25-1 이니로
알고리즘 멘토링

멘토 - 김수성

공유기 설치 / 2110

백준 2110 / https://www.acmicpc.net/problem/2110

문제

도현이의 집 N개가 수직선 위에 있다. 각각의 집의 좌표는 x_1, ..., x_N이고, 집 여러개가 같은 좌표를 가지는 일은 없다.

도현이는 언제 어디서나 와이파이를 즐기기 위해서 집에 공유기 C개를 설치하려고 한다. 최대한 많은 곳에서 와이파이를 사용하려고 하기 때문에, 한 집에는 공유기를 하나만 설치할 수 있고, 가장 인접한 두 공유기 사이의 거리를 가능한 크게 하여 설치하려고 한다.

C개의 공유기를 N개의 집에 적당히 설치해서, 가장 인접한 두 공유기 사이의 거리를 최대로 하는 프로그램을 작성하시오.

입력

첫째 줄에 집의 개수 N ($2 \le N \le 200,000$)과 공유기의 개수 C ($2 \le C \le N$)이 하나 이상의 빈 칸을 사이에 두고 주어진다. 둘째 줄부터 N개의 줄에는 집의 좌표를 나타내는 x_i ($0 \le x_i \le 1,000,000,000$)가 한 줄에 하나씩 주어진다.

출력

첫째 줄에 가장 인접한 두 공유기 사이의 최대 거리를 출력한다.

공유기 설치 / 2110

출력

첫째 줄에 가장 인접한 두 공유기 사이의 최대 거리를 출력한다.

예제 입력 1 복사

```
5 3
1
2
8
4
9
```

예제 출력 1 복사

```
3
```

힌트

공유기를 1, 4, 8 또는 1, 4, 9에 설치하면 가장 인접한 두 공유기 사이의 거리는 3이고, 이 거리보다 크게 공유기를 3개 설치할 수 없다.

공유기 설치 / 2110

예제 입력	예제 출력
5 3 1 2 8 4 9	3

1 4 9에 공유기를 설치하면 각 거리가 3 5가 됨
거리의 최솟값이 3보다 크게 공유기를 설치할 수 없음

공유기 설치 / 2110

최적화 문제

설치한 공유기 사이의 거리의 최솟값의 최댓값

결정 문제

공유기 사이의 거리의 최솟값이 K 일 때
M개 이상의 공유기를 설치 할 수 있는가

공유기 설치 / 2110

결정 문제

공유기 사이의 거리의 최솟값이 K 일 때
M개 이상의 공유기를 설치 할 수 있는가

최솟값을 크게 만들려면
공유기를 적게 설치해야 함

K 값이 커지면 공유기를 M개 이상 설치 할 수 없음

공유기 설치 / 2110

결정 문제

공유기 사이의 거리의 최솟값이 K 일 때
M개 이상의 공유기를 설치 할 수 있는가

K 값이 커지면 공유기를 M개 이상 설치 할 수 없음
-> decision(K) 1111100000
최댓값을 구하면 됨

공유기 설치 / 2110

결정 문제

공유기 사이의 거리의 최솟값이 K 일 때
M개 이상의 공유기를 설치 할 수 있는가

배열을 정렬하고 전에 공유기를 설치한 장소와
거리가 K이상이 되었을 때 공유기를 설치함
-> 항상 공유기 사이의 거리는 K 이상

이 때 설치한 공유기 개수가 M개 이상이면
Decision(K)는 1이 됨

공유기 설치 / 2110

시간 복잡도

배열 정렬 -> O(N log N)

결정 함수 -> O(N)

매개 변수 탐색 -> O(log 1e9) = 30...

O(N log N + N log 1e9)

공유기 설치 / 2110

C++

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m;
    for(int i = 1;i <= n;i++) cin >> a[i];
    sort(a + 1, a + n + 1);

    cout << maximization();

    return 0;
}
```

```cpp
bool decision(ll cur){
    // cnt -> 설치한 공유기 개수
    // last -> 마지막으로 설치한 공유기 위치
    ll cnt = 0, last = -1e12; // last는 음의 무한대로 초기화
    for(int i = 1;i <= n;i++){
        // 마지막으로 설치한 공유기의 위치와
        // 현재 위치의 차이가 K보다 작다면 건너 뜀
        if(a[i] - last < cur) continue;
        last = a[i]; cnt++;
    }

    // 설치한 공유기 개수가 M 이상이면
    // 공유기 거리의 차이의 최솟값을 K로 만들 수 있음
    return cnt >= m;
}

ll maximization(){
    // 최솟값은 1, 최댓값은 1e9
    // 실수 오차 있을 수 있으니 넉넉하게 2e9로 잡음
    ll lo = 1, hi = 2e9;
    while(lo < hi){
        ll mid = (lo + hi + 1) / 2; // 올림 값 사용
        if(decision(mid)) lo = mid;
        else hi = mid - 1;
    }

    return lo;
}
```

공유기 설치 / 2110

Python

```python
n, m = list(map(int, input().rstrip().split()))
a = []
for _ in range(n):
    x = int(input().rstrip())
    a.append(x)
a.sort()

def decision(cur):
    # cnt -> 설치한 공유기 개수
    # last -> 마지막으로 설치한 공유기 위치
    cnt = 0
    last = -1e12 # last는 음의 무한대로 초기화

    for i in a:
        # 마지막으로 설치한 공유기의 위치와
        # 현재 위치의 차이가 K보다 작다면 건너 뜀
        if i - last < cur:
            continue
        last = i
        cnt += 1

    # 설치한 공유기 개수가 M 이상이면
    # 공유기 거리의 차이의 최솟값을 K로 만들 수 있음
    return cnt >= m
```

```python
def maximization():
    # 최솟값은 1, 최댓값은 1e9
    # 실수 오차 있을 수 있으니 넉넉하게 2e9로 잡음
    lo = 1
    hi = int(2e9)
    while lo < hi:
        mid = (lo + hi + 1) // 2
        if decision(mid):
            lo = mid
        else:
            hi = mid - 1
    return lo

