25-1 이니로
알고리즘 멘토링

멘토 - 김수성

해킹 / 10282

백준 10282 / https://www.acmicpc.net/problem/10282

문제

최흉최악의 해커 yum3이 네트워크 시설의 한 컴퓨터를 해킹했다! 이제 서로에 의존하는 컴퓨터들은 점차 하나둘 전염되기 시작한다. 어떤 컴퓨터 a가 다른 컴퓨터 b에 의존한다면, b가 감염되면 그로부터 일정 시간 뒤 a도 감염되고 만다. 이때 b가 a를 의존하지 않는다면, a가 감염되더라도 b는 안전하다.

최흉최악의 해커 yum3이 해킹한 컴퓨터 번호와 각 의존성이 주어질 때, 해킹당한 컴퓨터까지 포함하여 총 몇 대의 컴퓨터가 감염되며 그에 걸리는 시간이 얼마인지 구하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수가 주어진다. 테스트 케이스의 개수는 최대 100개이다. 각 테스트 케이스는 다음과 같이 이루어져 있다.

- 첫째 줄에 컴퓨터 개수 n, 의존성 개수 d, 해킹당한 컴퓨터의 번호 c가 주어진다(1 ≤ n ≤ 10,000, 1 ≤ d ≤ 100,000, 1 ≤ c ≤ n).
- 이어서 d개의 줄에 각 의존성을 나타내는 정수 a, b, s가 주어진다(1 ≤ a, b ≤ n, a ≠ b, 0 ≤ s ≤ 1,000). 이는 컴퓨터 a가 컴퓨터 b를 의존하며, 컴퓨터 b가 감염되면 s초 후 컴퓨터 a도 감염됨을 뜻한다.

각 테스트 케이스에서 같은 의존성 (a, b)가 두 번 이상 존재하지 않는다.

출력

각 테스트 케이스마다 한 줄에 걸쳐 총 감염되는 컴퓨터 수, 마지막 컴퓨터가 감염되기까지 걸리는 시간을 공백으로 구분지어 출력한다.

해킹 / 10282

첫 번째 테스트 케이스
2 -> 3 (5)
두 번째 테스트 케이스
1 -> 2 -> 3 (6)

예제 입력 1 복사

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

예제 출력 1 복사

```
2 5
3 6
```

해킹 / 10282

두 번째 테스트 케이스
1 -> 2 -> 3 (6)

컴퓨터 b가 바이러스에 걸리면 컴퓨터 a는
c의 시간 뒤에 바이러스에 걸림

각 컴퓨터를 정점으로 보면
b -> a로 가는 간선의 비용이 c로 정의 가능

해킹 / 10282

각 컴퓨터를 정점으로 보면
b -> a로 가는 간선의 비용이 c로 정의 가능

문제에서 주어진 의존성을 그래프로 변형

마지막 컴퓨터가 감염되기까지 걸리는 시간
-> 최단 거리의 최댓값

총 감염되는 컴퓨터의 수
-> 거리가 INF가 아닌 컴퓨터의 수

해킹 / 10282

C++

```cpp
const ll MAX = 10101;
const ll INF = 1e12;
ll n, m, k, d[MAX];
vector <pair<ll, ll>> adj[MAX];

using pll = pair<ll, ll>;
priority_queue <pll, vector<pll>, greater<pll>> pq;

void init(){
    while(!pq.empty()) pq.pop();
    for(int i = 1;i <= n;i++) adj[i].clear();
}
```

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

    ll t; cin >> t;
    while(t--) run();

    return 0;
}
```

```cpp
void run(){
    cin >> n >> m >> k; init();
    while(m--){
        ll s, e, c; cin >> s >> e >> c;
        adj[e].push_back({s, c});
    }

    for(int i = 0;i < MAX;i++) d[i] = INF;
    pq.push({0, k});

    while(!pq.empty()){
        auto[cd, cur] = pq.top(); pq.pop();
        if(d[cur] <= cd) continue;
        d[cur] = cd;

        for(auto& [nxt, co] : adj[cur]){
            if(d[nxt] <= cd + co) continue;
            pq.push({cd + co, nxt});
        }
    }

    ll cnt = 0, result = 0;
    for(int i = 1;i <= n;i++){
        // 바이러스에 감염되지 않았으면 건너 뜀
        if(d[i] == INF) continue;

        // 개수 증가 및 최댓값 갱신
        cnt++; result = max(result, d[i]);
    }

    cout << cnt << " " << result << "\n";
}
```

해킹 / 10282

Python

```python
def init():
    global pq, adj
    pq.clear()
    for i in range(1, n + 1):
        adj[i].clear()
```

```python
t = int(sys.stdin.readline())
d = [INF] * MAX
adj = [[] for _ in range(MAX)]
pq = []
for _ in range(t):
    run()
```

```python
def run():
    global n, m, k, d, adj, pq
    n, m, k = map(int, sys.stdin.readline().split())
    init()
    for _ in range(m):
        s, e, c = map(int, sys.stdin.readline().split())
        adj[e].append((s, c))

    for i in range(MAX):
        d[i] = INF
    heapq.heappush(pq, (0, k))

    while pq:
        cd, cur = heapq.heappop(pq)
        if d[cur] <= cd:
            continue
        d[cur] = cd

        for nxt, co in adj[cur]:
            if d[nxt] <= cd + co:
                continue
            heapq.heappush(pq, (cd + co, nxt))

    cnt = 0
    result = 0
    for i in range(1, n + 1):
        # 바이러스에 감염되지 않았으면 건너 뜀
        if d[i] == INF:
            continue

