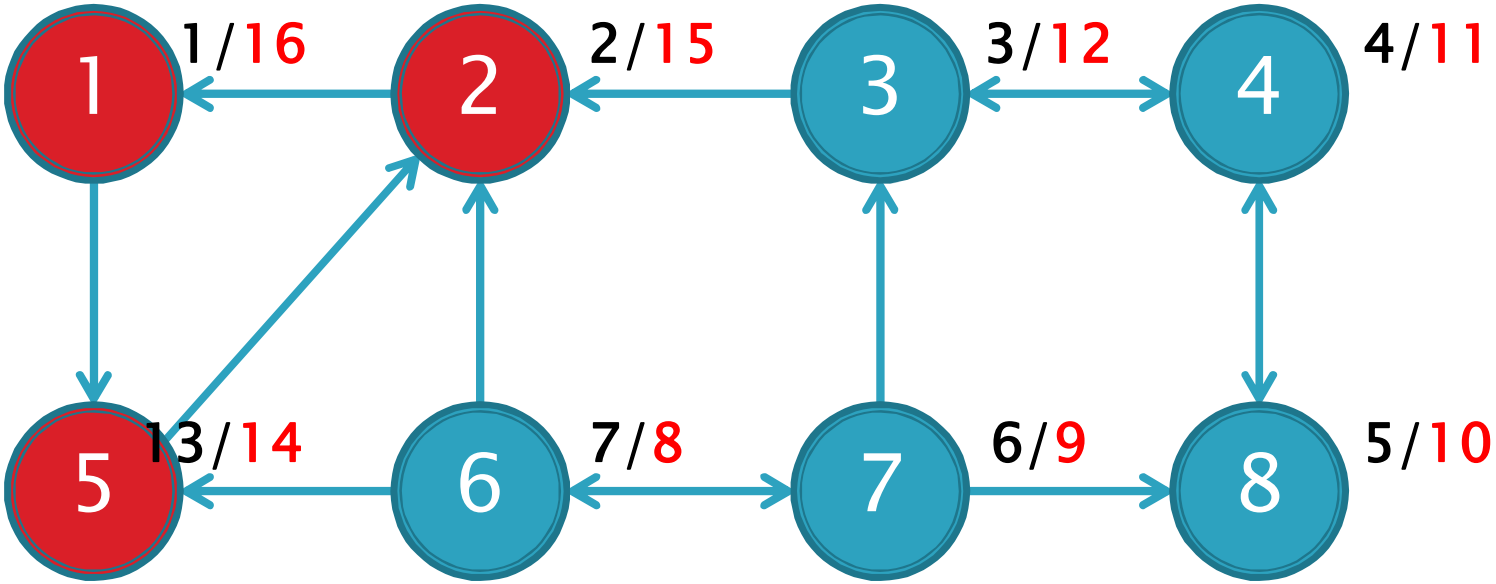
SCC



이렇게 노드를 방문할 수 있죠? 더 이상의 노드는

방문할 수 없습니다.

