SCC / 2-SAT

나정휘

https://justiceHui.github.io/

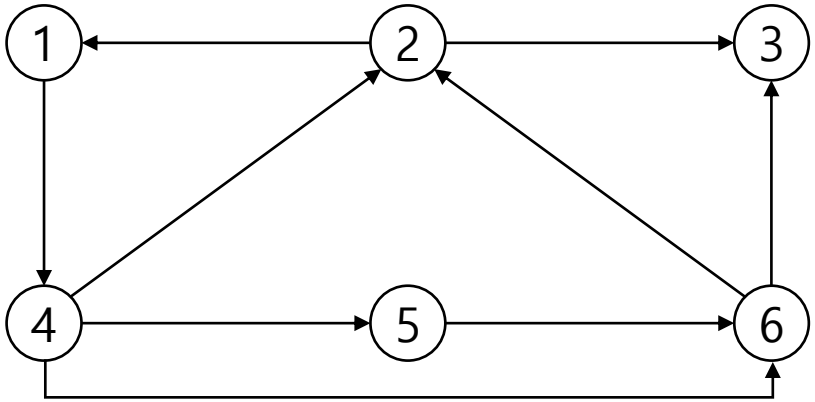
목차

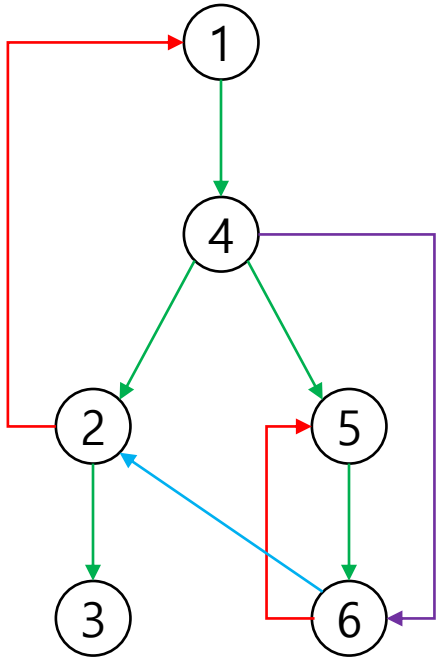
- DFS Tree
- SCC
- 2-SAT

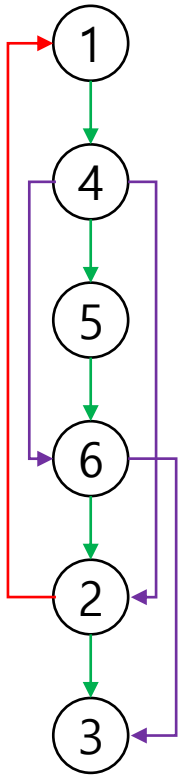
DFS Tree

DFS Tree

- DFS Tree
  - DFS를 수행하면서 방문한 간선들로 구성된 스패닝 트리
  - DFS Tree를 이용해 그래프의 간선을 4가지 종류로 분류할 수 있음
    - tree edge: DFS Tree에 포함된 간선
    - back edge: 조상으로 가는 간선
    - forward edge: 자식이 아닌 자손으로 가는 간선
    - cross edge: 그 외의 경우







DFS Tree

- DFS Tree
  - 간선의 분류를 이용해 해결할 수 있는 문제
    - 이분 그래프의 판별
      - 트리는 이분 그래프
      - tree edge로 연결된 정점을 서로 다른 색으로 칠한 다음 나머지 간선 확인
    - 무방향 그래프의 사이클 판별
      - back edge 있으면 사이클
    - 무방향 그래프의 단절점, 단절선
      - back edge를 이용해 구할 수 있음
      - 이번 캠프에서 다루지 않는 내용
    - 방향 그래프의 강한 연결 요소
      - 오늘 다룰 내용

질문?

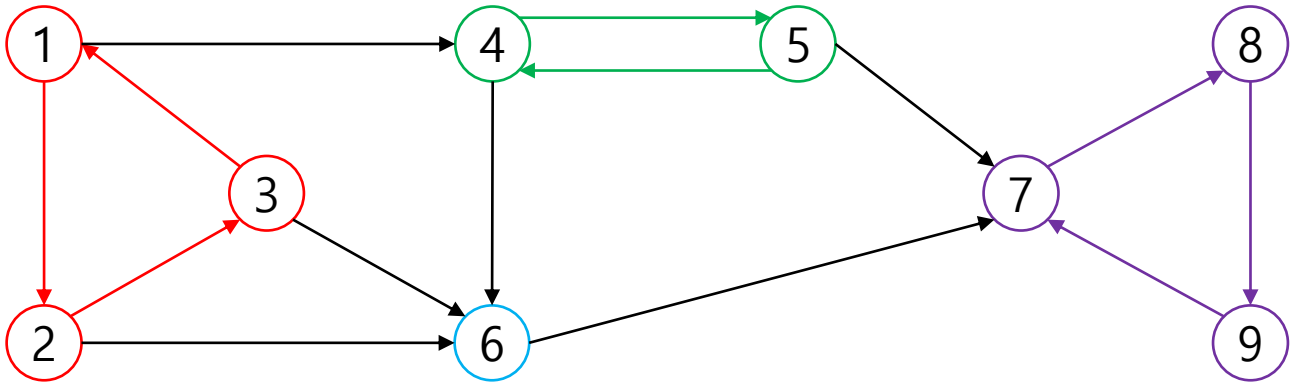
SCC

SCC

- 용어 정의
  - 무방향 그래프
    - connected component: 모든 정점이 서로 이동할 수 있음
    - 2-connected component: 정점을 하나 제거해도 서로 이동할 수 있음
    - 2-edge-connected component: 간선을 하나 제거해도 서로 이동할 수 있음
  - 방향 그래프
    - directed acyclic graph: 사이클이 없는 방향 그래프
    - weakly connected component: 간선의 방향을 무시했을 때 모든 정점이 서로 이동할 수 있음
    - strongly connected component: 모든 정점이 서로 이동할 수 있음