        # 개수 증가 및 최댓값 갱신
        cnt += 1
        result = max(result, d[i])
    print(cnt, result)
```

질문?

소가 길을 건너간 이유 7 / 14461

백준 14461 / https://www.acmicpc.net/problem/14461

문제

소가 길을 건너는 이유는 그냥 길이 많아서이다. 존의 농장에는 길이 너무 많아서, 길을 건너지 않고서는 별로 돌아다닐 수가 없다.

존의 농장에는 작은 정사각형 목초지가 N×N (3 ≤ N ≤ 100) 격자로 이루어져 있다. 농장의 바깥에는 높은 울타리가 있어서 소가 농장 밖으로 나갈 일은 없다. 이 농장에 사는 소 베시는 한 목초지에서 상하좌우로 인접한 다른 목초지로 이동할 수 있지만, 교통사고를 피하기 위해 차가 안 오는지 확인하고 길을 건너야 한다. 길을 건너는데는 T초 (0 ≤ T ≤ 1,000,000)가 걸린다.

존이 베시에게 체스 대결을 신청했다. 베시는 북서쪽 끝에 있는 목초지에서 남동쪽 끝에 있는 존의 집으로 가야 한다. 길이 멀기 때문에 베시는 가는 도중에 배가 고파진다. 그래서 길을 세 번 건널 때마다 목초지에 있는 풀을 먹어야 한다. 존의 집에 도착할 때도 해당되지만, 출발할 때는 해당되지 않는다. 목초지마다 풀이 자란 정도가 달라서, 풀을 먹는데 걸리는 시간도 다르다.

베시가 가능한 한 빨리 존의 집에 도착할 수 있도록 도와주자.

입력

첫 줄에 N과 T가 주어진다. 다음 N줄에는 목초지마다 풀을 먹는데 걸리는 시간이 N×N의 형태로 주어진다. 각각의 수는 모두 100,000 이하이다.

출력

베시가 존의 집까지 가는데 걸리는 최소 시간을 출력한다.

소가 길을 건너간 이유 7 / 14461

(0, 0)에서 (N – 1, N -1)까지 가는 최단 거리를 구해야 함
각 간선은 M의 비용이 들고, 3번의 이동마다
A[i][j]의 비용이 추가로 들음

(0,0) -> (0, 1) -> (0, 2) -> (0, 3) -> (1, 3) -> (2, 3) -> (2, 2)
-> (2, 3) -> (3, 3) = 2 * 8 + 10 + 5 = 31

예제 입력 1 복사

```
4 2
30 92 36 10
38 85 60 16
41 13 5 68
20 97 13 80
```

예제 출력 1 복사

```
31
```

소가 길을 건너간 이유 7 / 14461

최단 거리를 구해야 함 -> 다익스트라

하지만 각 간선을 단순히 M의 비용으로 연결하면
세 번째 이동의 추가 비용을 계산 할 수 없음

현재 몇 번째 이동인지 알아야 함
-> 추가적인 정보가 필요함
-> DP에서의 방법

소가 길을 건너간 이유 7 / 14461

현재 몇 번째 이동인지 알아야 함
-> 추가적인 정보가 필요함
-> DP에서의 방법

D[i][j] = (i, j)에서의 최단 거리
-> D[i][j][k] = (i, j)에서 k 번째 이동일 때의 최단 거리

소가 길을 건너간 이유 7 / 14461

D[i][j][k] = (i, j)에서 k 번째 이동일 때의 최단 거리

D[i][j][1] = D[py][px][0] + M
D[i][j][2] = D[py][px][1] + M
D[i][j][0] = D[py][px][2] + A[i][j]

소가 길을 건너간 이유 7 / 14461

C++

```cpp
const ll MAX = 3 * 10201;
const ll INF = 1e12;
ll n, m, d[MAX];
vector <pair<ll, ll>> adj[MAX];
ll a[101][101];
ll dx[4] = {0, 0, 1, -1}, dy[4] = {1, -1, 0, 0};

using pll = pair<ll, ll>;
priority_queue <pll, vector<pll>, greater<pll>> pq;

bool outrange(ll cy, ll cx){
    return cy <= 0 || cx <= 0 || cy > n || cx > n;
}

ll num(ll cy, ll cx, ll cnt){
    return cnt * 10101 + cy * 101 + cx;
}
```

```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++){
        for(int j = 1;j <= n;j++) cin >> a[i][j];
    }

    for(int i = 1;i <= n;i++){
        for(int j = 1;j <= n;j++){
            for(int k = 0;k < 4;k++){
                ll ny = i + dy[k], nx = j + dx[k];
                if(outrange(ny, nx)) continue;
                adj[num(i, j, 0)].push_back({num(ny, nx, 1), m});
                adj[num(i, j, 1)].push_back({num(ny, nx, 2), m});
                adj[num(i, j, 2)].push_back({num(ny, nx, 0), m + a[ny][nx]});
            }
        }
    }

    for(int i = 0;i < MAX;i++) d[i] = INF;
    pq.push({0, num(1, 1, 0)});

    while(!pq.empty()){
        auto[cd, cur] = pq.top(); pq.pop();
        if(d[cur] <= cd) continue;
        d[cur] = cd;

        for(auto& [nxt, co] : adj[cur]){
            if(d[nxt] <= cd + co) continue;
            pq.push({cd + co, nxt});
        }
    }

    ll result = INF;
    for(int i = 0;i < 3;i++) result = min(result, d[num(n, n, i)]);
    cout << result;

    return 0;
}
```

소가 길을 건너간 이유 7 / 14461

Python

```python
INF = 10**12
MAX = 3 * 10201

def outrange(cy, cx):
    return cy <= 0 or cx <= 0 or cy > n or cx > n

def num(cy, cx, cnt):
    return cnt * 10101 + cy * 101 + cx

n, m = map(int, input().split())
a = [[0] * (n + 2) for _ in range(n + 2)]
for i in range(1, n + 1):
    row = list(map(int, input().split()))
    for j in range(1, n + 1):
        a[i][j] = row[j - 1]

adj = [[] for _ in range(MAX)]
d = [INF] * MAX
dx = [0, 0, 1, -1]
dy = [1, -1, 0, 0]
```