print(maximization())
```

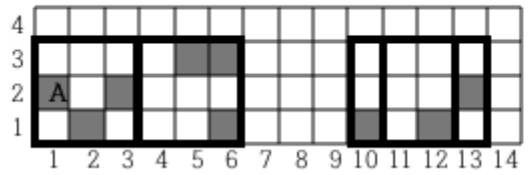
질문?

모자이크 / 2539

백준 2539 / https://www.acmicpc.net/problem/2539

1. 사용되는 색종이는 모두 크기가 같고 정사각형 모양이다.
2. 색종이 크기는 한 변의 길이로 나타내며, 원하는 크기의 색종이는 모두 구할 수 있다.
3. 모든 색종이는 반드시 도화지의 밑변에 맞추어 붙인다. 이때 색종이를 겹쳐서 붙일 수 있다.

도화지 위의 행은 다음 그림과 같이 맨 아래에서 위쪽으로 1번부터 순서대로 번호가 매겨져 있고, 열은 왼쪽에서 오른쪽으로 1번부터 번호가 매겨져 있다. 이 그림은 도화지에 가로선과 세로선을 그어서 4개의 행과 14개의 열, 그리고 56개의 칸으로 나눈 모양을 보여준다. 잘못 칠해진 칸은 회색으로 표시되어 있다.



모자이크 / 2539

입력

첫째 줄에는 도화지 위의 행의 개수와 열의 개수를 나타내는 자연수가 빈칸을 사이에 두고 주어진다. 행의 개수와 열의 개수는 모두 1000000 이하이다. 둘째 줄에는 사용할 색종이의 장수를 나타내는 자연수가 주어진다. 사용할 색종이는 100장 이하이다. 셋째 줄에는 도화지에 잘못 칠해진 칸의 개수를 나타내는 자연수가 주어진다. 잘못 칠해진 칸은 1000개 이하이다. 넷째 줄부터 마지막 줄까지 잘못 칠해진 칸의 위치가 한 줄에 하나씩 주어진다. 잘못 칠해진 칸의 위치는 빈칸을 사이에 두고 행 번호가 주어진 다음 열 번호가 주어진다.

출력

첫째 줄에 주어진 장수의 색종이를 사용하여 잘못 칠해진 칸을 모두 가릴 수 있는 가장 작은 색종이의 크기가 몇 cm인지를 나타내는 자연수를 출력한다.

예제 입력 1 복사

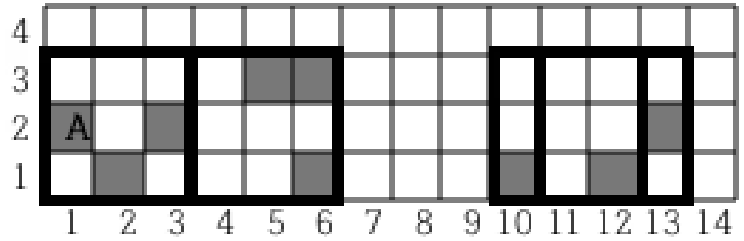
```
4 14
4
9
1 2
2 1
2 3
1 6
3 5
1 10
3 6
1 12
2 13
```

예제 출력 1 복사

```
3
```

모자이크 / 2539

정사각형을 밑변에 맞춰서 L개를 배치 할 때
모든 검은 칸을 가릴 수 있는 정사각형의 최소 크기



다음과 같이 정사각형을 배치하면
크기가 3인 정사각형으로 모든 검은 칸을 가릴 수 있음

모자이크 / 2539

최적화 문제
정사각형을 밑변에 맞춰서 L개를 배치 할 때
모든 검은 칸을 가릴 수 있는 정사각형의 길이의 최솟값

결정 문제
정사각형의 길이가 K일 때 모든 검은 칸을
L개 이하의 정사각형으로 가릴 수 있는가

모자이크 / 2539

결정 문제
정사각형의 길이가 K일 때 모든 검은 칸을
L개 이하의 정사각형으로 가릴 수 있는가

당연히 정사각형의 길이가 커지면 더 적은 개수로
검은 칸을 가릴 수 있음

decision(K) -> 00000**1**1111

최솟값을 찾으면 됨

모자이크 / 2539

결정 문제
정사각형의 길이가 K일 때 모든 검은 칸을
L개 이하의 정사각형으로 가릴 수 있는가

공유기 설치 문제처럼 x 좌표 순으로 정렬 후에
현재 검은 칸이 가려져 있으면 넘어가고
그렇지 않으면 정사각형을 배치하면 됨

y 좌표가 정사각형 보다 크면 검은 칸을 가릴 수 없음
예외 처리로 결정 문제는 O

모자이크 / 2539

시간 복잡도

N = 검은 칸의 개수

배열 정렬 -> O(N log N)

결정 함수 -> O(N)

매개 변수 탐색 -> O(log 1e6) = 20...

O(N log N + N log 1e6)

모자이크 / 2539

C++

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
using ll = long long;

const ll MAX = 201010;
ll n, m, l, black;
pair <ll, ll> a[MAX];

bool decision(ll cur){
    // cnt -> 사용한 정사각형의 개수
    // last -> 마지막으로 정사각형을 배치한 x 좌표
    ll cnt = 0, last = -1e12;
    for(int i = 1;i <= black;i++){
        // y좌표가 정사각형보다 크면
        // 검은 칸을 가릴 수 없음
        if(a[i].second > cur) return 0;

        // 현재 x좌표가 이미 배치한 정사각형
        // 안에 들어오면 이미 막힌 검은 칸
        if(a[i].first - last + 1 <= cur) continue;
        last = a[i].first; cnt++;
    }

    // 정사각형을 L개 이하로 사용 했으면
    // 검은 칸을 다 막을 수 있음
    return cnt <= l;
}
```

```cpp
ll minimization(){
    // 범위의 최댓값은 1e6
    ll lo = 1, hi = 2e6;
    while(lo < hi){
        ll mid = (lo + hi) / 2;
        if(decision(mid)) hi = mid;
        else lo = mid + 1;
    }

    return lo;
}

int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m >> l >> black;
    for(int i = 1;i <= black;i++) cin >> a[i].second >> a[i].first;
    sort(a + 1, a + black + 1);

    cout << minimization();

    return 0;
}
```

모자이크 / 2539

Python

```python
import sys
input = sys.stdin.readline

n, m = list(map(int, input().rstrip().split()))
l = int(input().rstrip())
black = int(input().rstrip())

a = []
for _ in range(black):
    y, x = list(map(int, input().rstrip().split()))
    a.append((x, y))
a.sort()

def decision(cur):
    cnt = 0 # 사용한 정사각형의 개수
    last = int(-1e12) # 마지막으로 정사각형을 배치한 x 좌표
    for x, y in a:
        # y좌표가 정사각형보다 크면
        # 검은 칸을 가릴 수 없음
        if y > cur:
            return 0

        # 현재 x좌표가 이미 배치한 정사각형
        # 안에 들어오면 이미 막힌 검은 칸
        if x - last + 1 <= cur:
            continue
        last = x
        cnt += 1

    # 정사각형을 L개 이하로 사용 했으면
    # 검은 칸을 다 막을 수 있음
    return cnt <= l
```

```python
def minimization():
    lo = 1
    # 범위의 최댓값은 1e6
    hi = int(2e6)
    while(lo < hi):
        mid = (lo + hi) // 2
        if(decision(mid)):
            hi = mid
        else:
            lo = mid + 1
    return lo

print(minimization())
```

질문?

