이분 매칭 / 네트워크 플로우 SCC / 2-SAT

AlCall / Sinbaram 2019 알고리즘 세미나

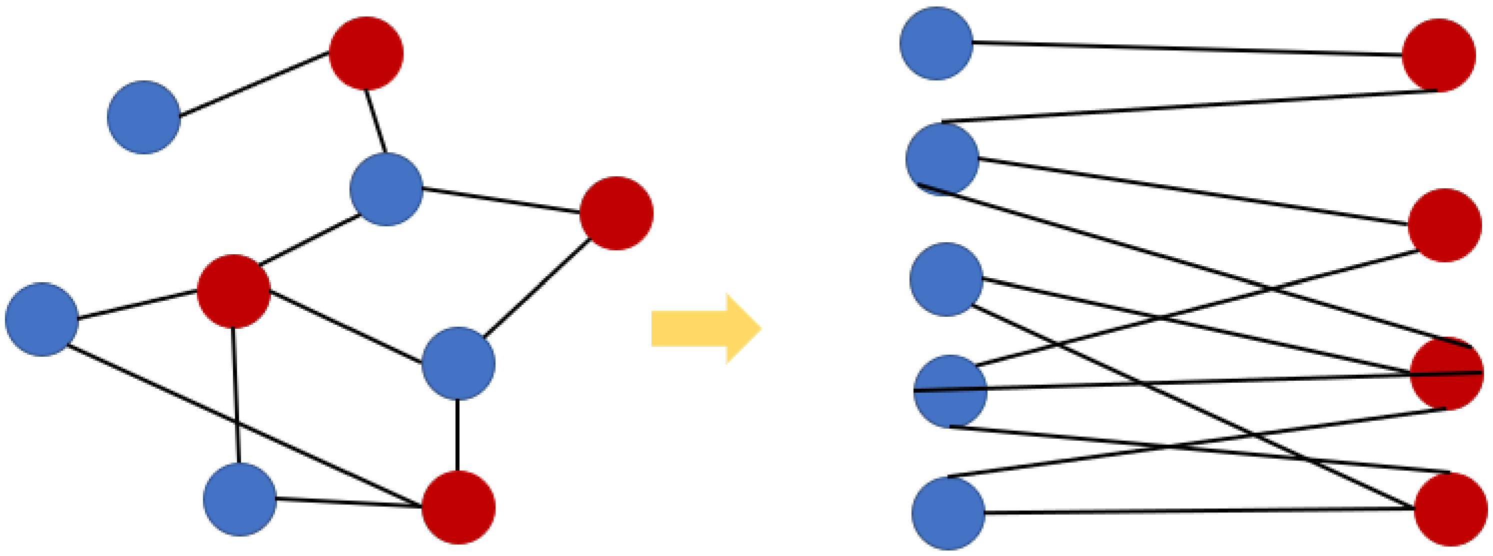
201624476 박상운

유의사항

- 어려우니까 마스터 안해도 됩니다. (대회 입상하려면 마스터 합니다.)
- 이 발표자료만으로는 충분한 공부자료가 되지 못합니다.

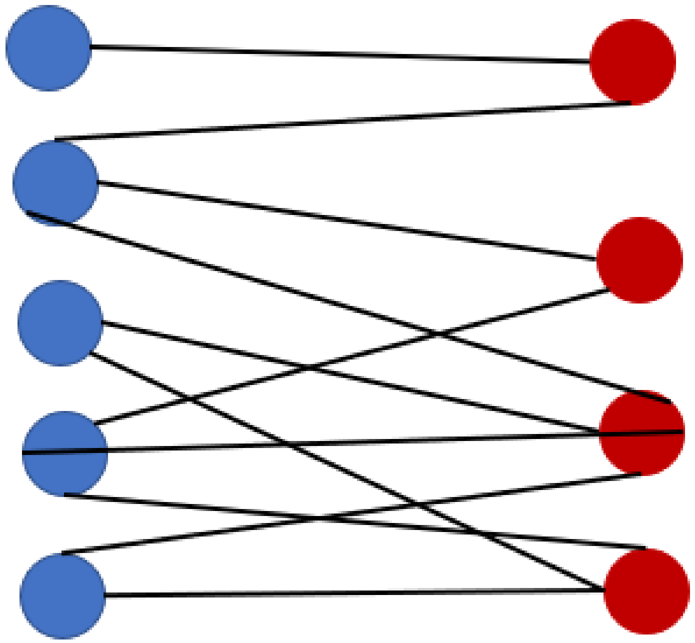
1. 이분매칭

이분 그래프



짝을 맺어주자

- 남자(왼쪽, 파랑)와 여자(오른쪽, 빨강)는 자기랑 선분으로 이어진 상대방과 짝을 짓는다.
- 이미 짝을 지은 사람은 또 다른 사람과 짝을 짓지 못한다.
- 최대한 많은 짝의 수 / 가짓수