```python
for i in range(1, n + 1):
    for j in range(1, n + 1):
        for k in range(4):
            ny = i + dy[k]
            nx = j + dx[k]
            if outrange(ny, nx):
                continue
            adj[num(i, j, 0)].append((num(ny, nx, 1), m))
            adj[num(i, j, 1)].append((num(ny, nx, 2), m))
            adj[num(i, j, 2)].append((num(ny, nx, 0), m + a[ny][nx]))

pq = []
heapq.heappush(pq, (0, num(1, 1, 0)))

while pq:
    cd, cur = heapq.heappop(pq)
    if d[cur] <= cd:
        continue
    d[cur] = cd
    for nxt, co in adj[cur]:
        if d[nxt] <= cd + co:
            continue
        heapq.heappush(pq, (cd + co, nxt))

result = INF
for i in range(3):
    result = min(result, d[num(n, n, i)])
print(result)
```

질문?

수열과 개구리 / 32294

백준 32294 / https://www.acmicpc.net/problem/32294

문제

길이가 n이고 양의 정수로 구성된 수열 a와 b가 있습니다. 두 수열의 인덱스는 1부터 시작합니다.

이 수열 위에 개구리 한 마리가 있습니다. 개구리는 초기에 어떤 정수 위치 $1 \le x \le n$에서 출발하며, 자신의 위치가 수열 밖[†]이 될 때까지 다음을 반복합니다.

b_x초 동안 기다린 뒤, 위치 $x - a_x$로 이동하거나 위치 $x + a_x$로 이동합니다. 이때 이동한 위치가 x의 새로운 값이 됩니다.

개구리가 시작하는 위치 x에 대해, 개구리의 위치가 수열 밖이 될 수 있는 최초의 시각을 $f(x)$초라고 정의할 때, 여러분은 $f(1), f(2), \cdots, f(n)$의 값을 모두 구해야 합니다.

† 어떤 위치 x에 대해서, $1 \le x \le n$이면 **수열 안**, $x < 1$ 또는 $x > n$이면 **수열 밖**이라고 부릅니다.

입력

첫 번째 줄에 두 수열의 길이 n이 주어집니다. ($1 \le n \le 2 \cdot 10^5$)

두 번째 줄에 n개의 정수 $a_1, a_2, \ldots, a_n$이 공백으로 구분되어 주어집니다. ($1 \le a_i \le n$)

세 번째 줄에 n개의 정수 $b_1, b_2, \ldots, b_n$이 공백으로 구분되어 주어집니다. ($1 \le b_i \le 10^6$)

출력

한 줄에 $f(1), f(2), \cdots, f(n)$의 값을 공백으로 구분하여 순서대로 출력합니다.

문제의 제한에 따라 모든 $1 \le i \le n$에 대해 $f(i)$가 유한함을 증명할 수 있습니다.

수열과 개구리 / 32294

각 인덱스 i에 대해서 i - A[i] 또는 i + A[i]로 이동 할 때
B[i]의 시간이 들음

각 인덱스 i에 대해서 0 이하 또는 N 초과로 이동 할 때
최단거리 출력

예제 입력 1 복사

```
5
3 4 2 5 1
2 5 1 4 3
```

예제 출력 1 복사

```
2 5 3 4 3
```

수열과 개구리 / 32294

각 인덱스 i에 대해서 i - A[i] 또는 i + A[i]로 이동 할 때
B[i]의 시간이 들음

각 인덱스 i는 i + A[i] 또는 i - A[i]와 연결되어 있고
비용이 B[i]인 간선을 가지는 그래프로 모델링 할 수 있음

수열과 개구리 / 32294

각 인덱스 i에 대해서 다익스트라를 돌리고
x <= 0 || x > n 인덱스 x의 최단거리를 구하면 됨
B[i]가 1e6까지 들어오기 때문에 모든 x를 시도 할 수는 없음

x <= 0 || x > n 일 때 간선을 0과 연결하면 d[0]의 값이 정답이 됨
-> O(N ^ 2 log N) N이 2e5이므로 시간 초과

수열과 개구리 / 32294

각 인덱스 i에 대해서 i -> 0을 구해야 함
저번주에 풀었던 파티 문제를 생각해보자

각 인덱스 i에 대해서 i -> 0를 구하는 건 시간이 많이 들음

모든 간선을 뒤집고 0 -> i 를 구하면 다익스트라를
한번만 사용해서 구할 수 있음

수열과 개구리 / 32294

C++

```cpp
const ll MAX = 201010;
const ll INF = 1e12;
ll n, m, d[MAX];
vector <pair<ll, ll>> adj[MAX];
ll a[MAX], b[MAX];

using pll = pair<ll, ll>;
priority_queue <pll, vector<pll>, greater<pll>> pq;
```

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

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

    for(int i = 1;i <= n;i++){
        ll nx = i - a[i];
        if(nx <= 0) adj[0].push_back({i, b[i]});
        else adj[nx].push_back({i, b[i]});

        nx = i + a[i];
        if(nx > n) adj[0].push_back({i, b[i]});
        else adj[nx].push_back({i, b[i]});
    }


    for(int i = 0;i < MAX;i++) d[i] = INF;
    pq.push({0, 0});

    while(!pq.empty()){
        auto[cd, cur] = pq.top(); pq.pop();
        if(d[cur] <= cd) continue;
        d[cur] = cd;

        for(auto& [nxt, co] : adj[cur]){
            if(d[nxt] <= cd + co) continue;
            pq.push({cd + co, nxt});
        }
    }

    for(int i = 1;i <= n;i++) cout << d[i] << " ";

    return 0;
}
```

수열과 개구리 / 32294

Python

```python
MAX = 201010
INF = 10**12

n = int(input())
a = [0] + list(map(int, input().split()))
b = [0] + list(map(int, input().split()))

adj = [[] for _ in range(MAX)]
d = [INF] * MAX
pq = []

for i in range(1, n + 1):
    nx = i - a[i]
    if nx <= 0:
        adj[0].append((i, b[i]))
    else:
        adj[nx].append((i, b[i]))

    nx = i + a[i]
    if nx > n:
        adj[0].append((i, b[i]))
    else:
        adj[nx].append((i, b[i]))
```

```python
heapq.heappush(pq, (0, 0))

while pq:
    cd, cur = heapq.heappop(pq)
    if d[cur] <= cd:
        continue
    d[cur] = cd

    for nxt, co in adj[cur]:
        if d[nxt] <= cd + co:
            continue
        heapq.heappush(pq, (cd + co, nxt))

print(' '.join(str(d[i]) for i in range(1, n + 1)))
```

질문?