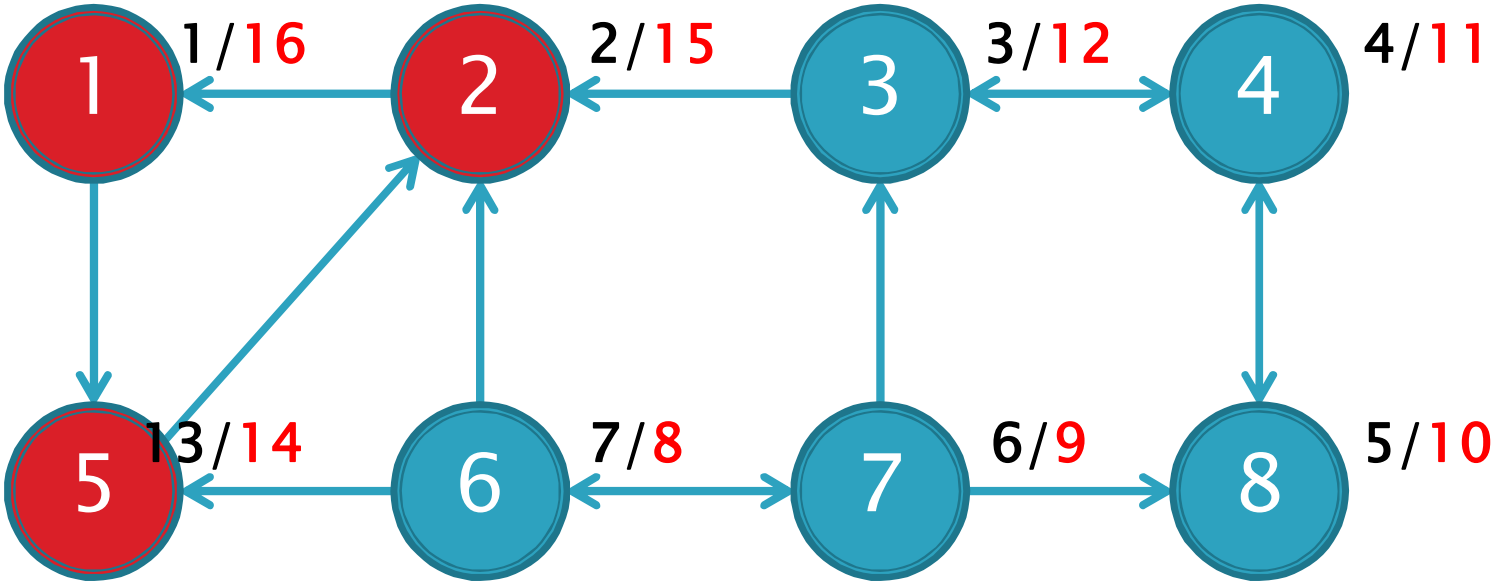
SCC



1에서 DFS가 끝났습니다. 그 다음으로 빠져나오는

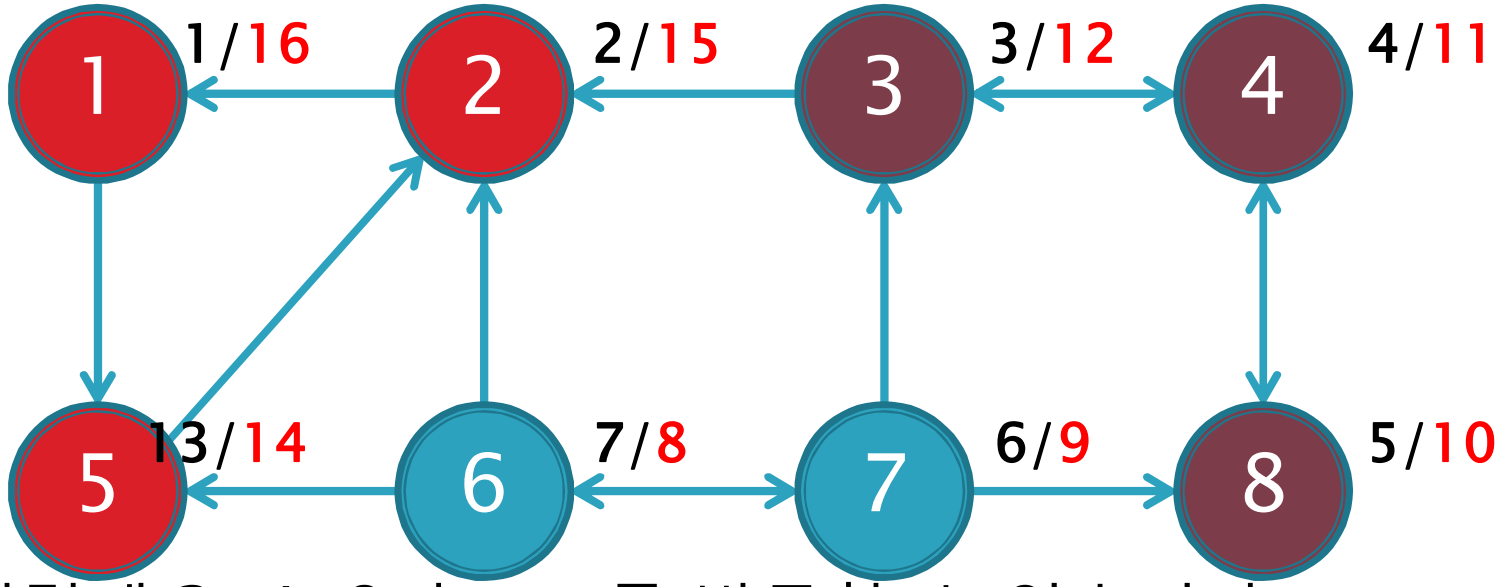
수가 큰 노드는 3번 노드죠?

SCC



그러니 3번노드에서 다시 DFS를 돌려주면!