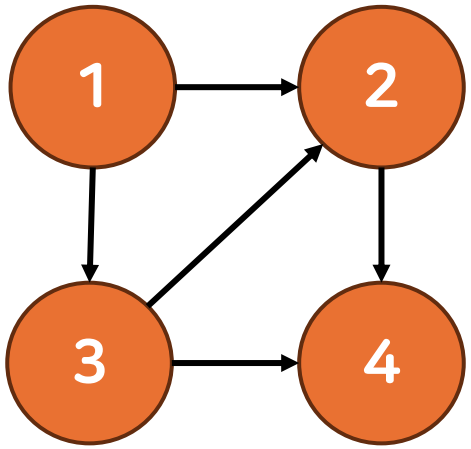
2주차 – 위상 정렬

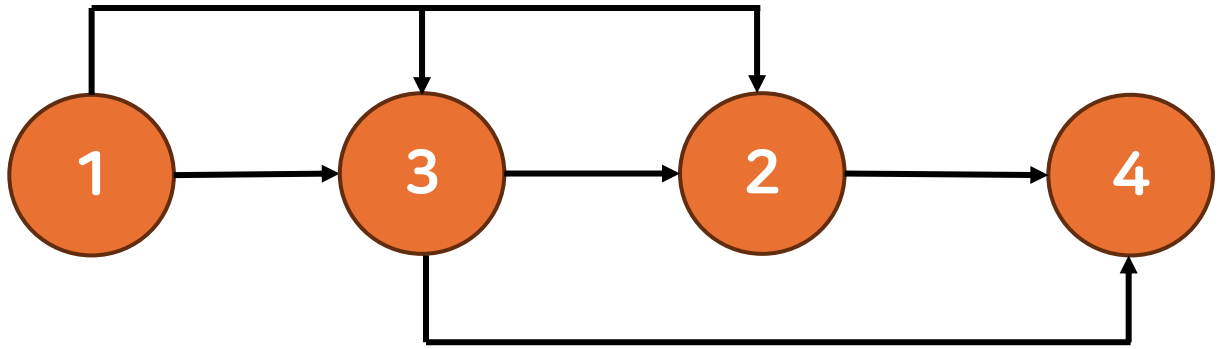
위상 정렬

위상 정렬

방향 그래프의 정점을 정렬 하는 것

간선의 방향 순서대로 정점을 나열 하는 것





줄 세우기 / 2252

백준 2252 / https://www.acmicpc.net/problem/2252

문제

N명의 학생들을 키 순서대로 줄을 세우려고 한다. 각 학생의 키를 직접 재서 정렬하면 간단하겠지만, 마땅한 방법이 없어서 두 학생의 키를 비교하는 방법을 사용하기로 하였다. 그나마도 모든 학생들을 다 비교해 본 것이 아니고, 일부 학생들의 키만을 비교해 보았다.

일부 학생들의 키를 비교한 결과가 주어졌을 때, 줄을 세우는 프로그램을 작성하시오.

입력

첫째 줄에 N(1 ≤ N ≤ 32,000), M(1 ≤ M ≤ 100,000)이 주어진다. M은 키를 비교한 횟수이다. 다음 M개의 줄에는 키를 비교한 두 학생의 번호 A, B가 주어진다. 이는 학생 A가 학생 B의 앞에 서야 한다는 의미이다.

학생들의 번호는 1번부터 N번이다.

출력

첫째 줄에 학생들을 앞에서부터 줄을 세운 결과를 출력한다. 답이 여러 가지인 경우에는 아무거나 출력한다.

줄 세우기 / 2252

출력

첫째 줄에 학생들을 앞에서부터 줄을 세운 결과를 출력한다. 답이 여러 가지인 경우에는 아무거나 출력한다.