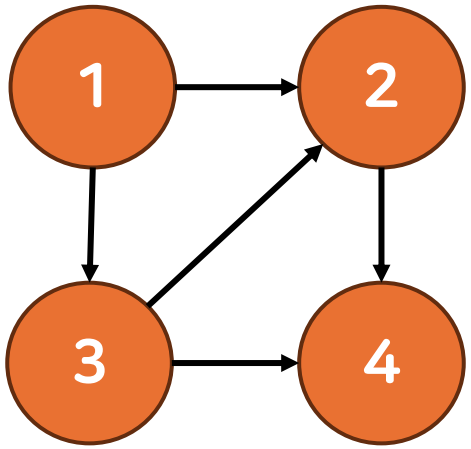
8주차 – 위상 정렬

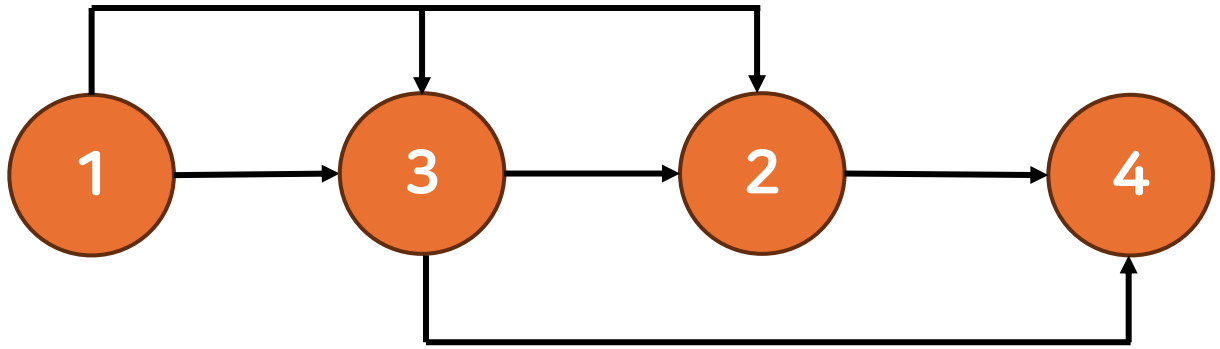
위상 정렬

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
```

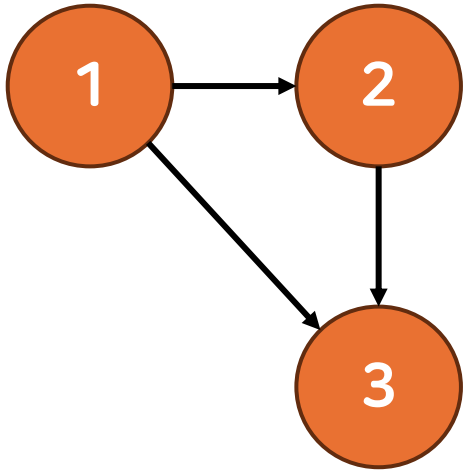
예제 입력 2 복사

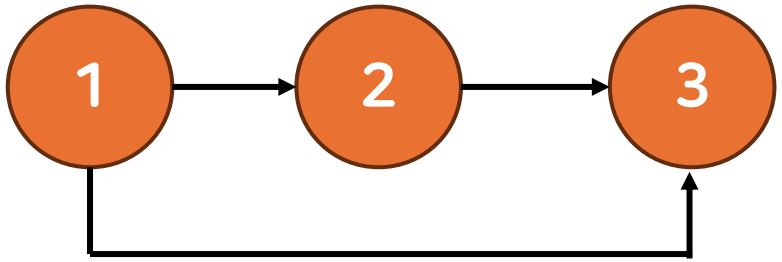
```
4 2
4 2
3 1
```

예제 출력 2 복사

```
4 2 3 1
```

줄 세우기 / 2252





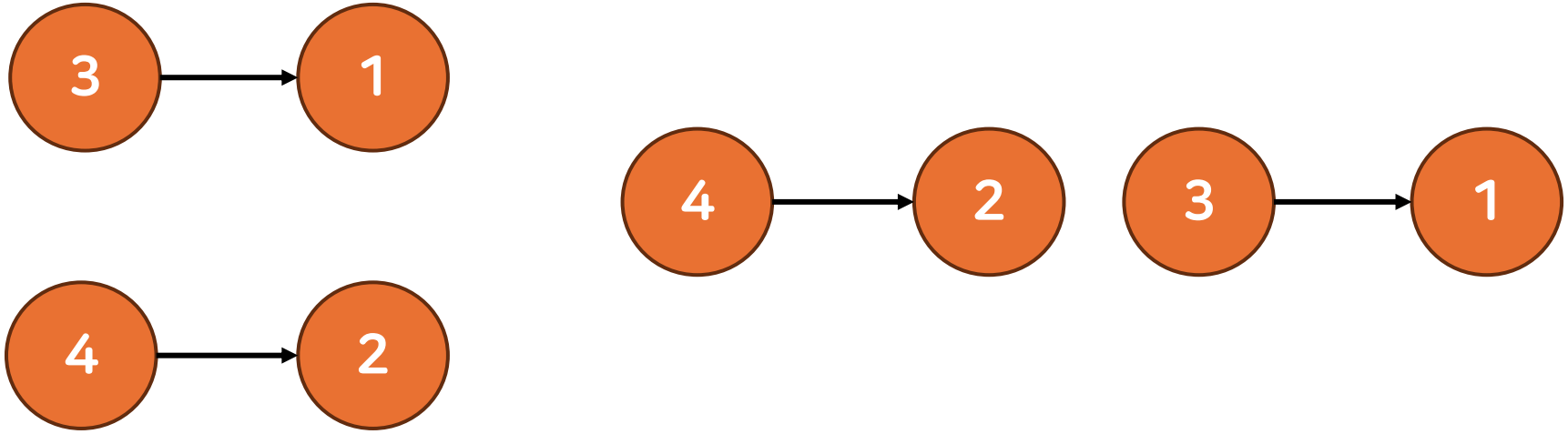
예제 입력 1 복사