SCC

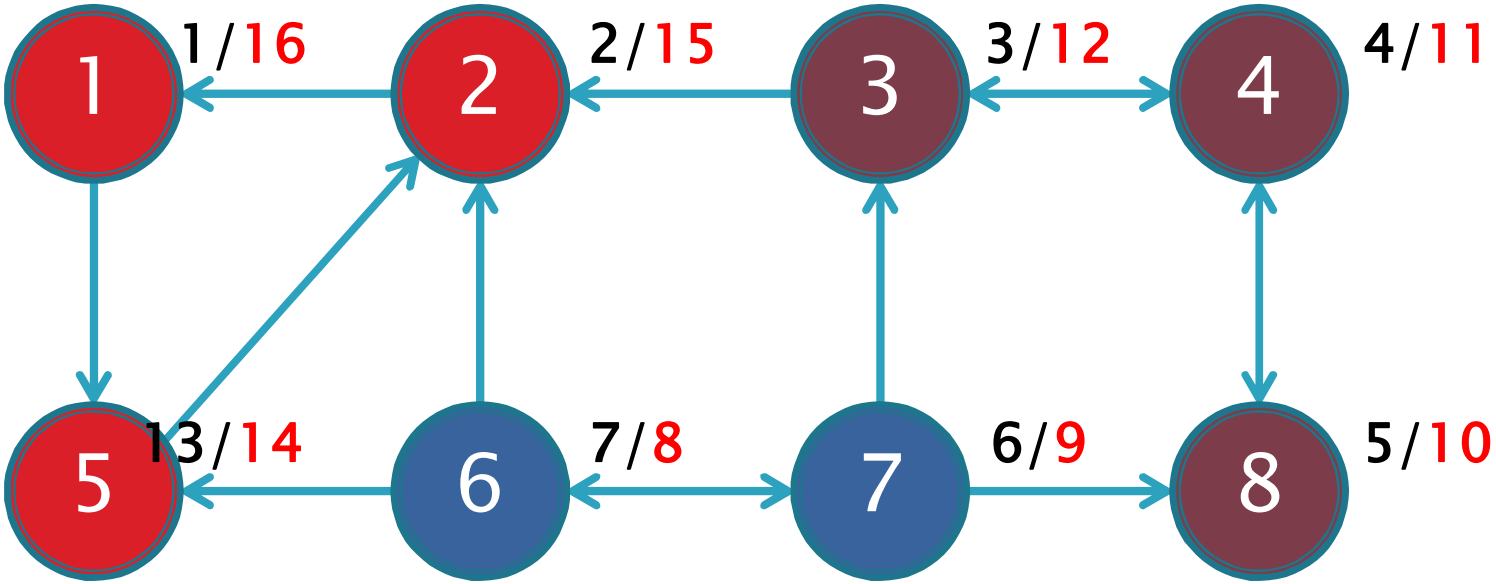


이렇게 3, 4, 8의 노드를 방문할 수 있습니다.

(이미 방문한 노드는 다시 방문하지 않습니다.)