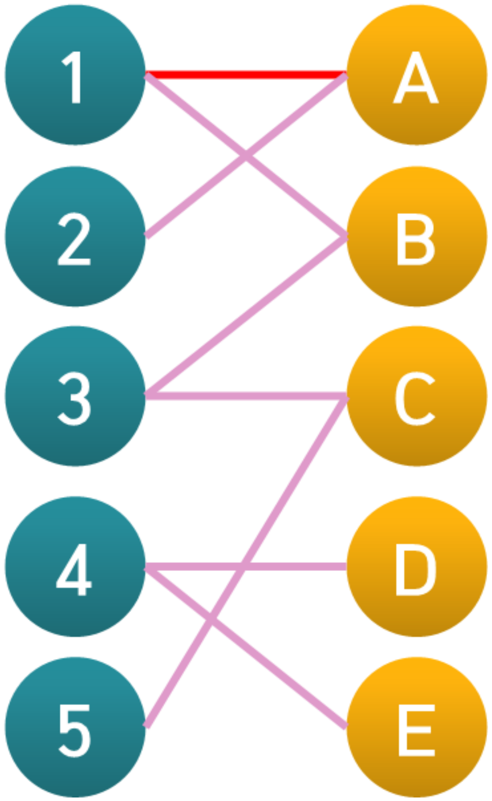


이분 매칭 구현

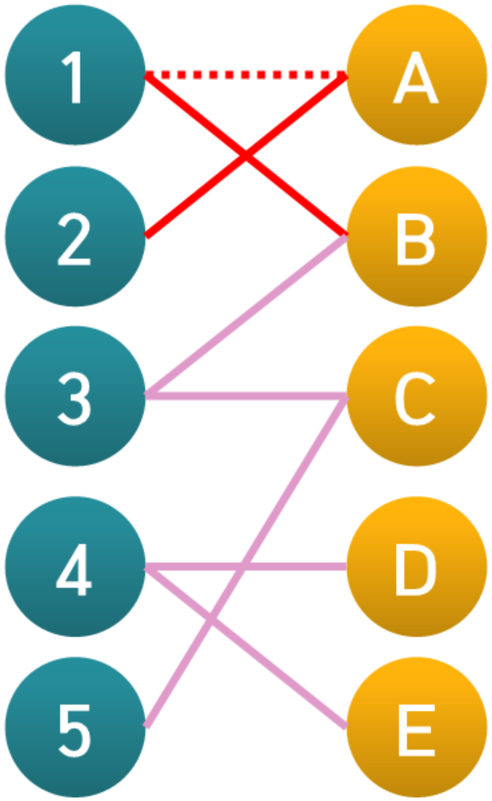
- Hopcroft-Karp Algorithm
  - $O(E\sqrt{V})$에서의 최대 매칭을 구하는 알고리즘
  - 짜기 귀찮다