예제 입력 1 복사

```
3 2
1 3
2 3
```

예제 출력 1 복사

```
1 2 3
```

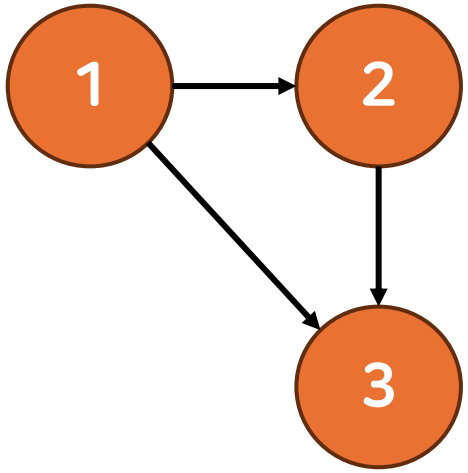
예제 입력 2 복사

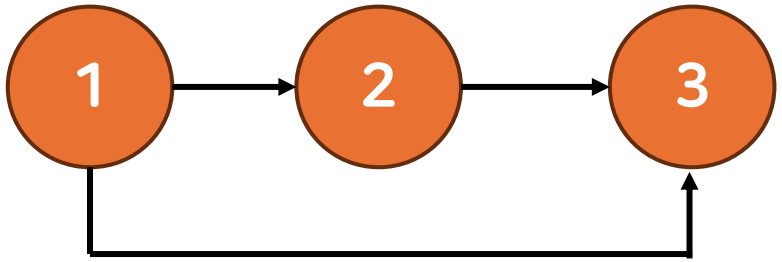
```
4 2
4 2
3 1
```

예제 출력 2 복사

```
4 2 3 1
```

줄 세우기 / 2252





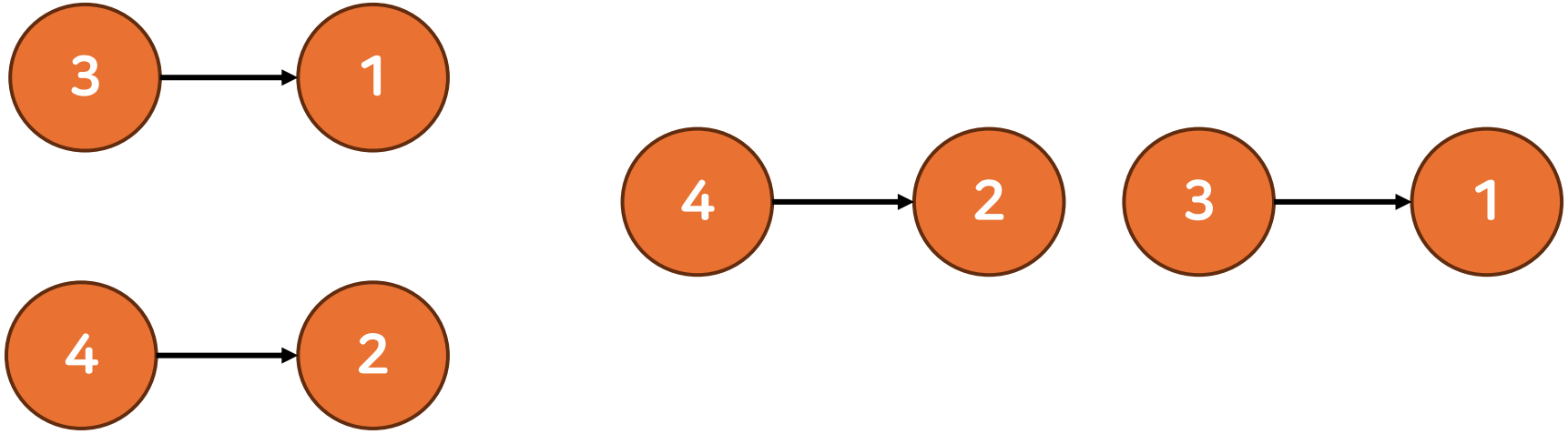
예제 입력 1 복사

```
3 2
1 3
2 3
```

예제 출력 1 복사

```
1 2 3
```

줄 세우기 / 2252



예제 입력 2 복사

```
4 2
4 2
3 1
```

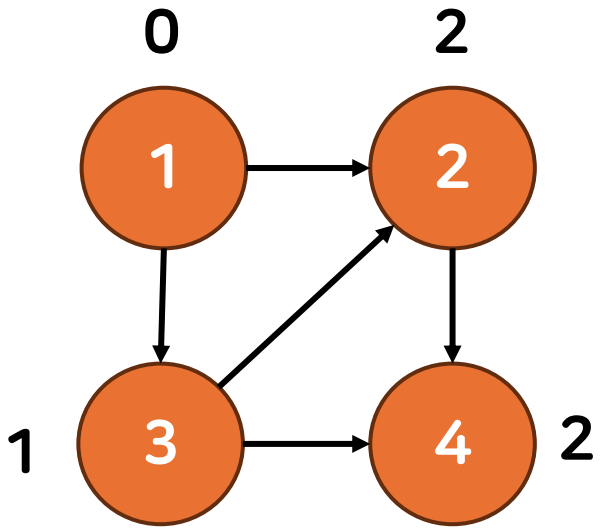
예제 출력 2 복사

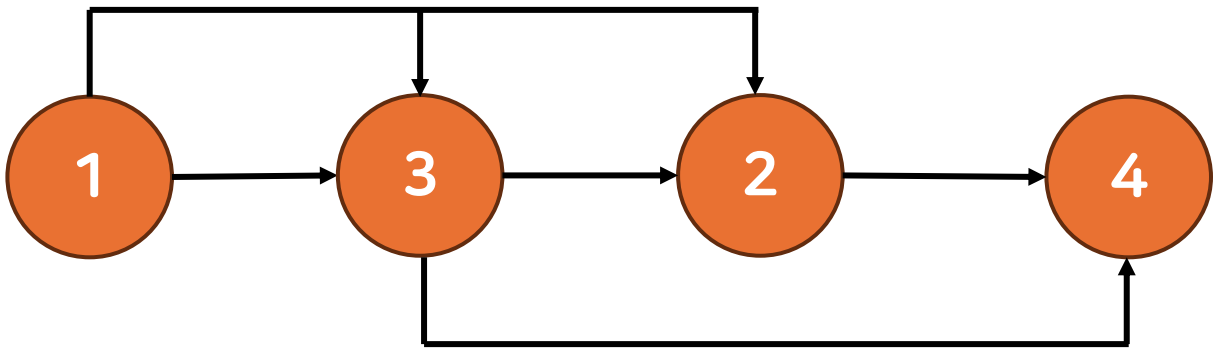
```
4 2 3 1
```

줄 세우기 / 2252

각 정점에 들어오는 간선의 수 -> indgree

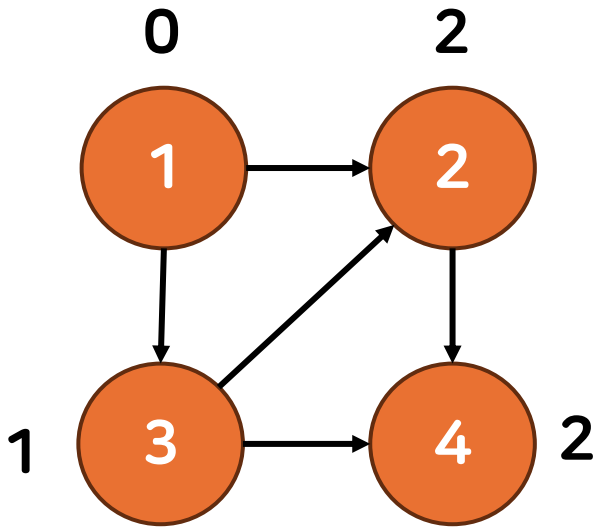
indgree가 0이 아니면 자신 이전의 값이 존재 함