```
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
```

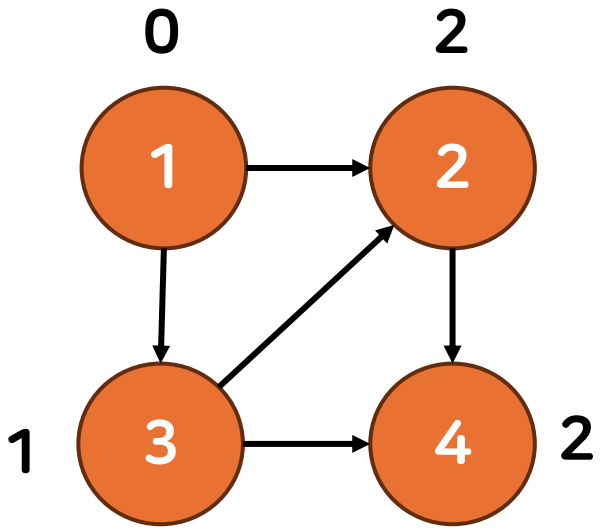
예제 출력 2 복사

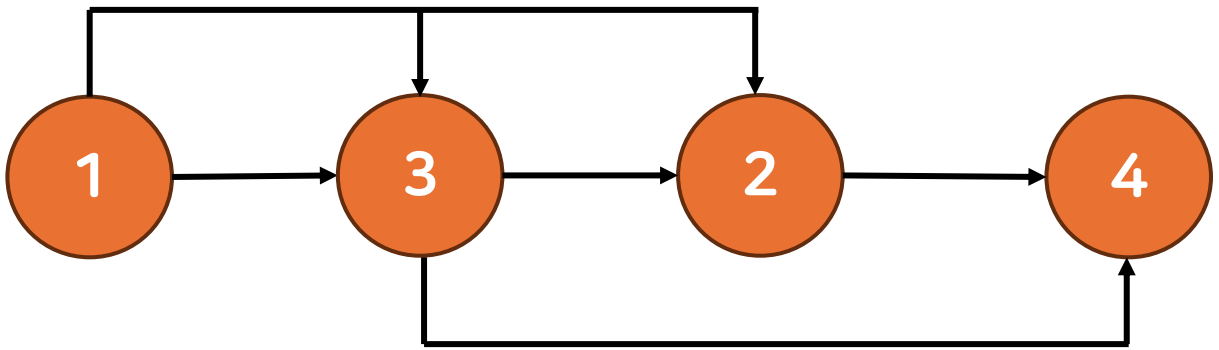
```
4 2 3 1
```

줄 세우기 / 2252

각 정점에 들어오는 간선의 수 -> indgree

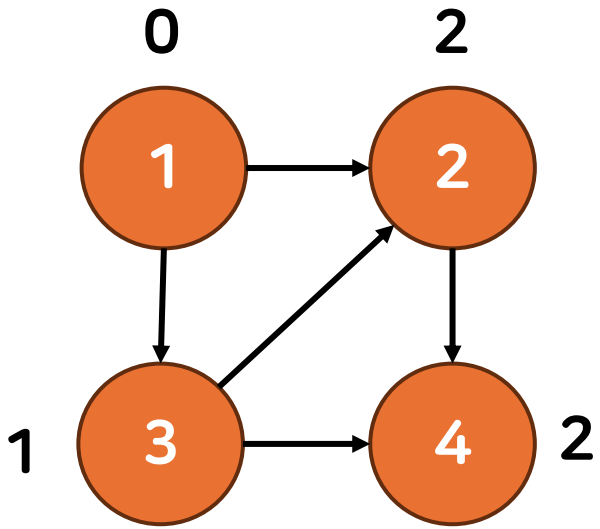
indgree가 0이 아니면 자신 이전의 값이 존재 함

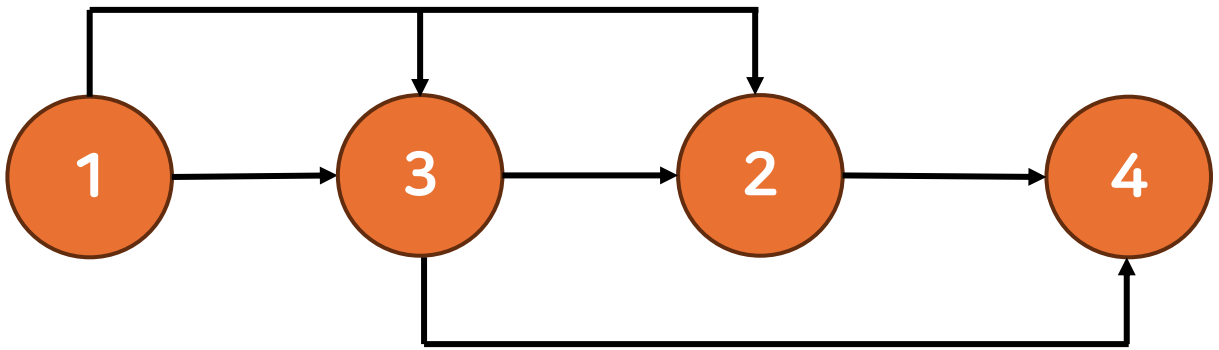




줄 세우기 / 2252