- DFS
  - $O(EV)$
  - 짜기 쉽다

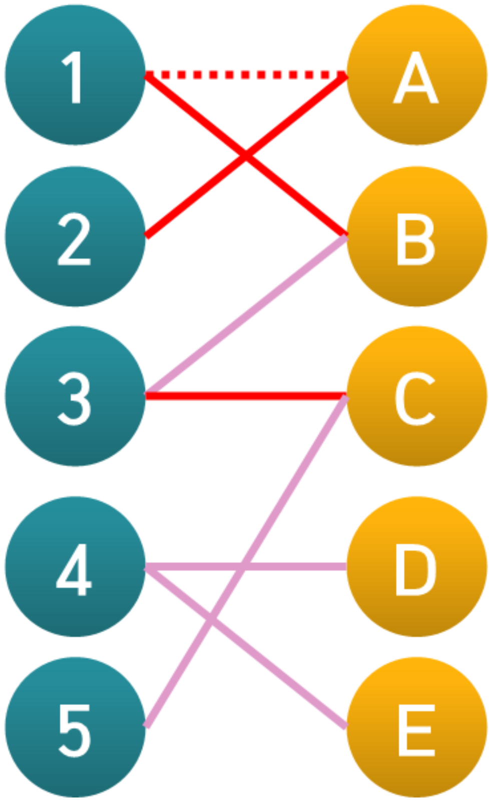
이분 매칭의 과정



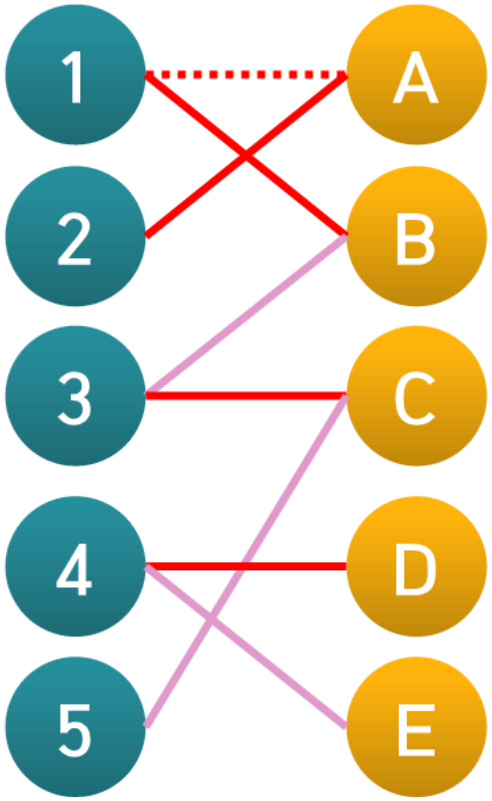
1→A



cannot 2→A
So 1→B, 2→A



3→C



4→D

이분 매칭 코드

```
#define MAX_N 1000001

int n;
int adj[MAX_N][2];
int aMatch[MAX_N];
int bMatch[MAX_N];
int visit[MAX_N];
int visitCnt = 1;

bool dfs(int a) {
    if (visit[a] == visitCnt)
        return false;
    visit[a] = visitCnt;
    for (int next = 0; next < 2; next++) {
        if (adj[a][next]) {
            int b = adj[a][next];
            if (bMatch[b] == -1 || dfs(bMatch[b])) {
                aMatch[a] = b;
                bMatch[b] = a;
                return true;
            }
        }
    }
    return false;
}
```

```

int bipartiteMatch() {
    memset(aMatch, -1, sizeof(aMatch));
    memset(bMatch, -1, sizeof(bMatch));
    int size = 0;
    for (int start = 0; start < n; start++) {
        visitCnt++;
        size += dfs(start);
    }
    return size;
}
```

https://www.crocus.co.kr/744 [Crocus]