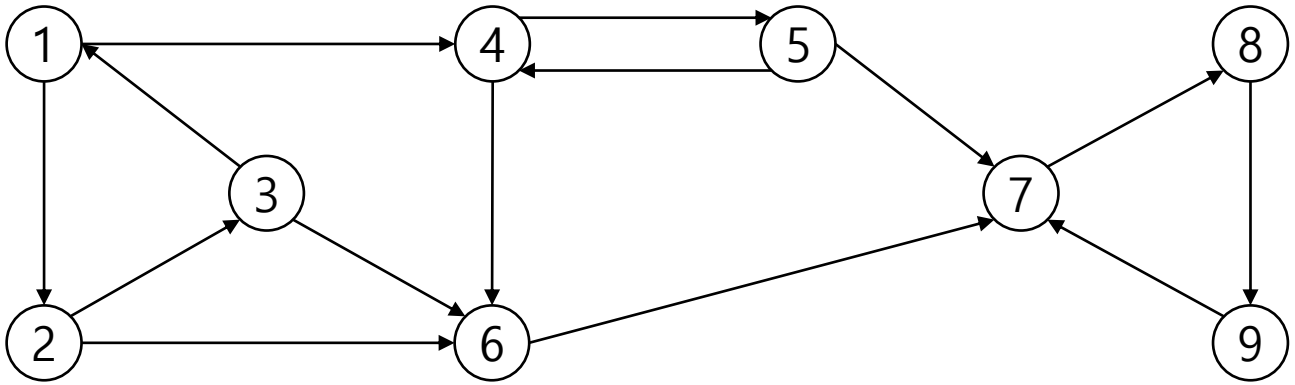
SCC

- SCC
  - SCC를 구하면 좋은 점
    - SCC를 정점 하나로 압축하면 임의의 방향 그래프를 DAG로 만들 수 있음 → DP
    - 서로 이동 가능하다는 것 자체가 중요한 경우도 있음
  - SCC를 구하는 방법
    - Tarjan's algorithm, Kosaraju's algorithm
  - ...



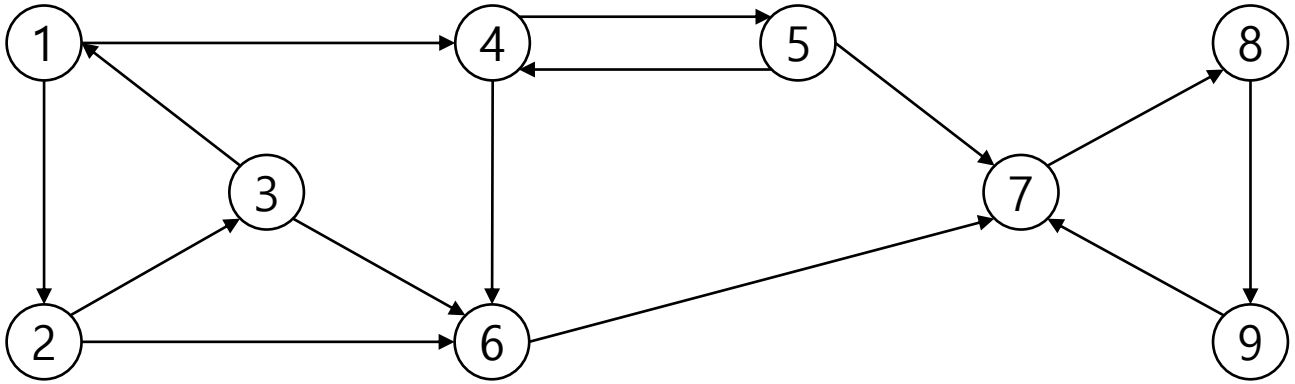
SCC

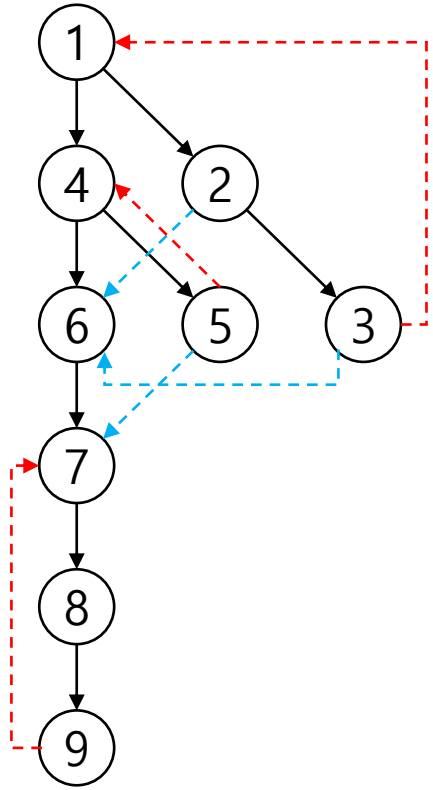
- Kosaraju's Algorithm
  - 알고리즘
    - DFS 하면서 빠져나온 순서대로 정점을 나열
    - 간선의 방향을 모두 뒤집음
    - 가장 늦게 빠져나온 정점부터 DFS
    - DFS를 할 때마다 SCC를 한 개씩 찾음