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

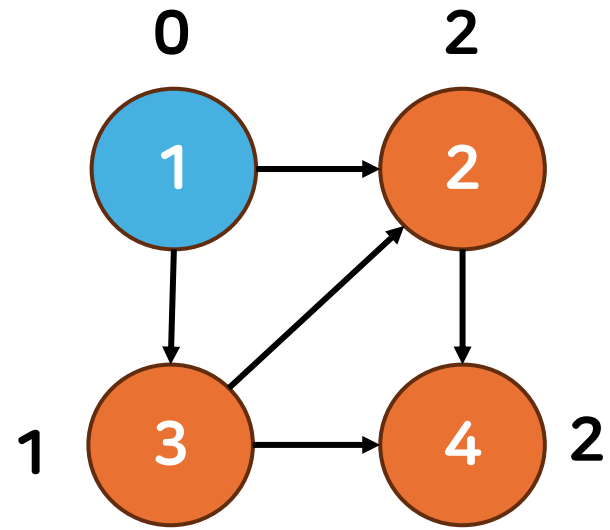




줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

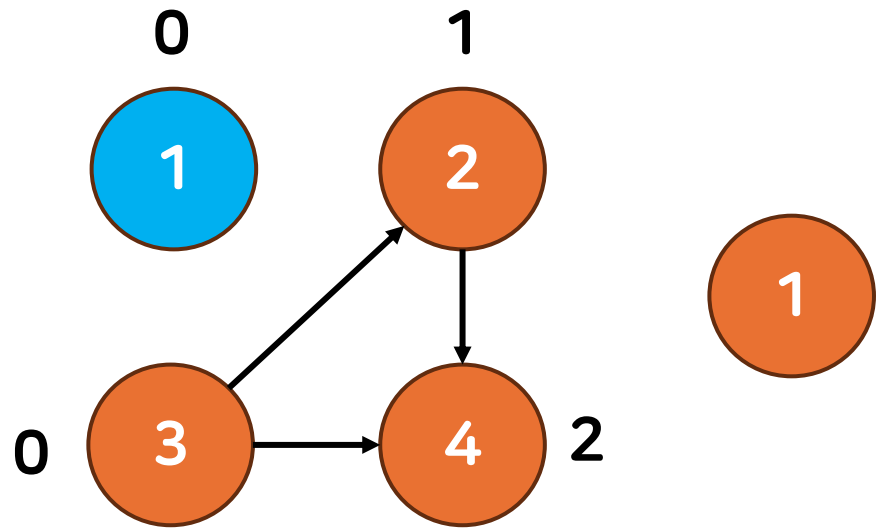
1. 모든 정점을 돌면서 indegree가 0인 값을 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

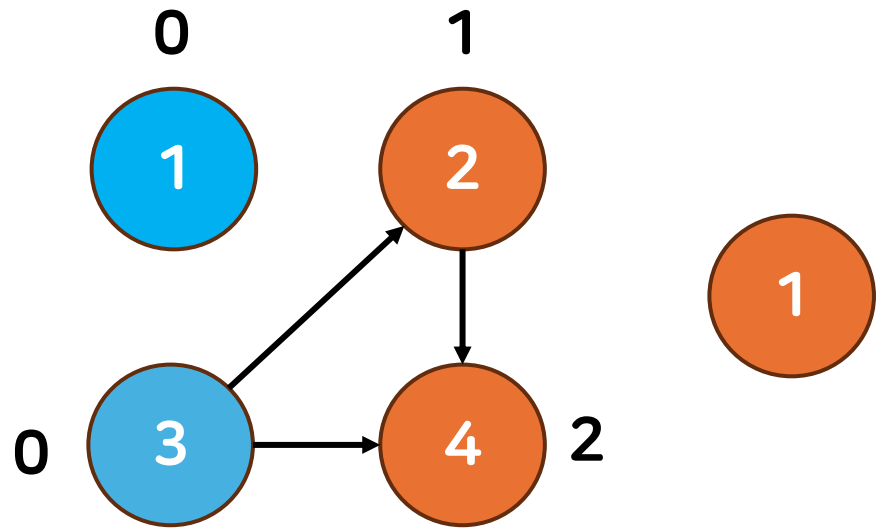
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