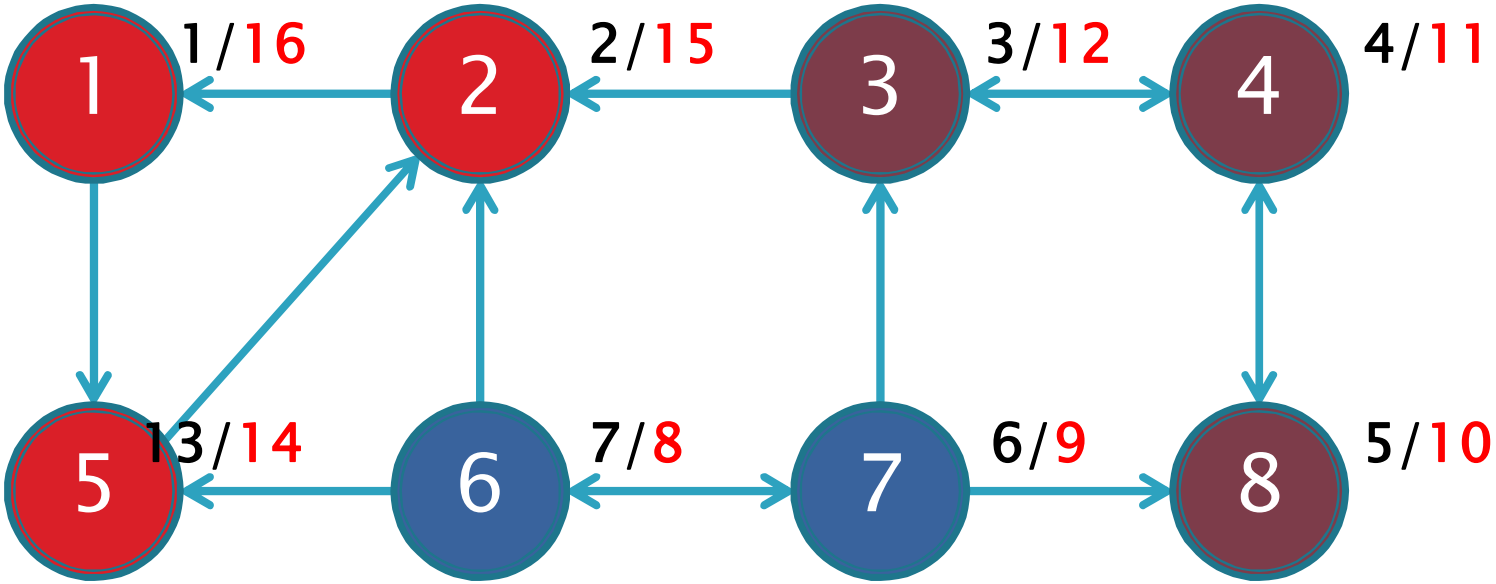
이제 DFS가 끝났죠? 그 다음으로 큰 노드는 7이니까

SCC



이렇게 7에서 6의 노드만 방문할 수 있습니다.

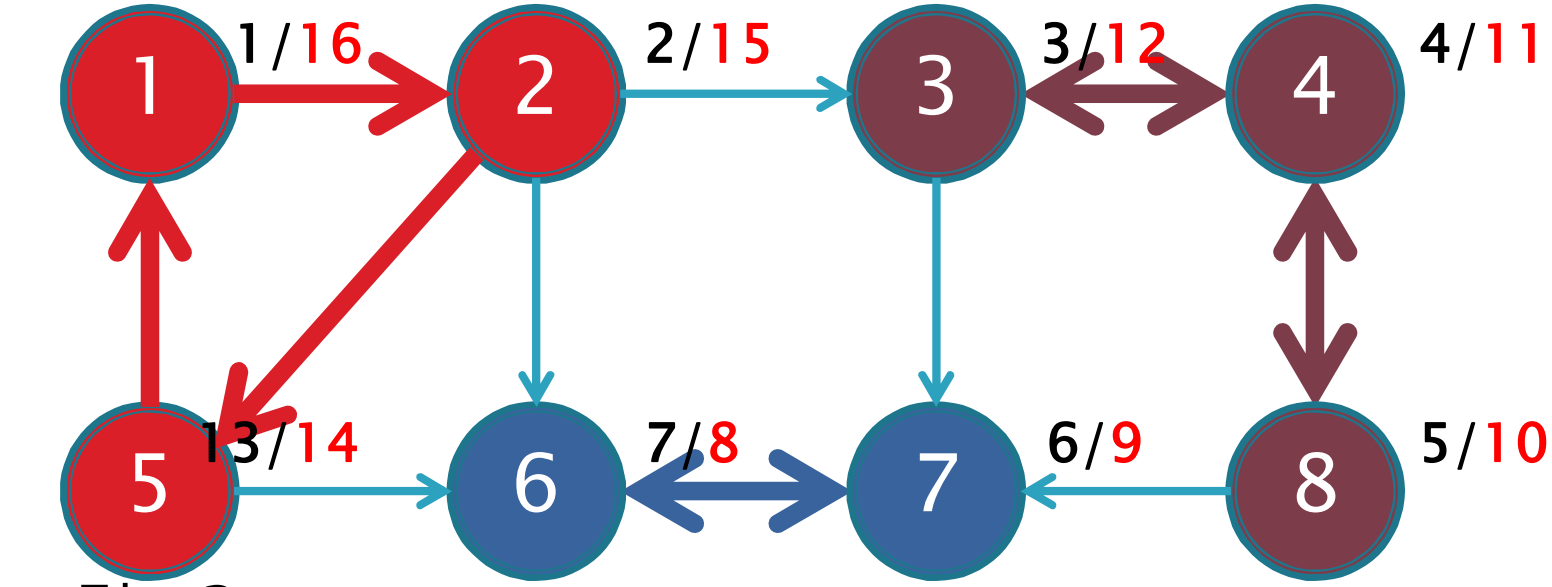
SCC



이와 같이 노드들을 묶어줄 수 있습니다.