SCC

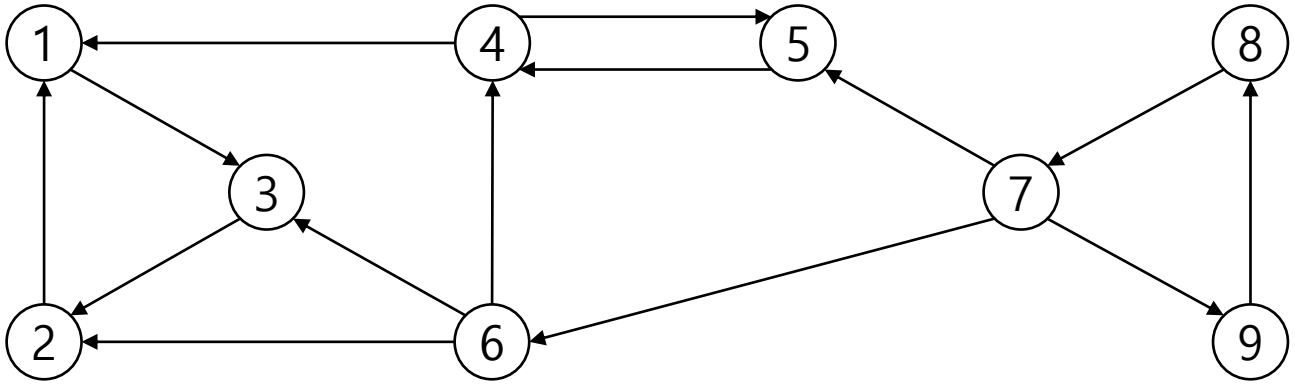
- Kosaraju's Algorithm
  - 알고리즘
    - DFS 하면서 빠져나온 순서대로 정점을 나열
      - 9 8 7 6 5 4 3 2 1
    - 간선의 방향을 모두 뒤집음
    - 가장 늦게 빠져나온 정점부터 DFS
    - DFS를 할 때마다 SCC를 한 개씩 찾음

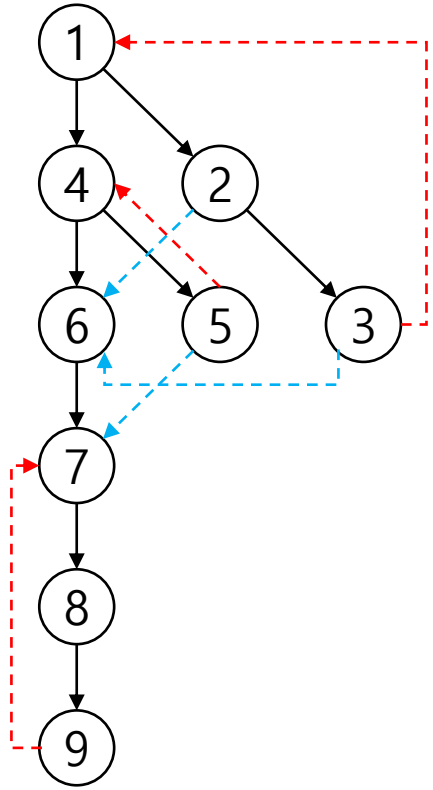




SCC

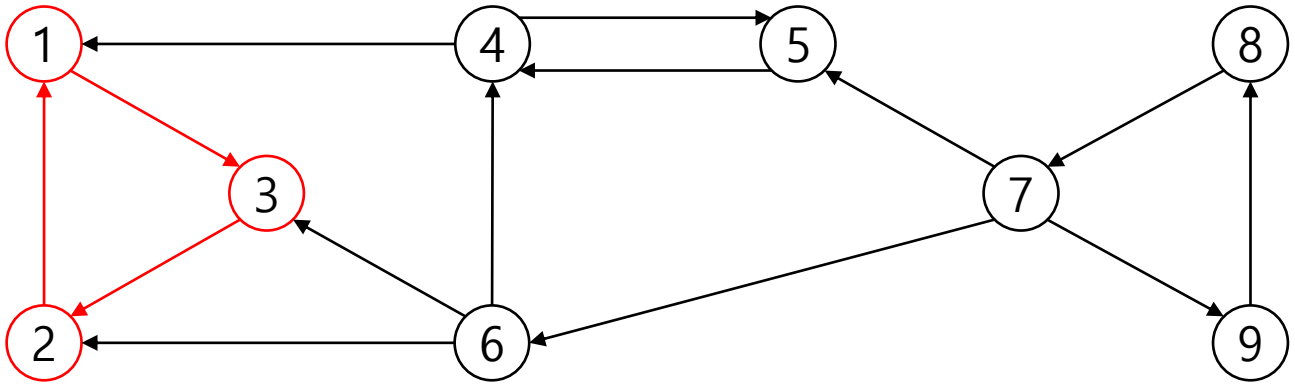
- Kosaraju's Algorithm
  - 알고리즘
    - DFS 하면서 빠져나온 순서대로 정점을 나열
      - 9 8 7 6 5 4 3 2 1
    - 간선의 방향을 모두 뒤집음
    - 가장 늦게 빠져나온 정점부터 DFS
    - DFS를 할 때마다 SCC를 한 개씩 찾음