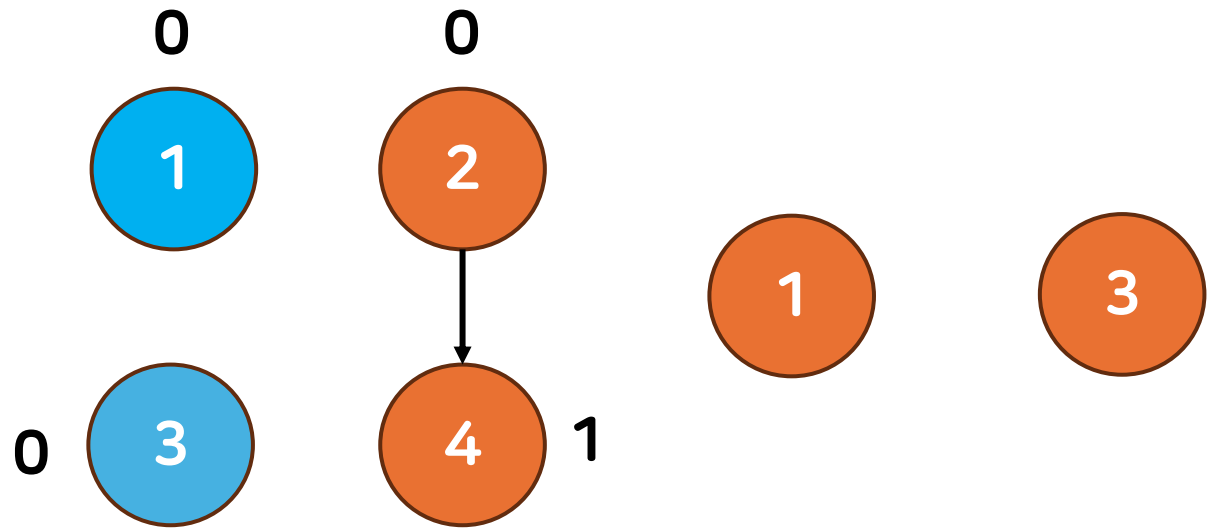
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

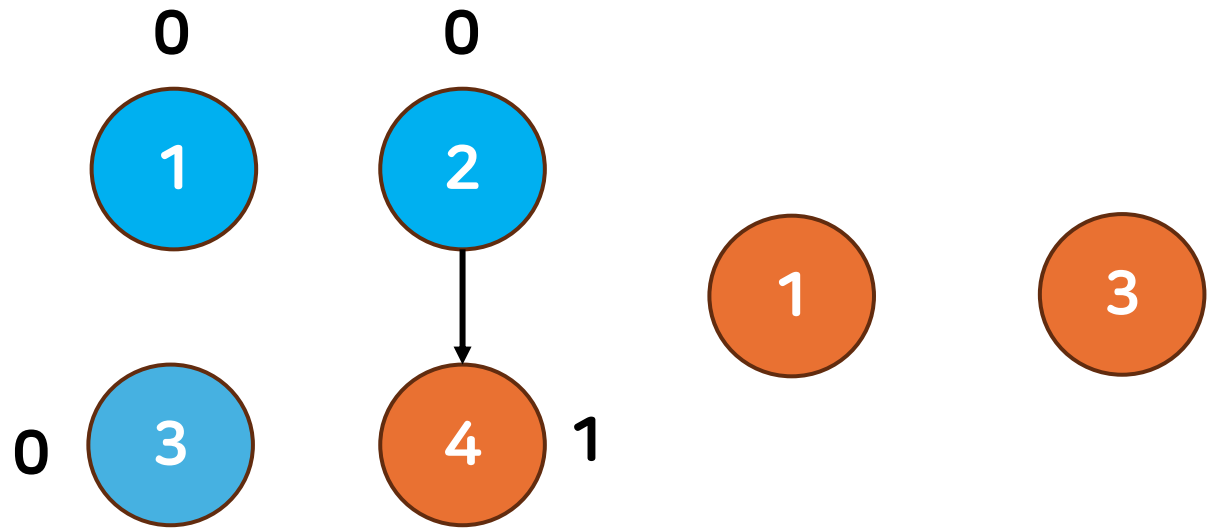
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

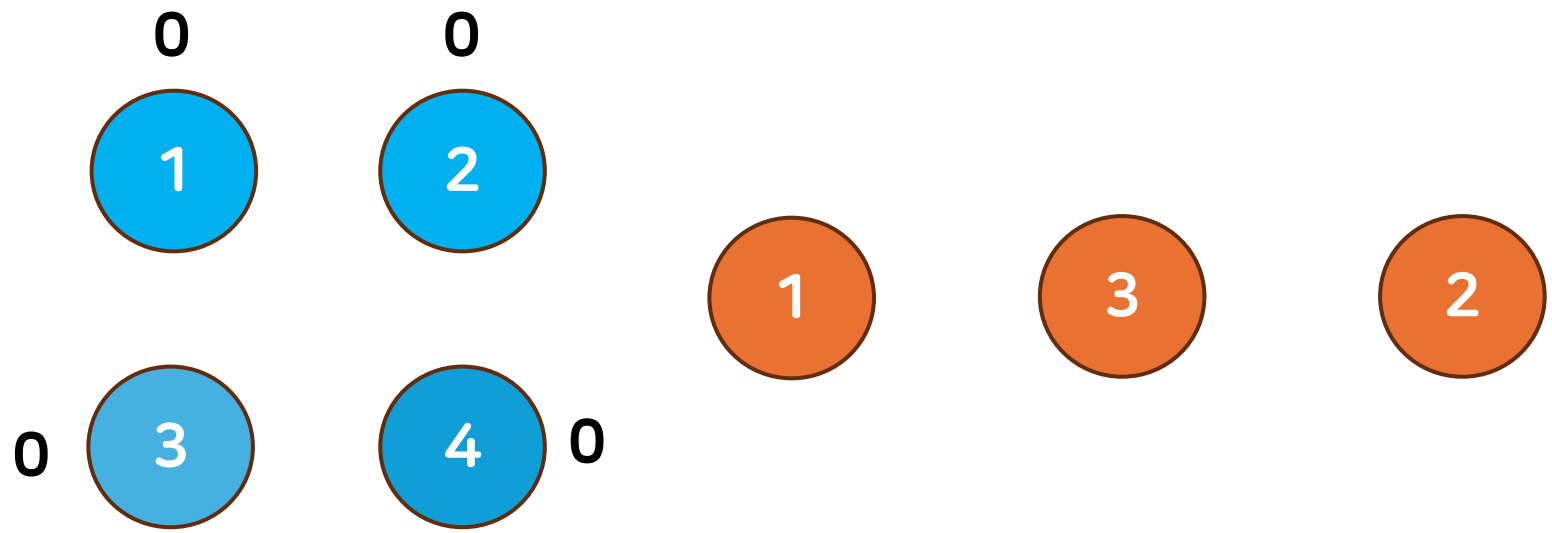
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



줄 세우기 / 2252

우선 indegree가 0인 값에서 부터 시작해야 함

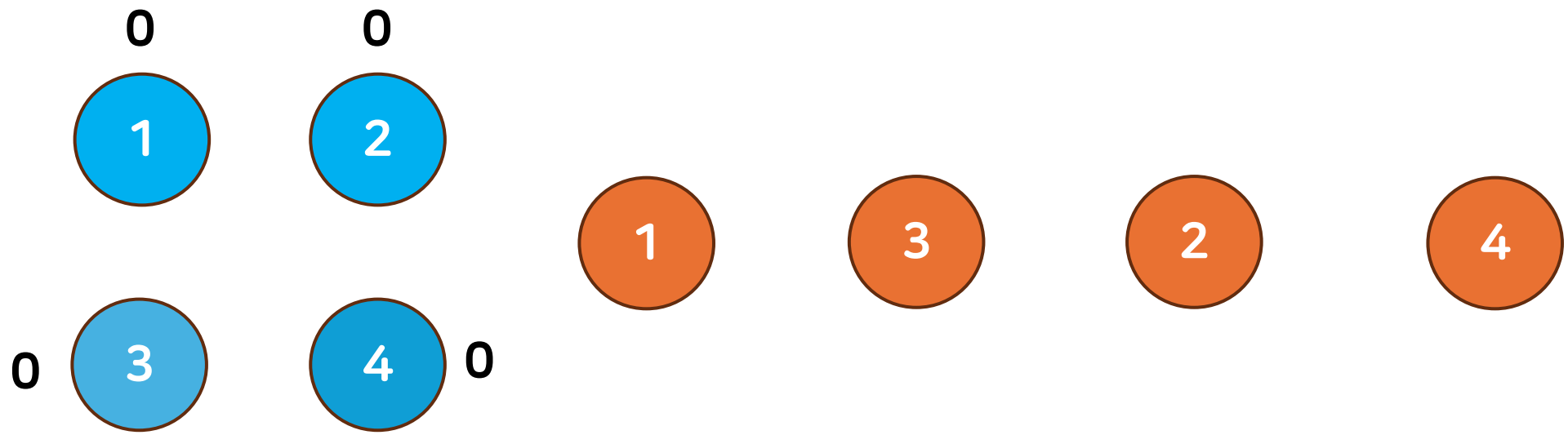