같은 그룹에 묶여있는 노드들끼리 SCC를 이룹니다!

SCC

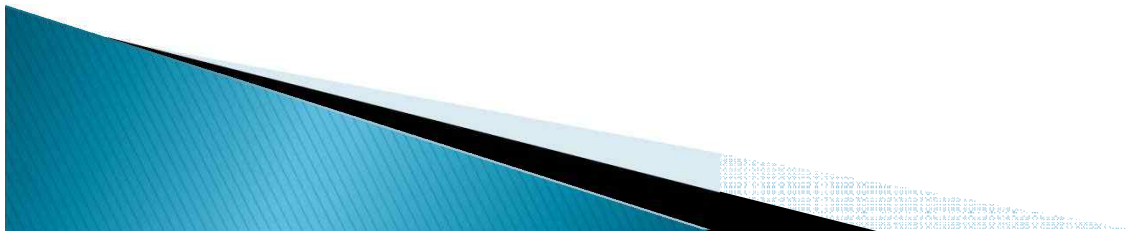


그렇죠?

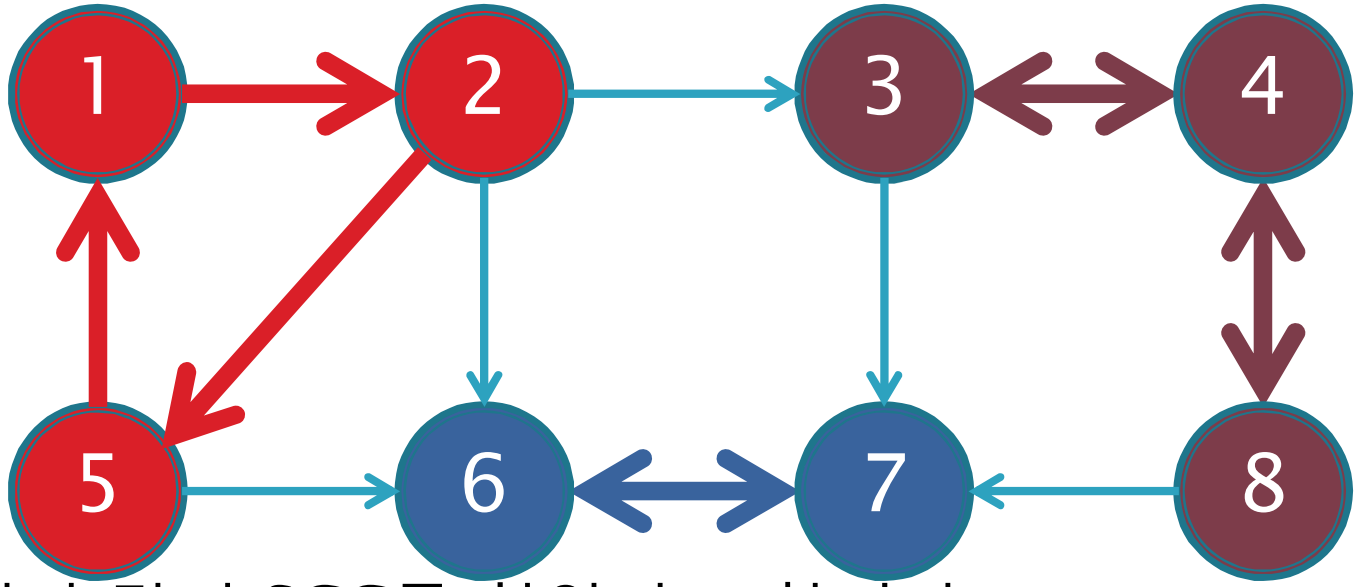
(헷갈리지 마세요. 원래 그래프는 위의 그래프입니다. 우리는 SCC를 구하기 위해 간선 방향을 바꿨죠?)

SCC

- SCC는 여태껏 KOI와 지역본선에 거의 나온 적이 없습니다. 이제 나올 때가 된 것 같은데…
- ☺ 잘 알아두도록 합니다!
- 참, 여기서 한가지만 더 알아볼게요.