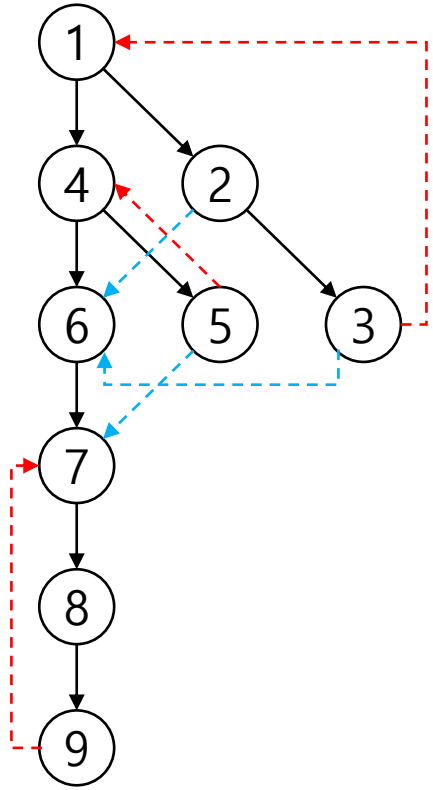




SCC

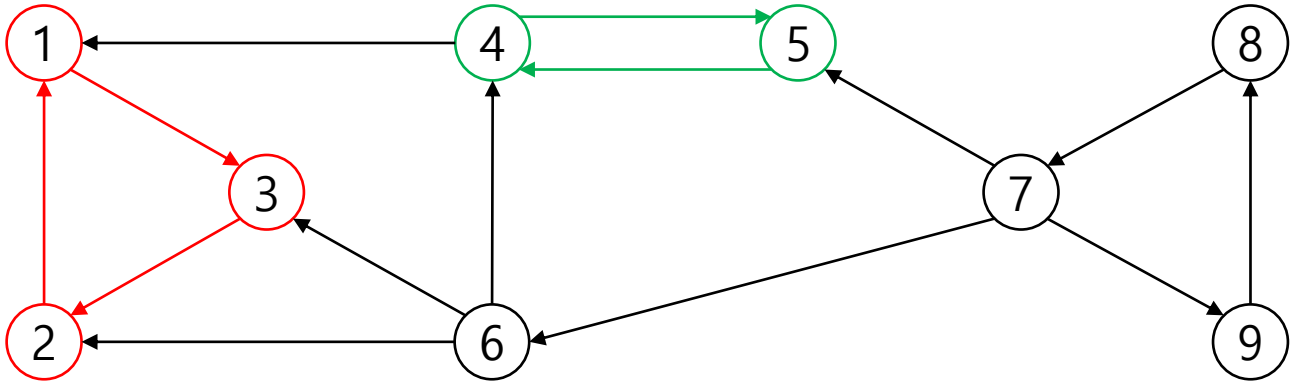
- Kosaraju's Algorithm
  - 알고리즘
    - DFS 하면서 빠져나온 순서대로 정점을 나열
      - 9 8 7 6 5 4 3 2 1
    - 간선의 방향을 모두 뒤집음
    - 가장 늦게 빠져나온 정점부터 DFS
    - DFS를 할 때마다 SCC를 한 개씩 찾음

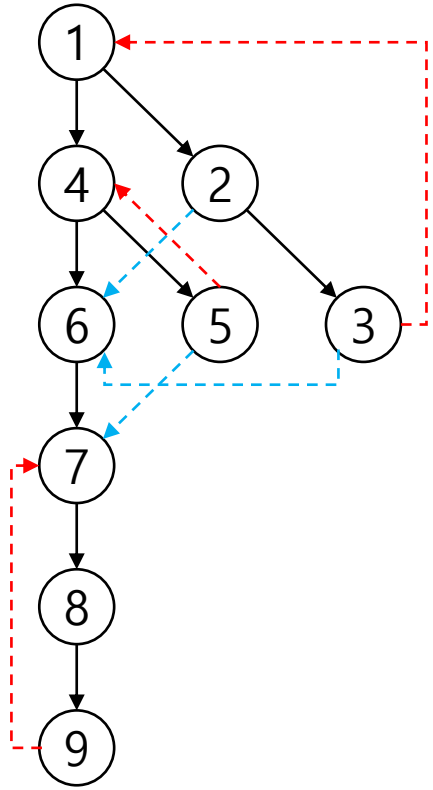




SCC

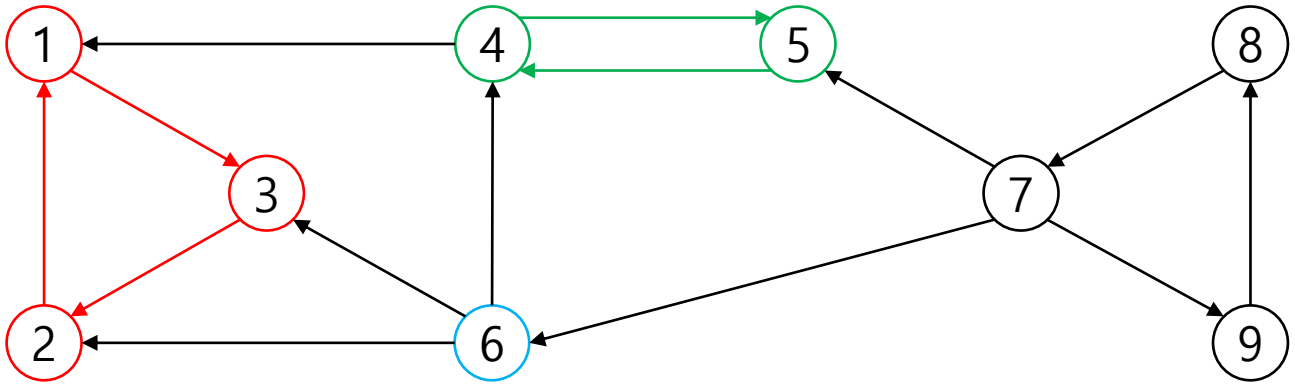
- Kosaraju's Algorithm
  - 알고리즘
    - DFS 하면서 빠져나온 순서대로 정점을 나열
      - 9 8 7 6 5 4 3 2 1
    - 간선의 방향을 모두 뒤집음
    - 가장 늦게 빠져나온 정점부터 DFS
    - DFS를 할 때마다 SCC를 한 개씩 찾음