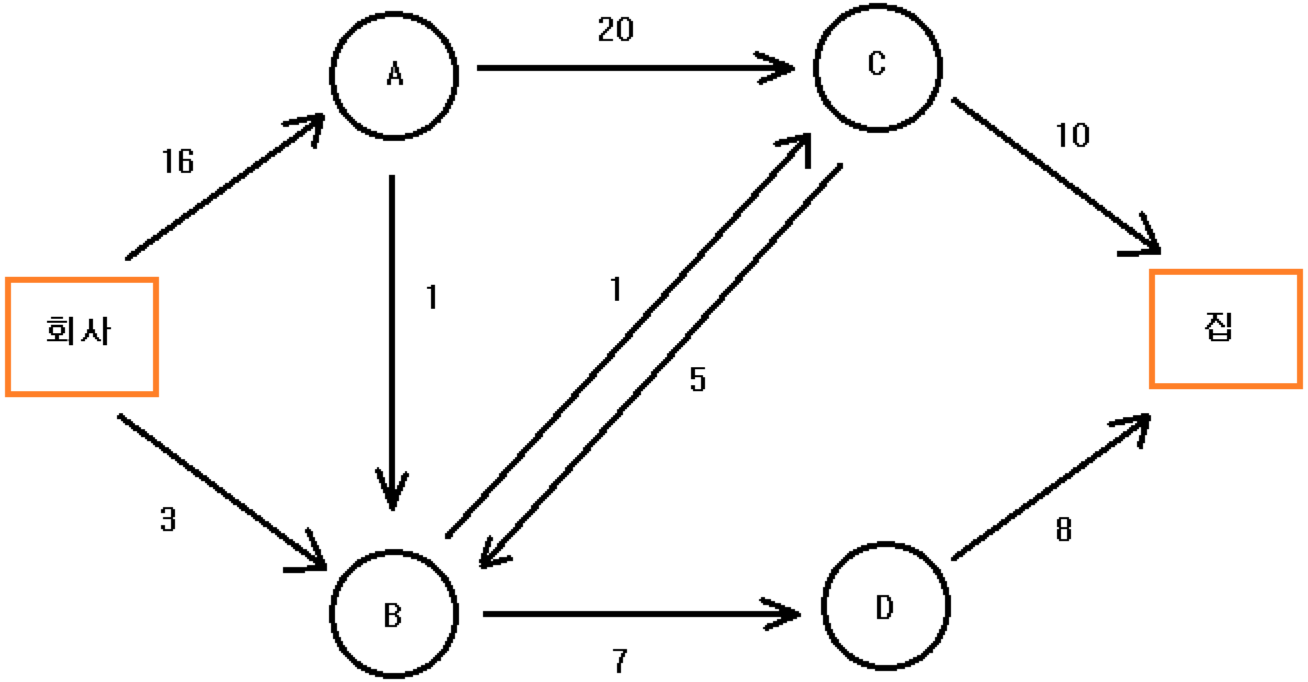
2. 네트워크 플로우

네트워크 플로우란

- 그래프의 두 정점에서의 최대 유량(maximum flow)를 구하는 문제
- 대용량 데이터 전송, 교통 상황, 하수도 처리 등 다양한 실생활 문제와 접목됨