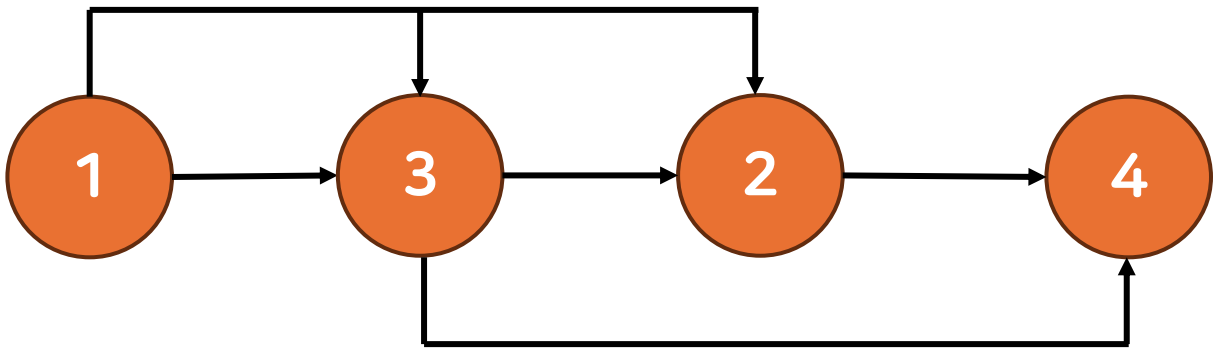




줄 세우기 / 2252

indgree가 0이 아니면 자신 이전의 값이 존재 함
-> 우선 indgree가 0인 값에서 부터 시작해야 함

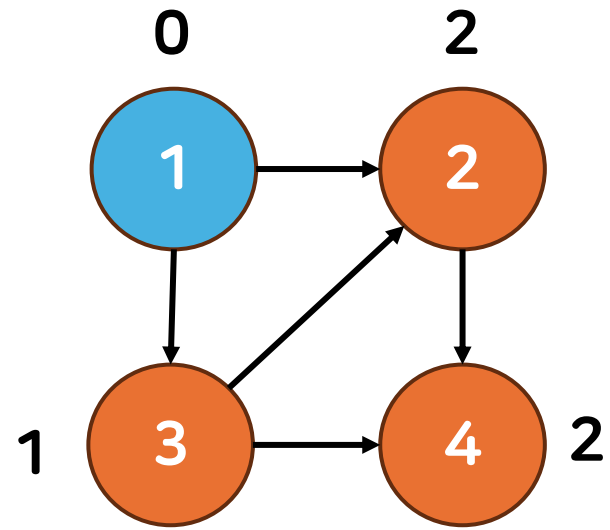




줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

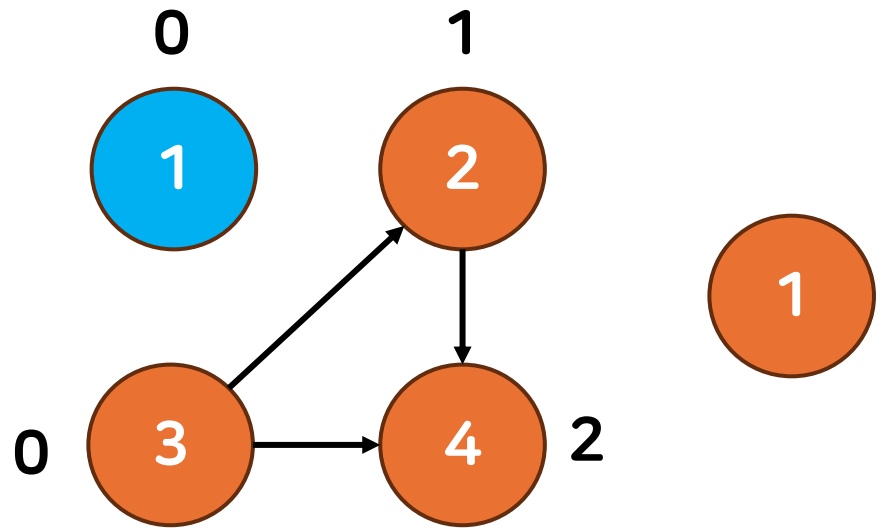
1. 모든 정점을 돌면서 indegree가 0인 값을 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

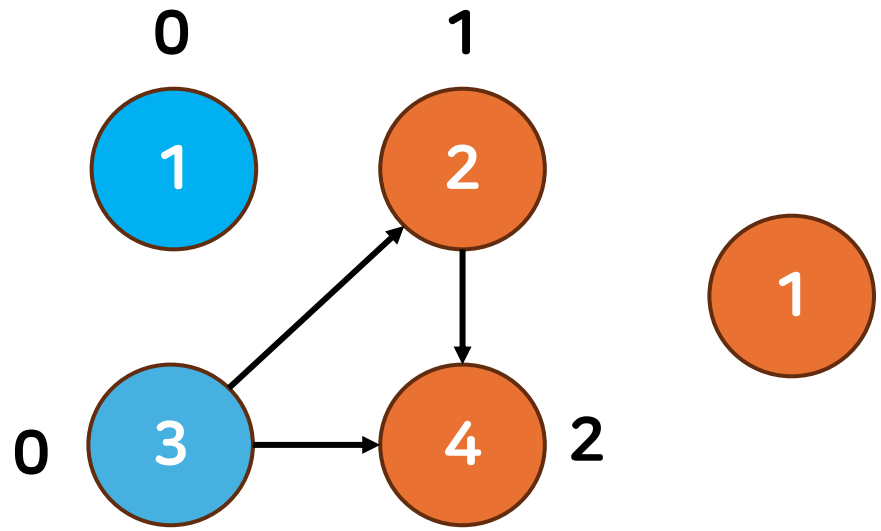
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

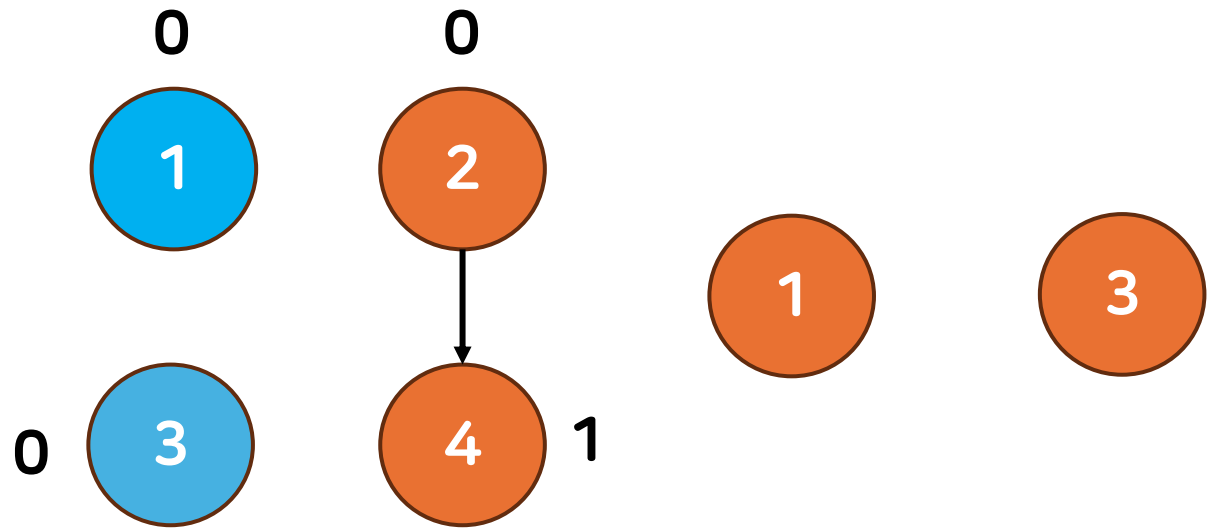
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

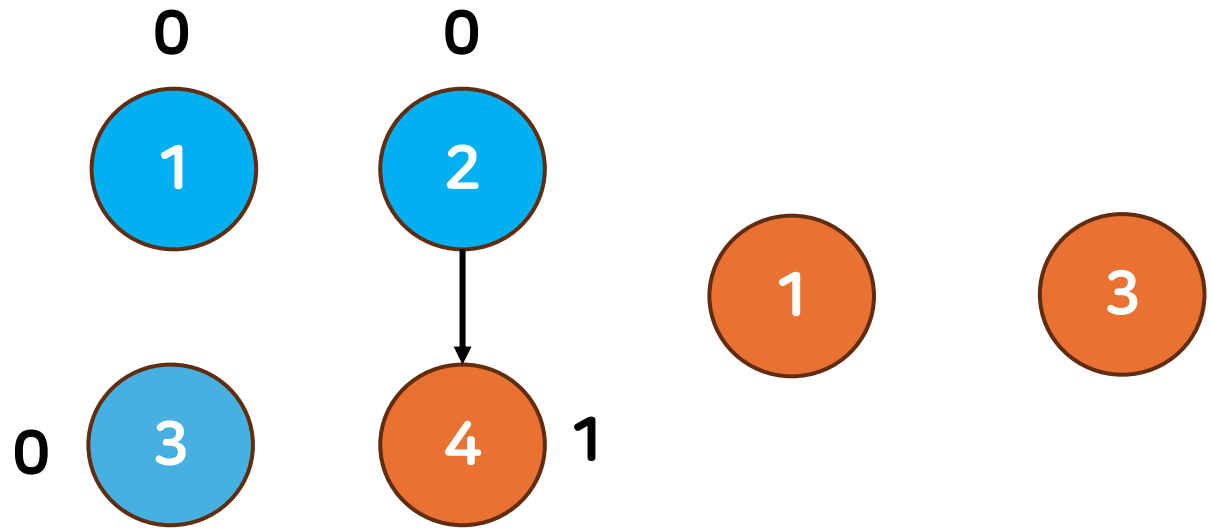
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