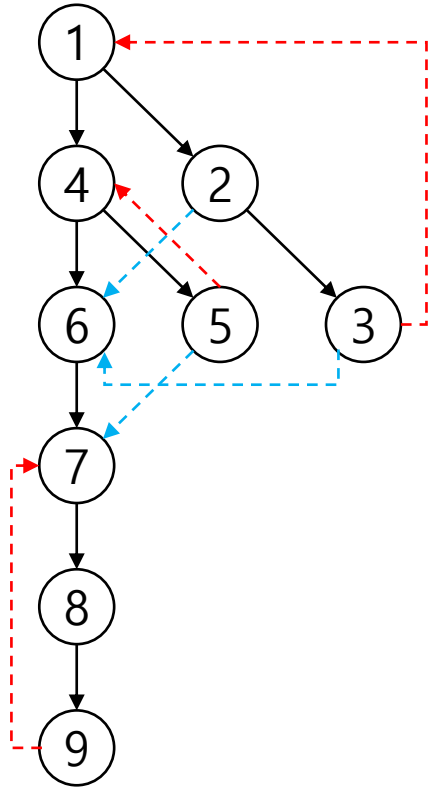




SCC

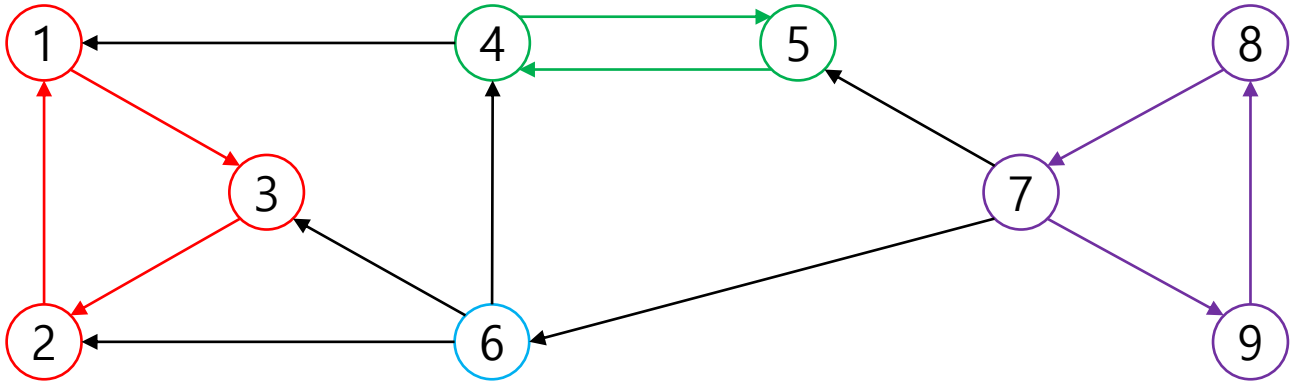
- Kosaraju's Algorithm
  - 알고리즘
    - DFS 하면서 빠져나온 순서대로 정점을 나열
      - 9 8 7 6 5 4 3 2 1
    - 간선의 방향을 모두 뒤집음
    - 가장 늦게 빠져나온 정점부터 DFS
    - DFS를 할 때마다 SCC를 한 개씩 찾음

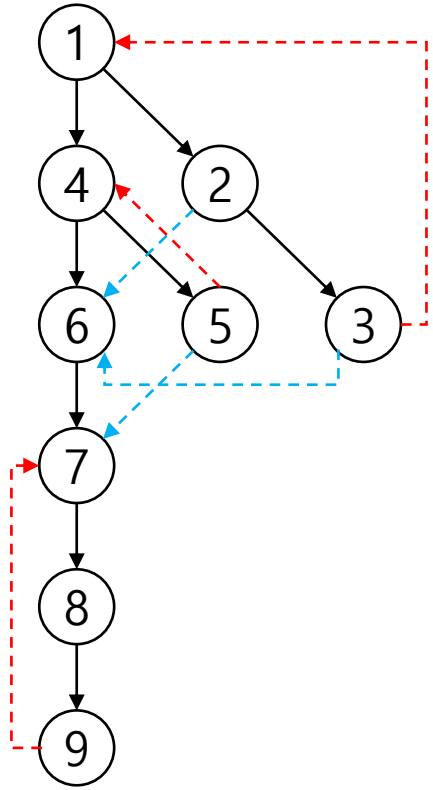




SCC

- Kosaraju's Algorithm
  - 알고리즘
    - DFS 하면서 빠져나온 순서대로 정점을 나열
      - 9 8 7 6 5 4 3 2 1
    - 간선의 방향을 모두 뒤집음
    - 가장 늦게 빠져나온 정점부터 DFS
    - DFS를 할 때마다 SCC를 한 개씩 찾음

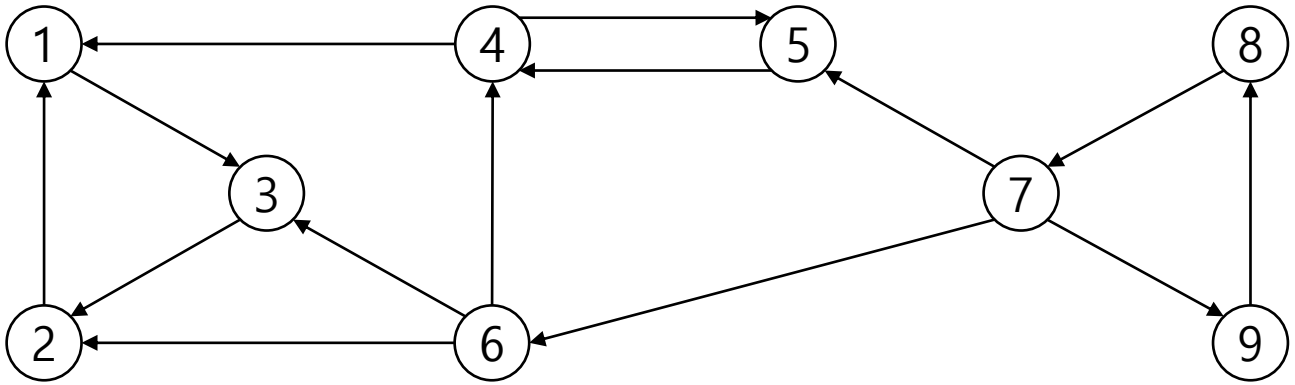


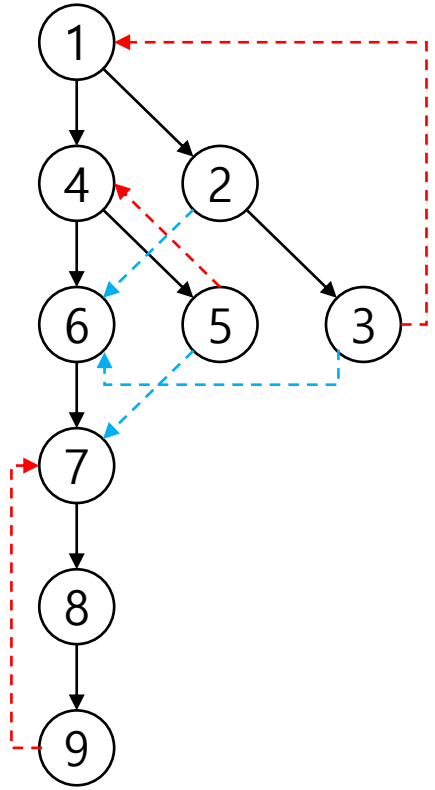


질문?