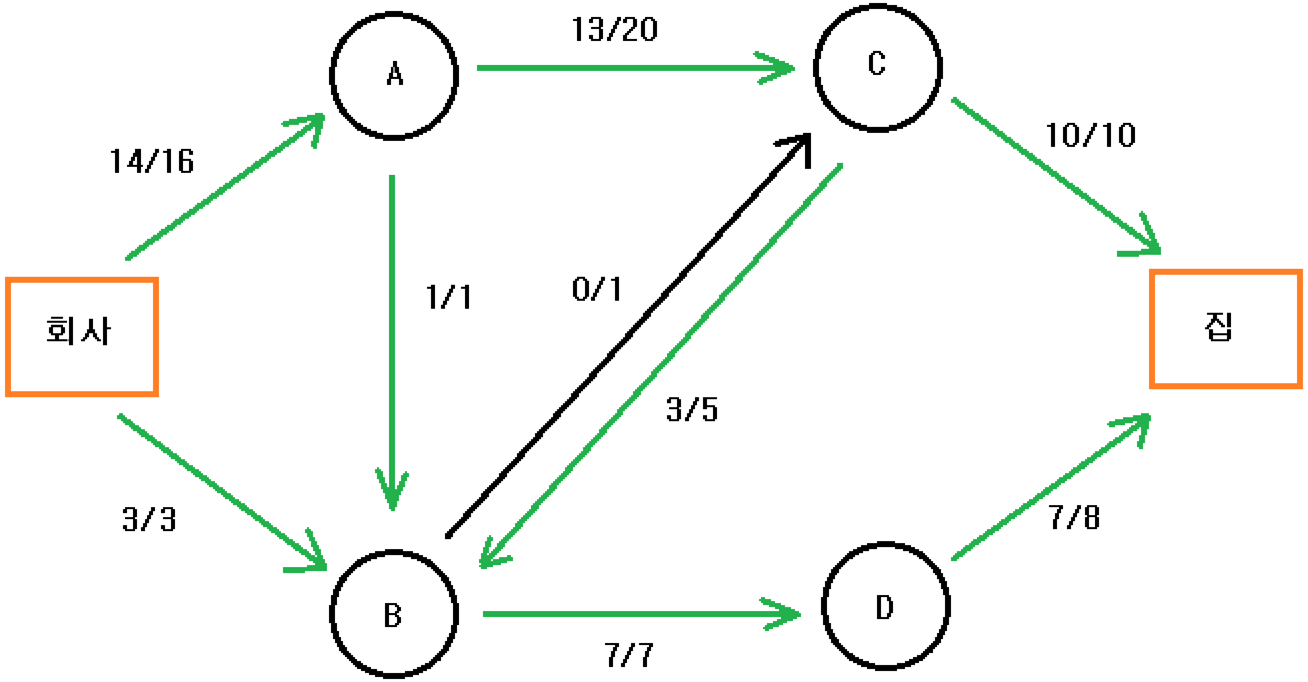
유량 그래프

- 방향과 한계 유량을 가지고 있는 그래프



네트워크 플로우의 목적

- 회사에서 집으로 가는 자동차가 무한정 나오는 상황일때, 길이 어떻게 점유되는가



Edmond-Karp Algorithm

- $O(VE^2)$
- https://wondy1128.tistory.com/181
- https://ideone.com/2OlnzF

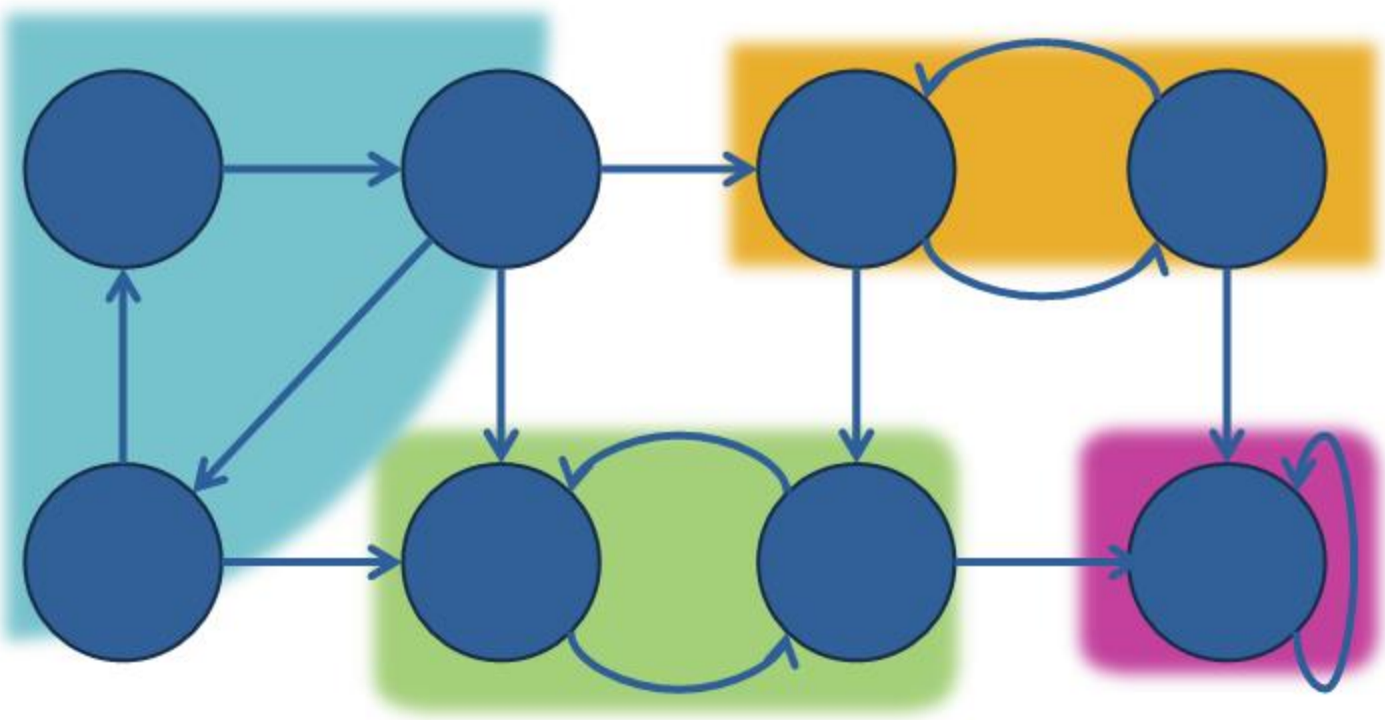
Dinic

- $O(EV^2)$
- https://jason9319.tistory.com/150

3. SCC

SCC(Strongly Connected Components)

- 같은 SCC 내의 임의의 두 정점 A,B사이의 경로가 항상 존재하며
- SCC 끼리는 사이클이 존재하지 않는다.