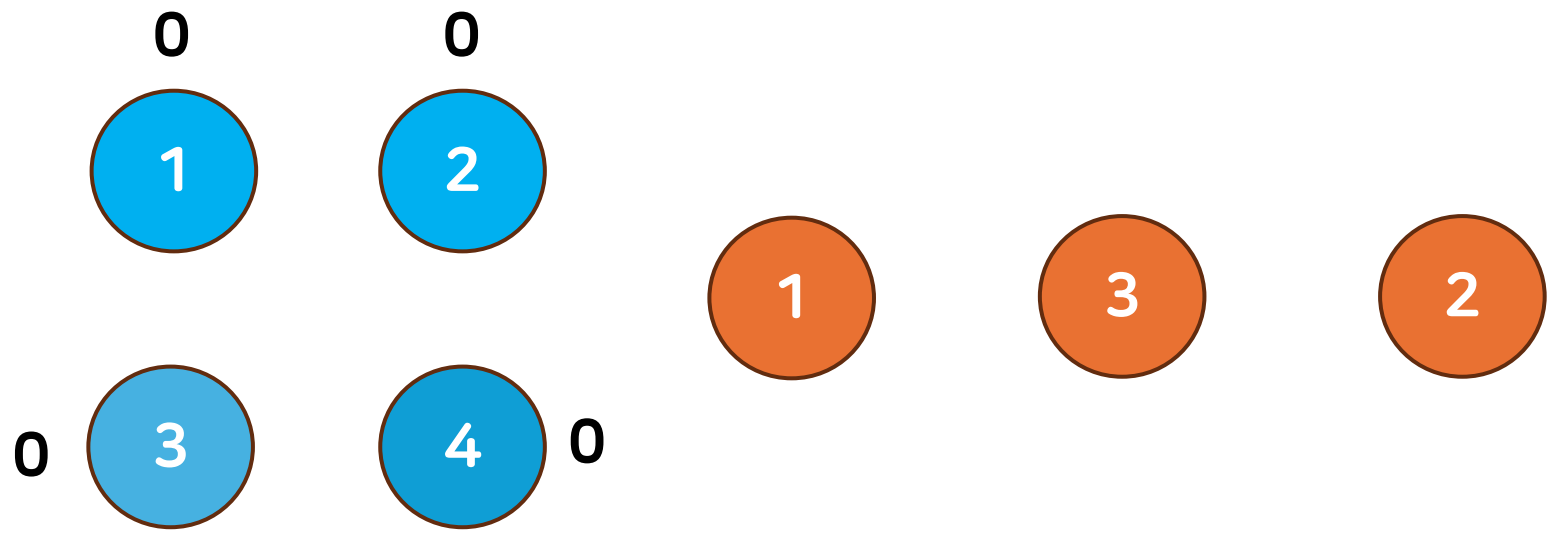
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

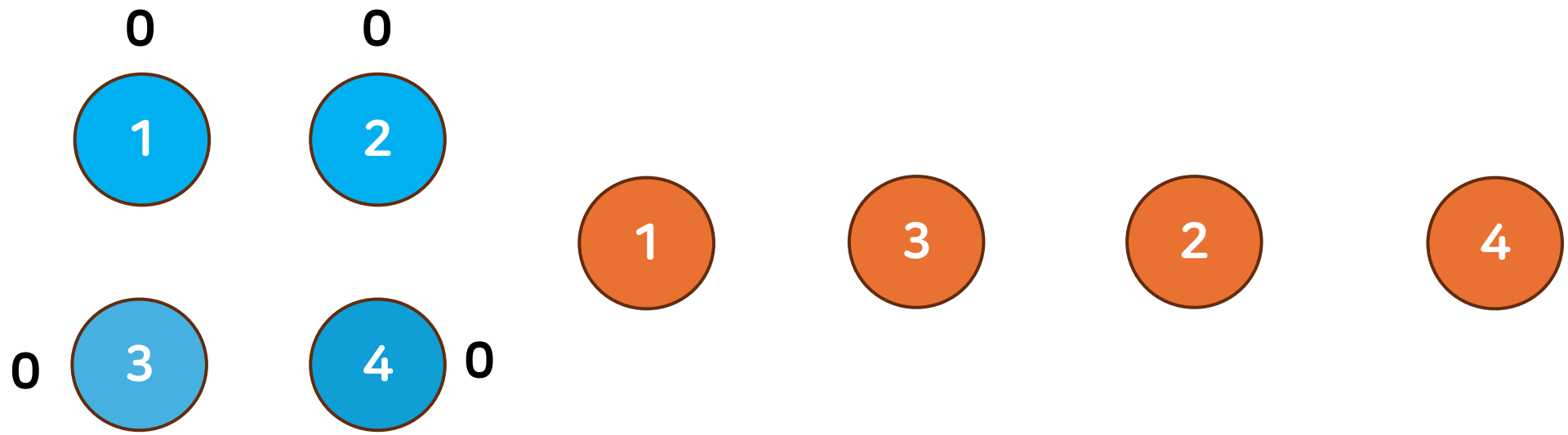
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