SCC

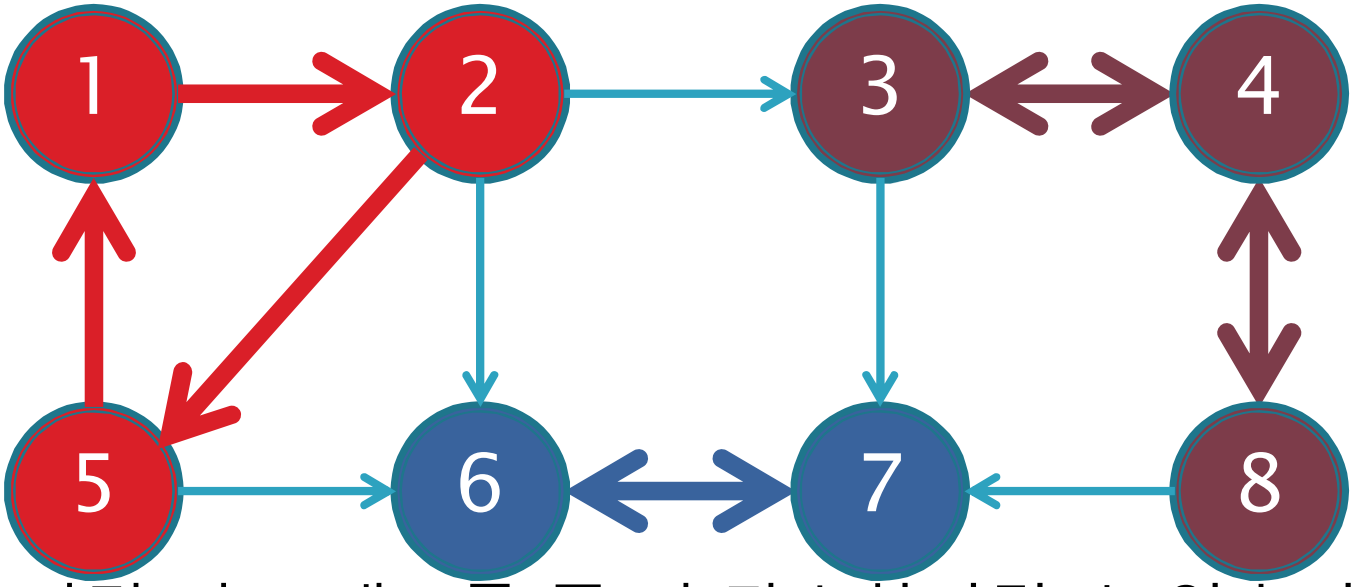


이와 같이 SCC를 찾았다고 합시다.

빨간색을 1번, 파란색을 2번, 보라색을 3번 묶음

이라고 할게요.