뭐에다 쓰는가

- 그래프 압축
- 2-SAT
- https://jason9319.tistory.com/98
- https://ideone.com/vgpZoz

4. 2-SAT

2-SAT(2-satisfiability)

- 충족 가능성 문제(문제에서 주어진 조건을 만족하는 해가 존재하는가?)
- 임의의 논리식이 주어졌을때, 해당 식을 만족하는 논리 변수의 조합을 찾기
- 이때 각 절(Closure)의 변수가 최대 2개인 경우를 2-SAT이라 함

2-SAT(2-satisfiability)

$f = (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_1 \vee x_3) \wedge (x_3 \vee x_2)$

x1,x2 = F, x3=T 이면 참

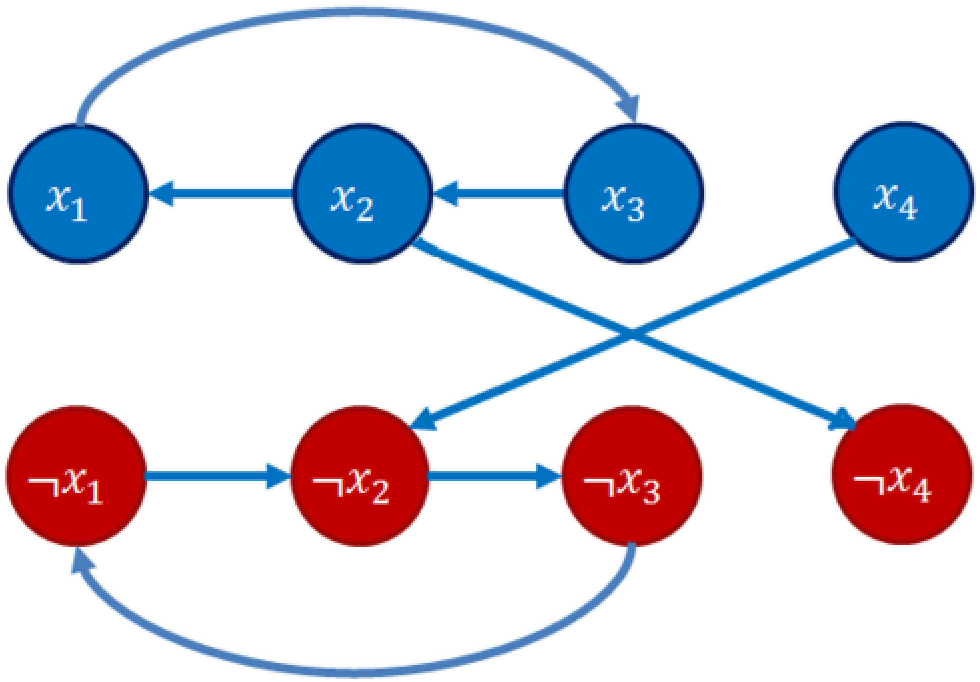
$f = (x_1 \vee x_1) \wedge (\neg x_1 \vee \neg x_1)$

항상 거짓

2-SAT과 SCC

- 각각의 변수와 그의 NOT형 변수를 정점으로 두고, $\neg p \vee q = p \rightarrow q$임을 바탕으로 연결 관계를 추출

$$f = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1) \wedge (\neg x_4 \vee \neg x_2)$$