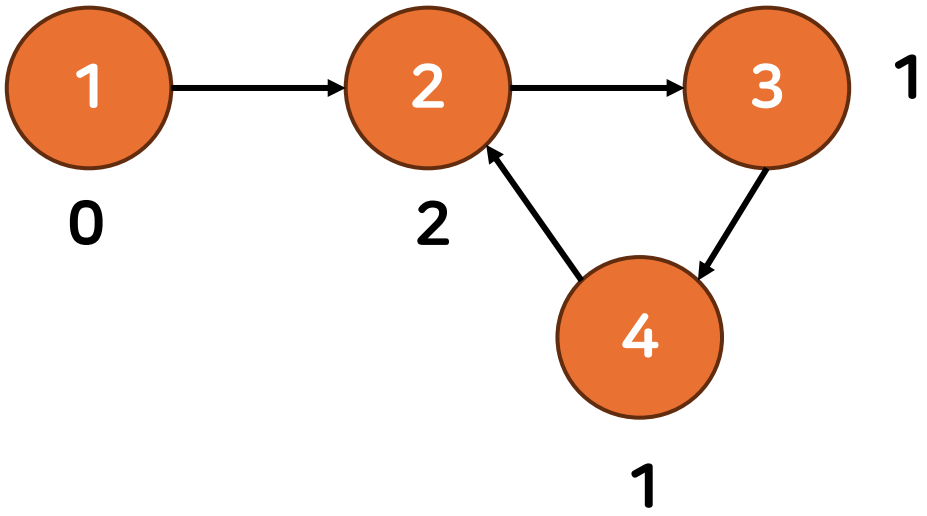
1. 모든 정점을 돌면서 indegree가 0인 값을 큐에 삽입

2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입

indegree가 0인 모든 값을 큐에 삽입
-> 방문 순서에 따라 위상 정렬의 값은 여러 개도 가능함

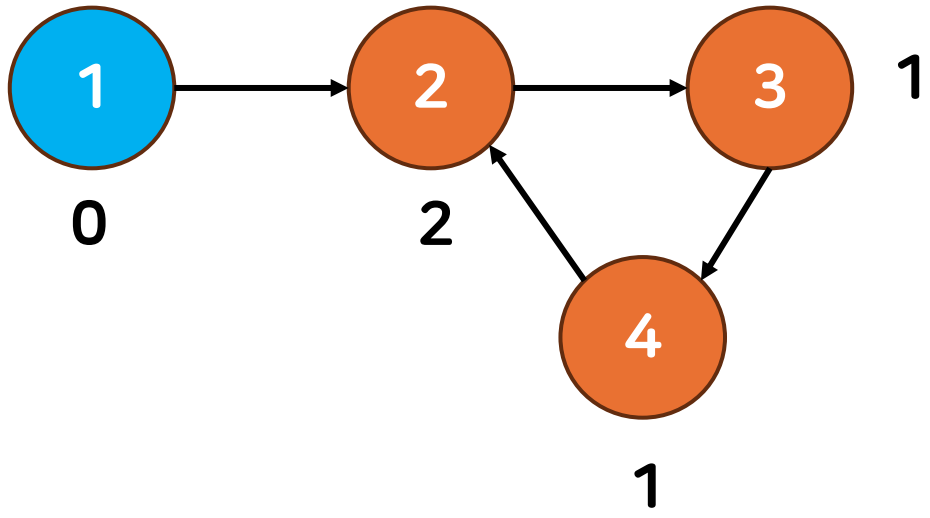
줄 세우기 / 2252

사이클이 존재 할 때 위상 정렬을 하면?



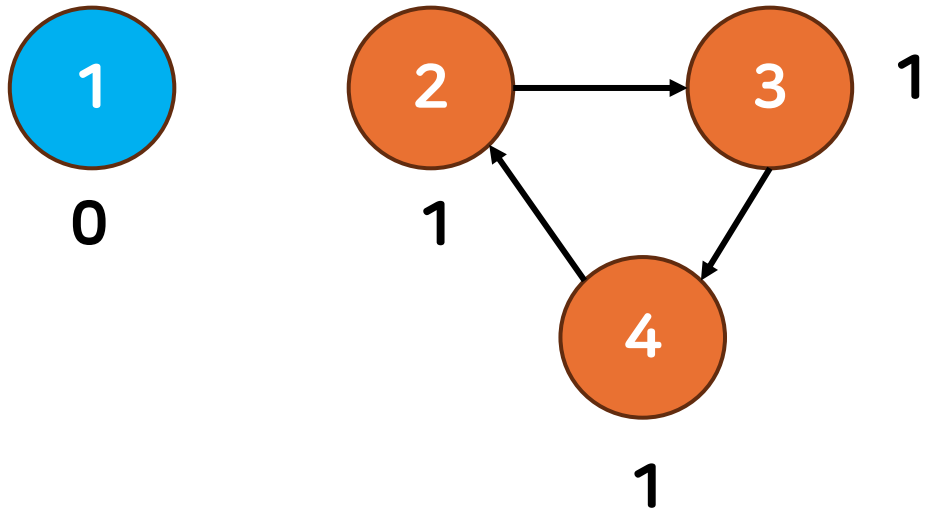
줄 세우기 / 2252

사이클이 존재 할 때 위상 정렬을 하면?



줄 세우기 / 2252

사이클이 존재 할 때 위상 정렬을 하면?

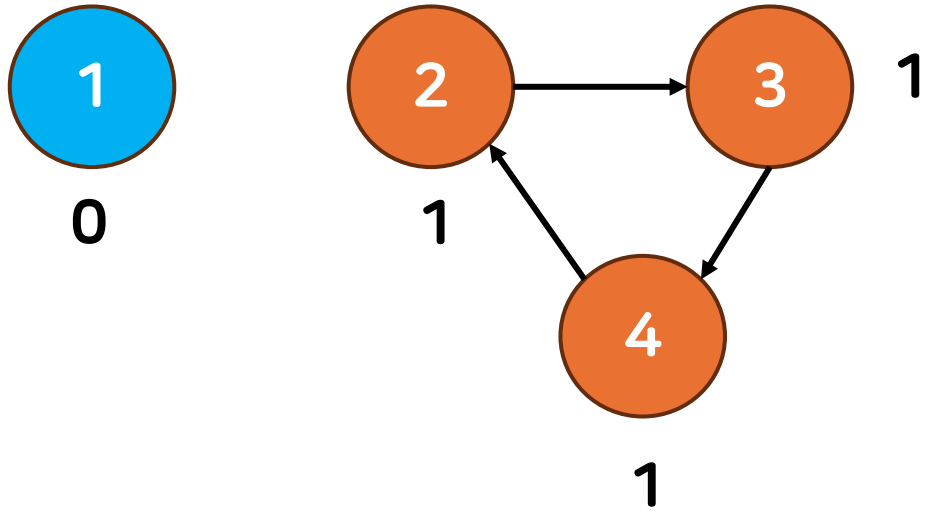


줄 세우기 / 2252

사이클이 존재 할 때 위상 정렬을 하면?

-> 어떤 indegree 값이 0이 아님

-> 위상 정렬로 사이클 여부를 판단 할 수 있음



줄 세우기 / 2252

C++

```cpp
const ll MAX = 50101;
const ll INF = 1e12;
ll n, m, ind[MAX];
vector <ll> adj[MAX];
queue <ll> q;

int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m;
    while(m--){
        ll s, e; cin >> s >> e;
        adj[s].push_back(e);
        ind[e]++;
    }

    for(int i = 1;i <= n;i++){
        // 들어오는 간선이 없으면 큐에 삽입
        if(!ind[i]) q.push(i);
    }

    while(!q.empty()){
        auto cur = q.front(); q.pop();
        // 현재 큐의 값 출력
        cout << cur << " ";
        for(auto& nxt : adj[cur]){
            // 다음 정점이 들어오는 간선이 없으면 큐에 삽입
            if(!--ind[nxt]) q.push(nxt);
        }
    }

    return 0;
}
```

줄 세우기 / 2252

Python

```python
MAX = 50101
INF = 10**12

n = m = 0
ind = [0] * MAX
adj = [[] for _ in range(MAX)]
q = deque()

n, m = map(int, input().split())

for _ in range(m):
    s, e = map(int, input().split())
    adj[s].append(e)
    ind[e] += 1

for i in range(1, n + 1):
    # 들어오는 간선이 없으면 큐에 삽입
    if ind[i] == 0:
        q.append(i)

while q:
    cur = q.popleft()
    # 현재 큐의 값 출력
    print(cur, end=' ')
    for nxt in adj[cur]:
        ind[nxt] -= 1
        # 다음 정점이 들어오는 간선이 없으면 큐에 삽입
        if ind[nxt] == 0:
            q.append(nxt)
```

질문?