SCC

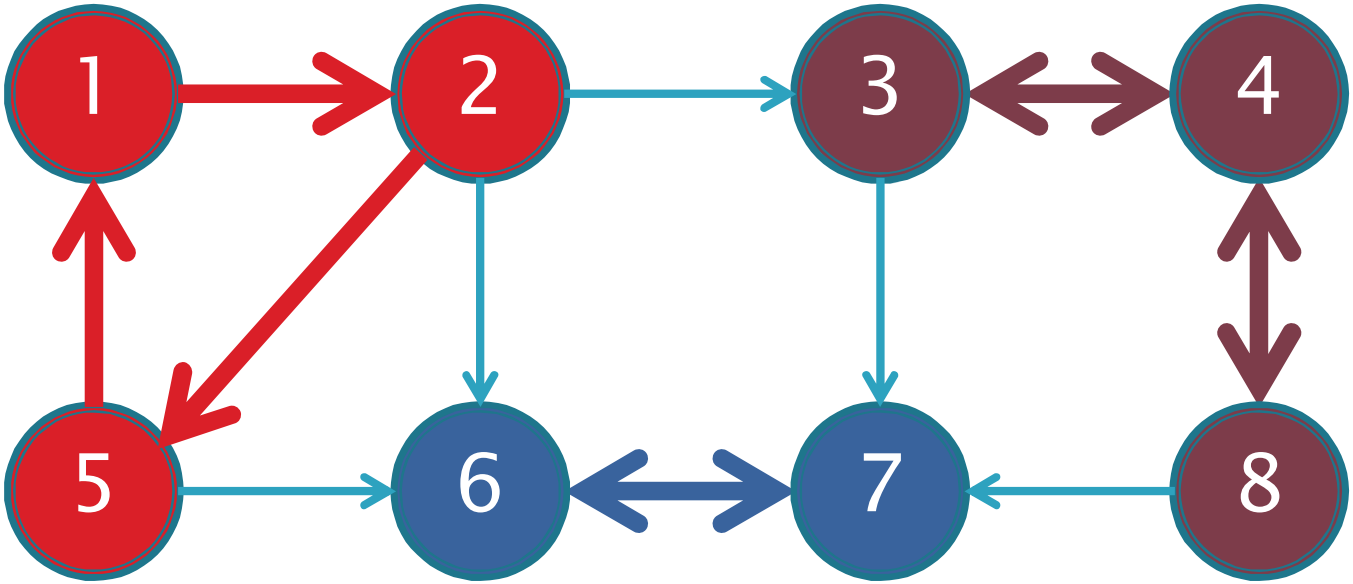


그러면 이 그래프를 좀 더 단순화시킬 수 있습니다.

1번 그룹에서는 **2번**과 **3번** 그룹으로 이동할 수

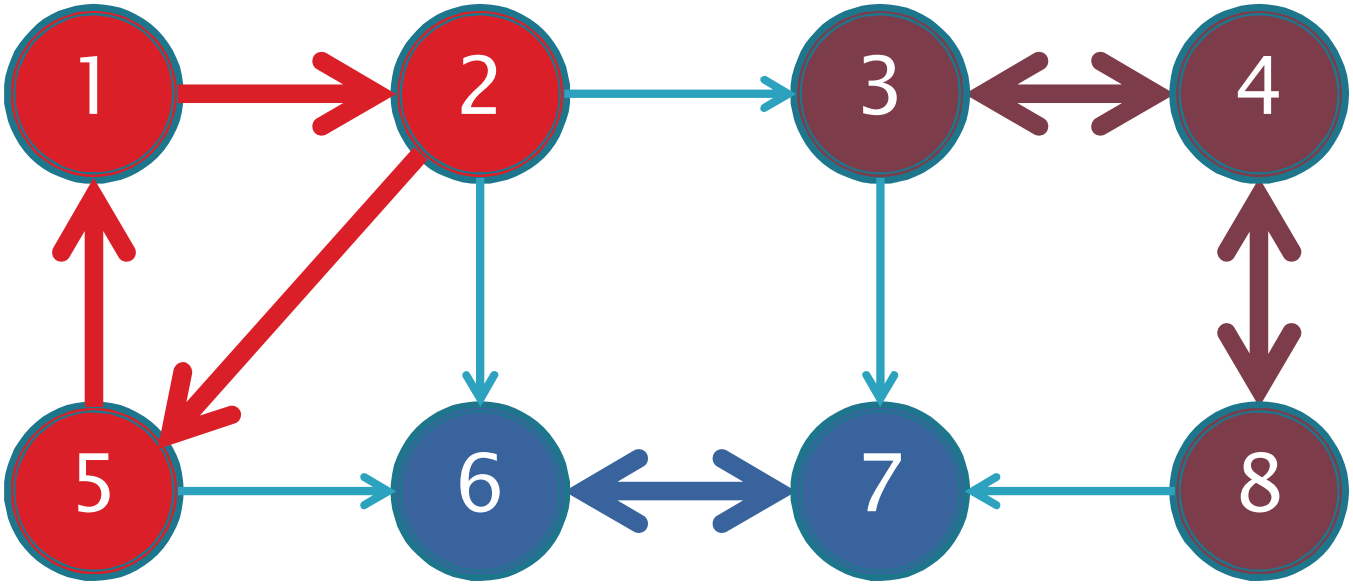
있죠? (2 -> 3, 2 -> 6, 5 -> 6의 간선이 있습니다.)