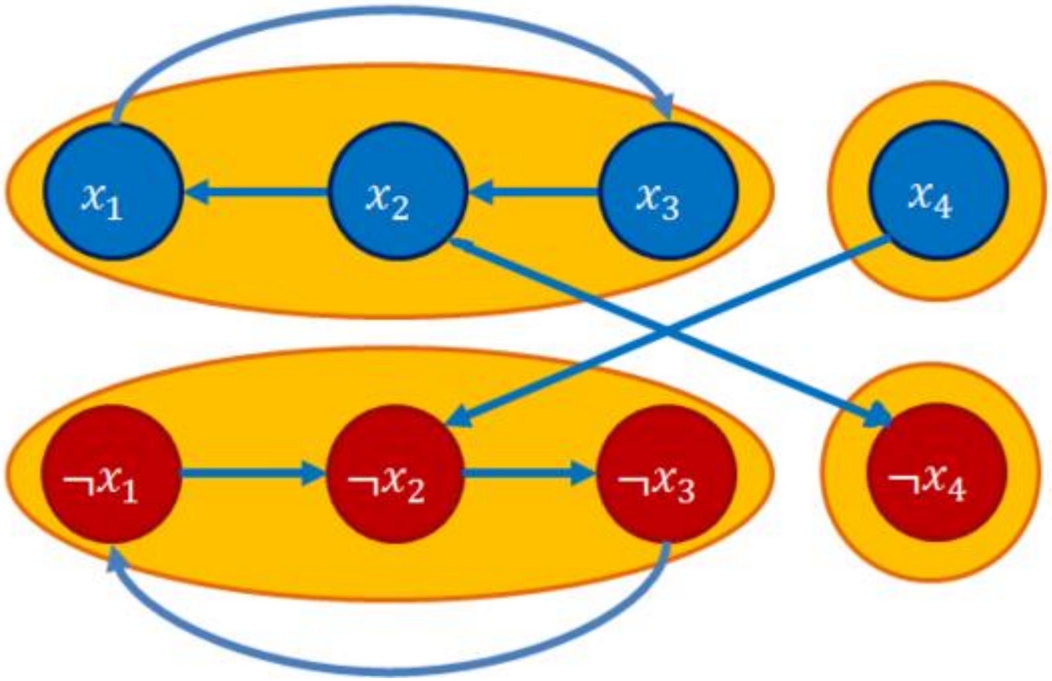


x1 → x2
¬x2 → ¬x1
x2 → x3
¬x3 → ¬x2
¬x1 → x3
¬x3 → x1
¬x3 → x2
¬x2 → x3

2-SAT과 SCC

- SCC 적용

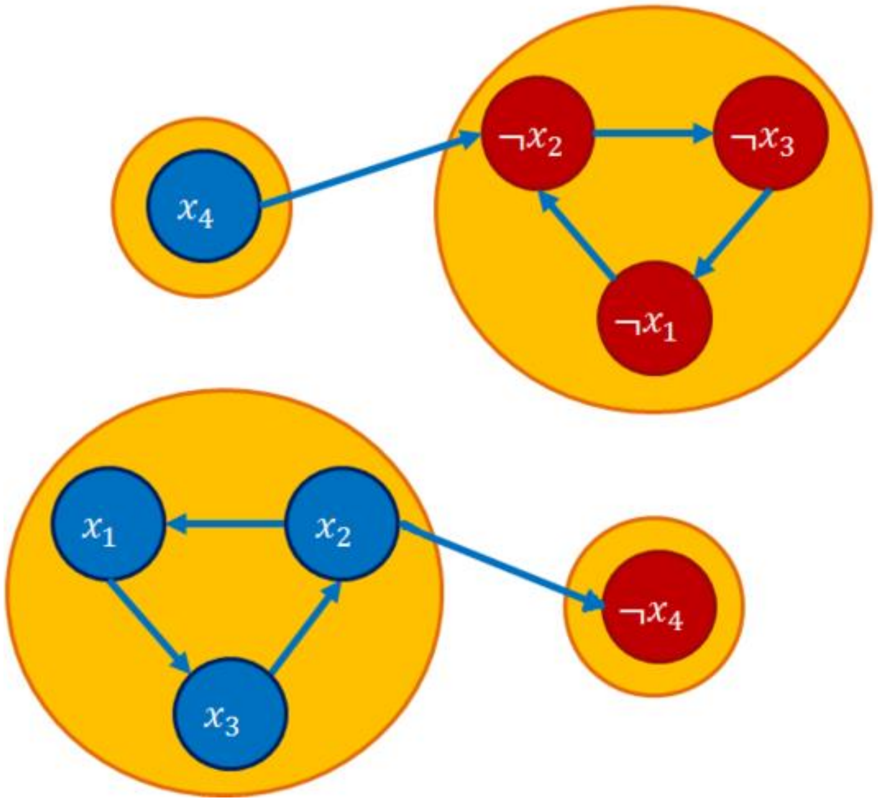
$$f = (x_1 \lor \neg x_2) \land (x_2 \lor \neg x_3) \land (x_3 \lor \neg x_1) \land (\neg x_4 \lor \neg x_2)$$



2-SAT과 SCC

- x → ~x, ~x→x 인 경로가 존재하지 않으므로 만족 가능

$$f = (x_1 \lor \neg x_2) \land (x_2 \lor \neg x_3) \land (x_3 \lor \neg x_1) \land (\neg x_4 \lor \neg x_2)$$



참고 페이지 / 2-SAT

- https://blog.qwaz.io/problem-solving/scc%EC%99%80-2-sat
- https://blog.naver.com/PostView.nhn?blogId=kks227&logNo=220803009418
- https://algospot.com/wiki/read/2-SAT