SCC

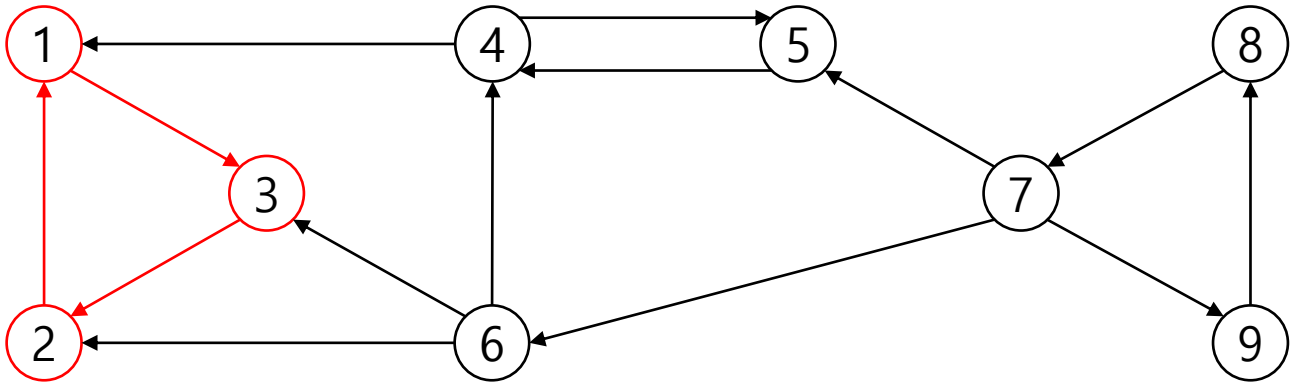
- Kosaraju's Algorithm
  - 시간 복잡도: O(V+E)
  - 위상 정렬 순서대로 SCC를 구함
    - DFS Tree의 위쪽부터 떼어낸다고 생각해 보자.

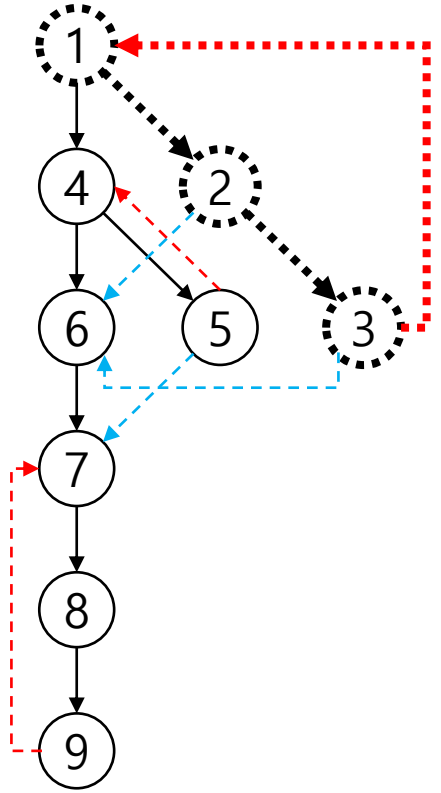




SCC

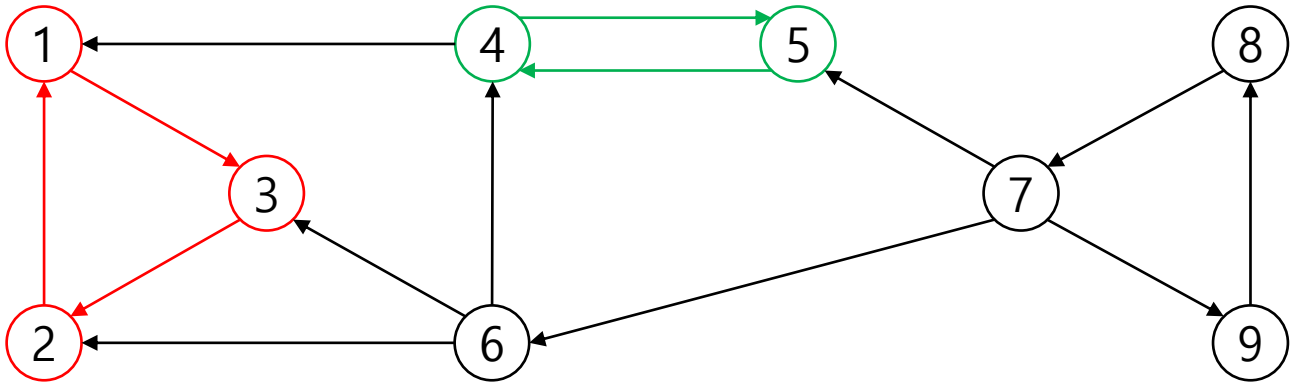
- Kosaraju's Algorithm
  - 시간 복잡도: O(V+E)
  - 위상 정렬 순서대로 SCC를 구함
    - DFS Tree의 위쪽부터 떼어낸다고 생각해 보자.