SCC



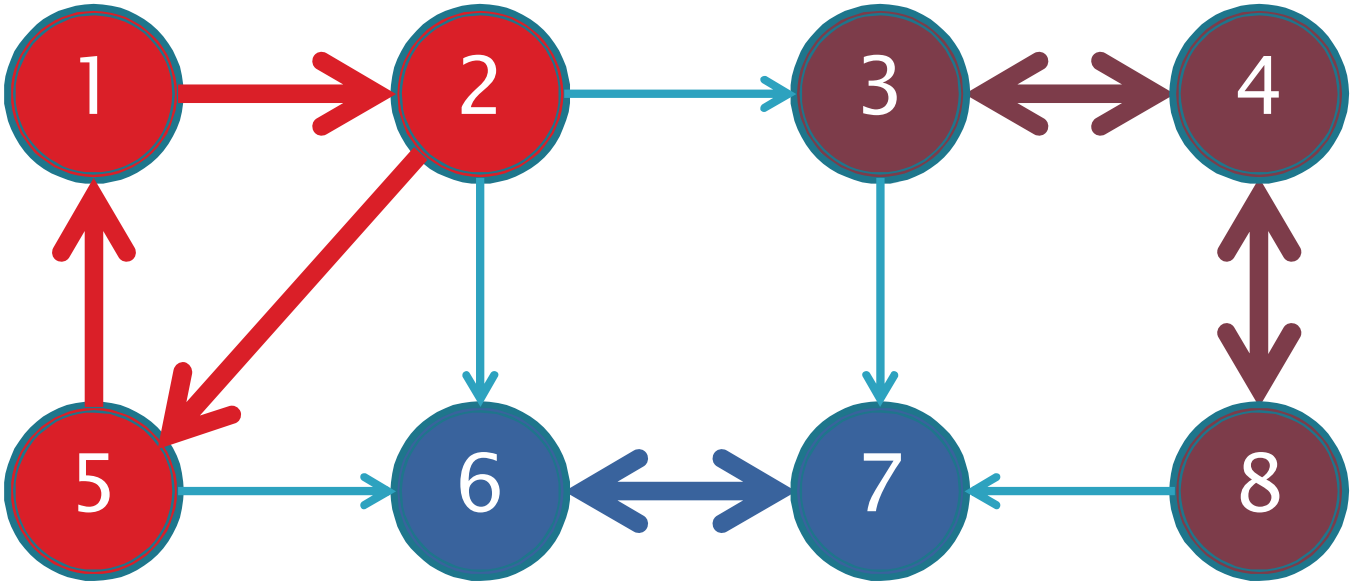
2번 그룹에서는 아무곳으로도 이동할 수 없습니다.

SCC



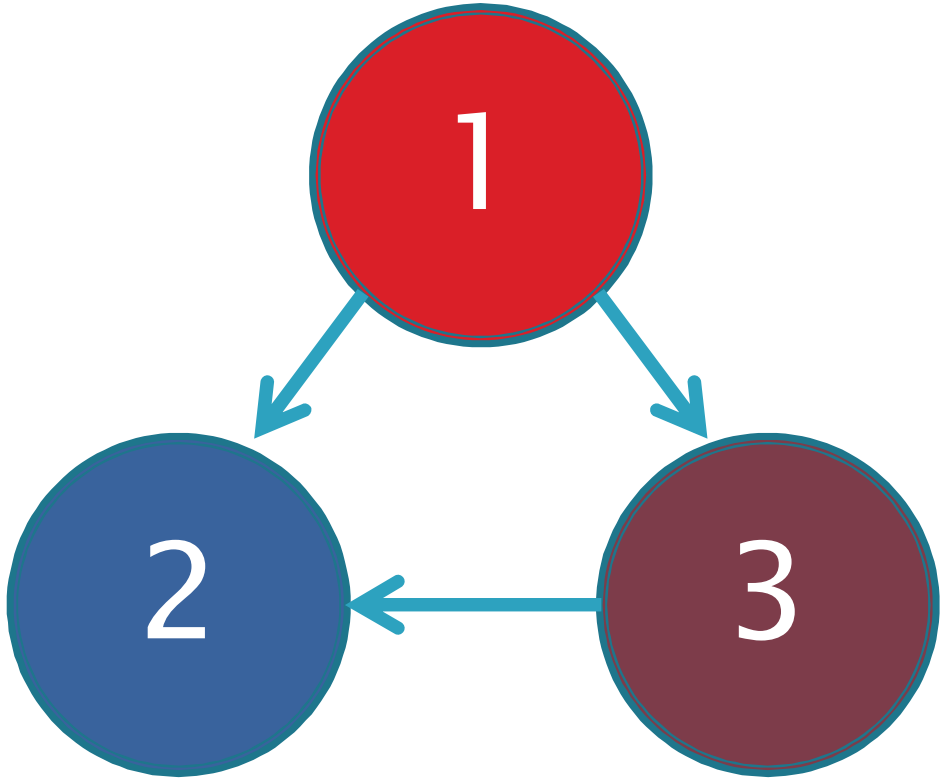
3번 그룹에서는 2번 그룹으로 이동할 수 있죠?

SCC



그러면 이것을 더 단순화시키면!

SCC



이렇게 나타낼 수 있습니다.

(어짜피 하나의 그룹 사이에서는 모두 이동할 수 있으니 의미없겠죠?