문제집 / 1766

백준 1766 / https://www.acmicpc.net/problem/1766

민오는 1번부터 N번까지 총 N개의 문제로 되어 있는 문제집을 풀려고 한다. 문제는 난이도 순서로 출제되어 있다. 즉 1번 문제가 가장 쉬운 문제이고 N번 문제가 가장 어려운 문제가 된다.

어떤 문제부터 풀까 고민하면서 문제를 훑어보던 민오는, 몇몇 문제들 사이에는 '먼저 푸는 것이 좋은 문제'가 있다는 것을 알게 되었다. 예를 들어 1번 문제를 풀고 나면 4번 문제가 쉽게 풀린다거나 하는 식이다. 민오는 다음의 세 가지 조건에 따라 문제를 풀 순서를 정하기로 하였다.

1. N개의 문제는 모두 풀어야 한다.
2. 먼저 푸는 것이 좋은 문제가 있는 문제는, 먼저 푸는 것이 좋은 문제를 반드시 먼저 풀어야 한다.
3. 가능하면 쉬운 문제부터 풀어야 한다.

예를 들어서 네 개의 문제가 있다고 하자. 4번 문제는 2번 문제보다 먼저 푸는 것이 좋고, 3번 문제는 1번 문제보다 먼저 푸는 것이 좋다고 하자. 만일 4-3-2-1의 순서로 문제를 풀게 되면 조건 1과 조건 2를 만족한다. 하지만 조건 3을 만족하지 않는다. 4보다 3을 충분히 먼저 풀 수 있기 때문이다. 따라서 조건 3을 만족하는 문제를 풀 순서는 3-1-4-2가 된다.

문제의 개수와 먼저 푸는 것이 좋은 문제에 대한 정보가 주어졌을 때, 주어진 조건을 만족하면서 민오가 풀 문제의 순서를 결정해 주는 프로그램을 작성하시오.

입력

첫째 줄에 문제의 수 N(1 ≤ N ≤ 32,000)과 먼저 푸는 것이 좋은 문제에 대한 정보의 개수 M(1 ≤ M ≤ 100,000)이 주어진다. 둘째 줄부터 M개의 줄에 걸쳐 두 정수의 순서쌍 A,B가 빈칸을 사이에 두고 주어진다. 이는 A번 문제는 B번 문제보다 먼저 푸는 것이 좋다는 의미이다.

항상 문제를 모두 풀 수 있는 경우만 입력으로 주어진다.

출력

첫째 줄에 문제 번호를 나타내는 1 이상 N 이하의 정수들을 민오가 풀어야 하는 순서대로 빈칸을 사이에 두고 출력한다.

문제집 / 1766

아래에 예제에서는
4 -> 2
3 -> 1 순서로 문제를 풀어야 함

예제 입력 1 복사

```
4 2
4 2
3 1
```

예제 출력 1 복사

```
3 1 4 2
```

문제집 / 1766

1. N개의 문제는 모두 풀어야 한다.
2. 먼저 푸는 것이 좋은 문제가 있는 문제는, 먼저 푸는 것이 좋은 문제를 반드시 먼저 풀어야 한다.
3. 가능하면 쉬운 문제부터 풀어야 한다.

먼저 푸는 것이 좋은 문제를 먼저 풀어야 하는 것은 위상 정렬로 해결 할 수 있음

예제 입력 1 복사

```
4 2
4 2
3 1
```

예제 출력 1 복사

```
3 1 4 2
```

문제집 / 1766

1. N개의 문제는 모두 풀어야 한다.
2. 먼저 푸는 것이 좋은 문제가 있는 문제는, 먼저 푸는 것이 좋은 문제를 반드시 먼저 풀어야 한다.
3. 가능하면 쉬운 문제부터 풀어야 한다.

3번의 조건이 없다면

4 3 1 2

4 2 3 1

3 4 2 1

3 4 1 2

예제 입력 1 복사

```
4 2
4 2
3 1
```

예제 출력 1 복사

```
3 1 4 2
```

문제집 / 1766

가능하면 먼저 쉬운 문제를 풀어야 함

처음에 indegree가 0인 값을 큐에 넣으면 3, 4가 들어 감
3, 4에는 우선순위가 없기 때문에 둘 중 아무 값이 먼저 와도 괜찮음

하지만 이 문제에서는 작은 숫자가 먼저 와야 함

예제 입력 1 복사

```
4 2
4 2
3 1
```

예제 출력 1 복사

```
3 1 4 2
```

문제집 / 1766

하지만 이 문제에서는 작은 숫자가 먼저 와야 함
-> 큐에 값이 여러 개 있으면 가장 작은 값을 출력해야 함
우선순위 큐

문제집 / 1766

C++

```cpp
const ll MAX = 40101;
const ll INF = 1e12;
ll n, m, ind[MAX];
vector <ll> adj[MAX];
priority_queue <ll, vector<ll>, greater<ll>> pq;
```

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m;
    while(m--){
        ll s, e; cin >> s >> e;
        adj[s].push_back(e);
        ind[e]++;
    }

    // 우선순위 큐 사용
    for(int i = 1;i <= n;i++){
        if(!ind[i]) pq.push(i);
    }

    while(!pq.empty()){
        ll cur = pq.top(); pq.pop();
        cout << cur << " ";
        for(auto& nxt : adj[cur]){
            if(!--ind[nxt]) pq.push(nxt);
        }
    }

    return 0;
}
```

문제집 / 1766

Python

```python
MAX = 40101
INF = 10**12

n, m = map(int, input().split())
ind = [0] * MAX
adj = [[] for _ in range(MAX)]
pq = []
```

```python
for _ in range(m):
    s, e = map(int, input().split())
    adj[s].append(e)
    ind[e] += 1

# 우선순위 큐 사용
for i in range(1, n + 1):
    if ind[i] == 0:
        heapq.heappush(pq, i)

while pq:
    cur = heapq.heappop(pq)
    print(cur, end=' ')
    for nxt in adj[cur]:
        ind[nxt] -= 1
        if ind[nxt] == 0:
            heapq.heappush(pq, nxt)
```

질문?

기본 과제

줄 세우기 – https://www.acmicpc.net/problem/2252

음악 프로그램 – https://www.acmicpc.net/problem/2623

문제집 - https://www.acmicpc.net/problem/1766

고생하셨습니다