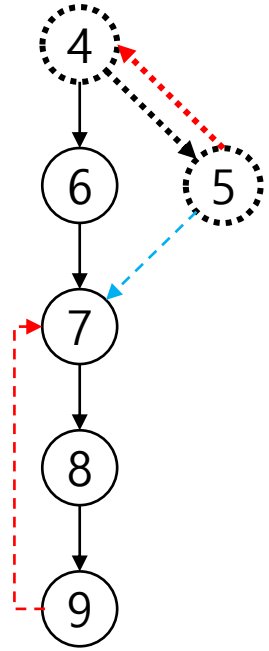




SCC

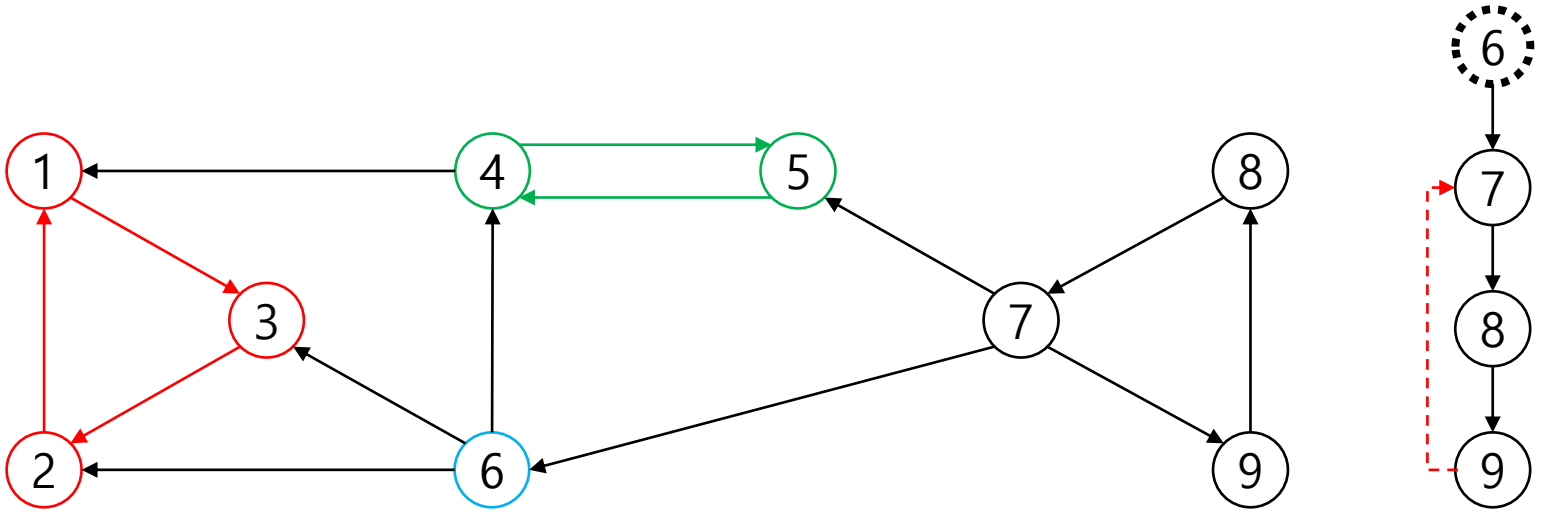
- Kosaraju's Algorithm
  - 시간 복잡도: O(V+E)
  - 위상 정렬 순서대로 SCC를 구함
    - DFS Tree의 위쪽부터 떼어낸다고 생각해 보자.





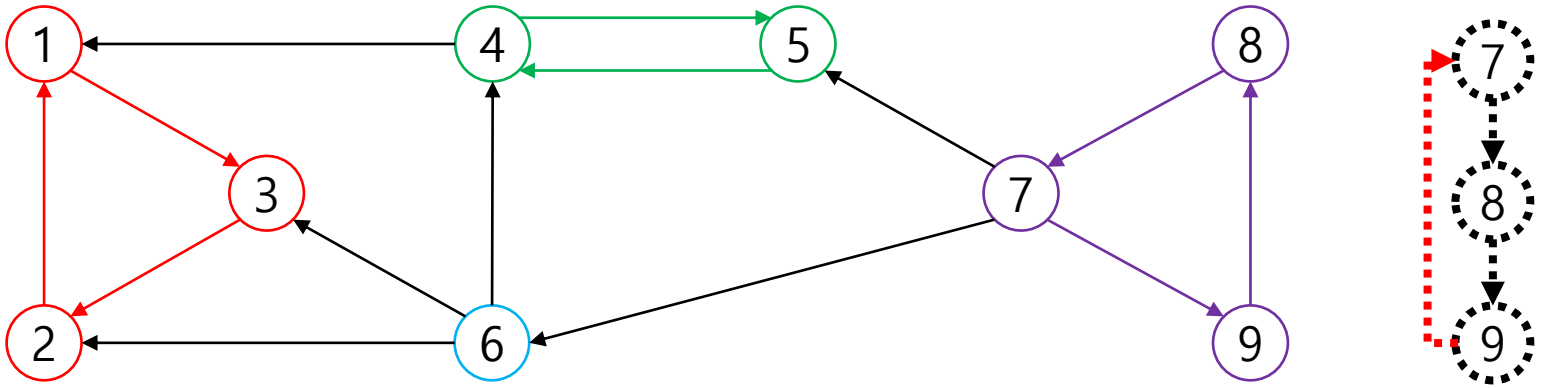
SCC

- Kosaraju's Algorithm
  - 시간 복잡도: O(V+E)
  - 위상 정렬 순서대로 SCC를 구함
    - DFS Tree의 위쪽부터 떼어낸다고 생각해 보자.



SCC

- Kosaraju's Algorithm
  - 시간 복잡도: O(V+E)
  - 위상 정렬 순서대로 SCC를 구함
    - DFS Tree의 위쪽부터 떼어낸다고 생각해 보자.



SCC

- Kosaraju's Algorithm
  - 정당성 증명
    - 역방향 그래프에서 DFS 할 때, v에서 출발하는 DFS는 v와 같은 SCC에 포함된 모든 정점을 방문함
      - 그래프 G에서 u, v가 같은 SCC에 속하면 u, v는 서로 이동 가능함
      - 역방향 그래프 G'에서도 서로 이동 가능하므로 v에서 출발하는 DFS는 u를 방문함
    - 역방향 그래프에서 아직 방문하지 않은 정점 중 finish time이 가장 큰 정점 v에서 출발하는 DFS는 v와 다른 SCC에 포함된 정점을 방문하지 않는다.
      - 서로 다른 두 SCC s_1, s_2가 있고, 각 SCC에서 finish time이 가장 큰 정점을 v_1, v_2라고 하자. (finish(v_1) > finish(v_2))
      - G에 $s_2 \rightarrow s_1$ 간선이 없으면 G'에 $s_1 \rightarrow s_2$ 간선 없으므로 v_1에서 출발하는 DFS는 s_2를 방문하지 않음
      - 따라서 G에 $s_2 \rightarrow s_1$ 간선이 없음을 보이면 됨
        - $s_2 \rightarrow s_1$이 back edge가 되기 위해서는 같은 SCC에 속해야 함
        - back edge가 아닌 다른 간선이 되기 위해서는 finish(v_1) < finish(v_2)가 되어야 함

질문?

SCC

- BOJ 2150

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, M, C[10101];
vector<int> G[10101], R[10101], V;
vector<vector<int>> S;
void AddEdge(int s, int e){
    G[s].push_back(e);
    R[e].push_back(s);
}
void DFS1(int v){
    C[v] = -1;
    for(auto i : G[v]) if(!C[i]) DFS1(i);
    V.push_back(v);
}
void DFS2(int v, int c){
    C[v] = c; S.back().push_back(v);
    for(auto i : R[v]) if(C[i] == -1) DFS2(i, c);
}
int GetSCC(){
    for(int i=1; i<=N; i++) if(!C[i]) DFS1(i);
    reverse(V.begin(), V.end());
    int cnt = 0;
    for(auto i : V) if(C[i] == -1) S.emplace_back(), DFS2(i, ++cnt);
    return cnt;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1,s,e; i<=M; i++) cin >> s >> e, AddEdge(s, e);
    int K = GetSCC();
    for(auto &vec : S) sort(vec.begin(), vec.end());
    sort(S.begin(), S.end());
    cout << K << "\n";
    for(const auto &vec : S){
        for(auto i : vec) cout << i << " ";
        cout << -1 << "\n";
    }
}
```

SCC

- BOJ 20504 I번은 쉬운 문제
  - in degree가 0인 SCC의 크기의 합을 구하는 문제

- BOJ 4013 ATM
  - 그래프에서 그냥 DP를 시도하면 사이클이 생기기 때문에 안 됨
  - SCC를 묶으면 DAG가 되므로 DP 할 수 있음

질문?

2-SAT

2-SAT

- 2-SAT
  - 충족 가능성 문제 (SAT)
    - $(X_{A_1} \vee X_{A_2} \vee \cdots \vee X_{A_n}) \wedge (X_{B_1} \vee X_{B_2} \vee \cdots \vee X_{B_m}) \wedge \cdots$ 를 참으로 만드는 변수 X_i의 값을 찾는 문제
    - 변수 앞에 ¬ (not)이 붙어있을 수도 있음
  - k-SAT
    - 한 절에 최대 k개의 변수가 들어간 충족 가능성 문제
    - 임의의 SAT 문제는 3-SAT으로 바꿀 수 있음. 3-SAT은 NP-Complete
    - 하지만 2-SAT은 선형 시간에 해결할 수 있음

2-SAT

- 2-SAT
  - $(A \vee B)$꼴의 절 여러 개가 and 연산으로 연결되어 있음
  - 각각의 절을 모두 참으로 만들어야 함

  - $(A \vee B) \Leftrightarrow (\neg A \Rightarrow B) \wedge (\neg B \Rightarrow A)$
    - 둘 중 하나는 참이 되어야 함
    - A가 거짓이면 B는 참, B가 거짓이면 A는 참
    - $A \vee B$ 꼴의 식보다는 $A \Rightarrow B$ 꼴의 식으로 생각하는 것이 편함

2-SAT

- 2-SAT
  - 몇 가지 예시를 보자.
  - $(A \vee B) \wedge (\neg B \vee A)$
    - $\neg A \Rightarrow B \Rightarrow A$
    - 변수 A만 놓고 보면$\neg A \Rightarrow A$
    - $\neg A$가 거짓이면 A는 아무거나 될 수 있으므로 $A = true$로 하면 충족 가능
  - $(A \vee B) \wedge (\neg B \vee A) \wedge (\neg A \vee B) \wedge (\neg B \vee \neg A)$
    - $\neg A \Rightarrow B \Rightarrow A \Rightarrow B \Rightarrow \neg A$
    - 변수 A만 놓고 보면$\neg A \Rightarrow A \Rightarrow \neg A$
    - A가 어떤 값이 되더라도 충족 불가능
  - $\neg A \Rightarrow A$이고 $A \Rightarrow \neg A$이면 충족 불가능
  - SCC를 쓰면 된다!

2-SAT

- 2-SAT
  - 그래프 구성
    - 각 변수 X_i마다 X_i를 나타내는 정점과 $\neg X_i$를 만족하는 정점을 만들고
    - $(A \vee B)$ 주어지면 $\neg A \rightarrow B$, $\neg B \rightarrow A$ 간선 생성
    - X_i와 $\neg X_i$가 같은 SCC에 속하면 불가능
    - 아니면 가능
  - 변수 값 추적
    - X_i와 $\neg X_i$는 서로 다른 SCC에 속함
    - 만약 X_i가 속한 SCC가 위상 정렬 상에서 뒤에 있으면 $\neg X_i \Rightarrow X_i$이므로 $X_i = true$
    - X_i가 속한 SCC가 앞에 있으면 $X_i \Rightarrow \neg X_i$이므로 $X_i = false$

질문?

2-SAT

- BOJ 11281 2-SAT - 4
  - 단순 구현

- BOJ 3648 아이돌
  - 단순 구현

2-SAT

- BOJ 16367 TV Show Game
  - R을 true, B를 false라고 생각하자.
  - 3개의 조건 중 2개 이상 만족해야 함을 2-CNF로 표현해야 함
    - 3개의 조건 중 2개 이상을 만족한다면
    - 비둘기집의 원리에 의해 2개 선택했을 때 무조건 만족하는 조건 1개 이상 선택됨
    - (X or Y) and (Y or Z) and (Z or X)

- BOJ 1739 도로 정비하기
  - 불대수에서 $A + BC = (A + B)(A + C)$임을 이용

질문?

과제

- 필수
  - 2150 Strongly Connected Com...
  - 20504 I번은 쉬운 문제
  - 4013 ATM
  - 11281 2-SAT - 4
  - 3648 아이돌
  - 16367 TV Show Game
- 심화
  - 1739 도로 정비하기
  - 20942 신촌지역 초중고등학생 프로...
  - 14866 산만한 고양이
  - 3654 L퍼즐
  - 25011 칠하기
  - 14737 Dev, Please Add This!