2 - 1. 큐의 다음 정점들의 indegree에 1을 빼줌
2 – 2. 큐의 다음 정점의 indegree가 0이면 큐에 삽입



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

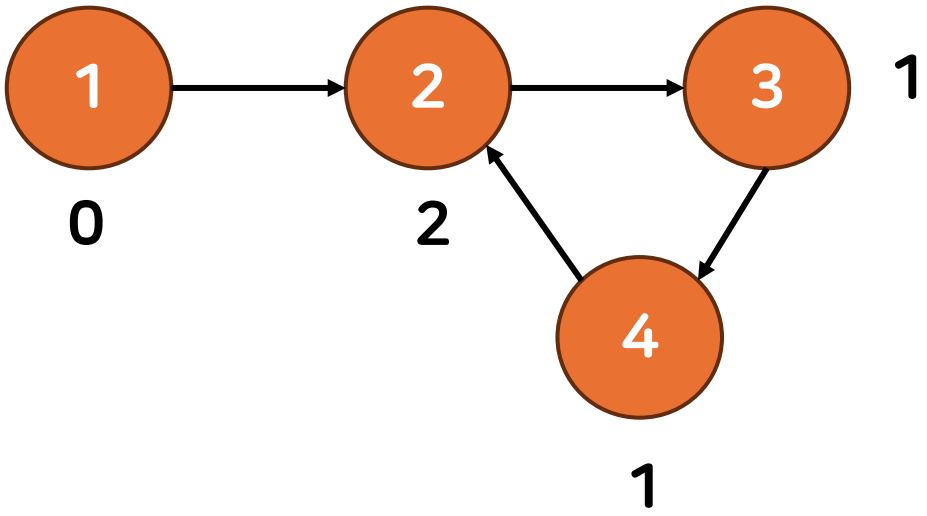
1. 모든 정점을 돌면서 indegree가 0인 값을 큐에 삽입

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

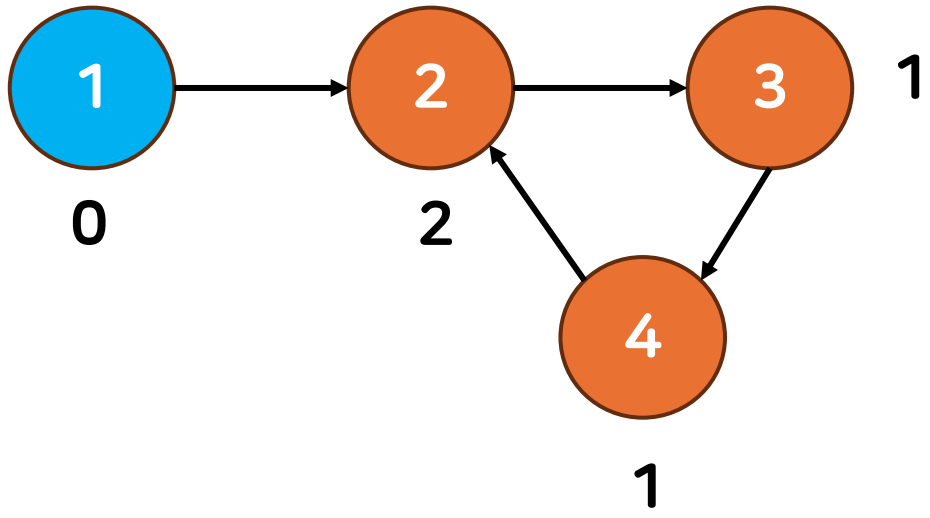
줄 세우기 / 2252

사이클이 존재 할 때 위상 정렬을 하면?



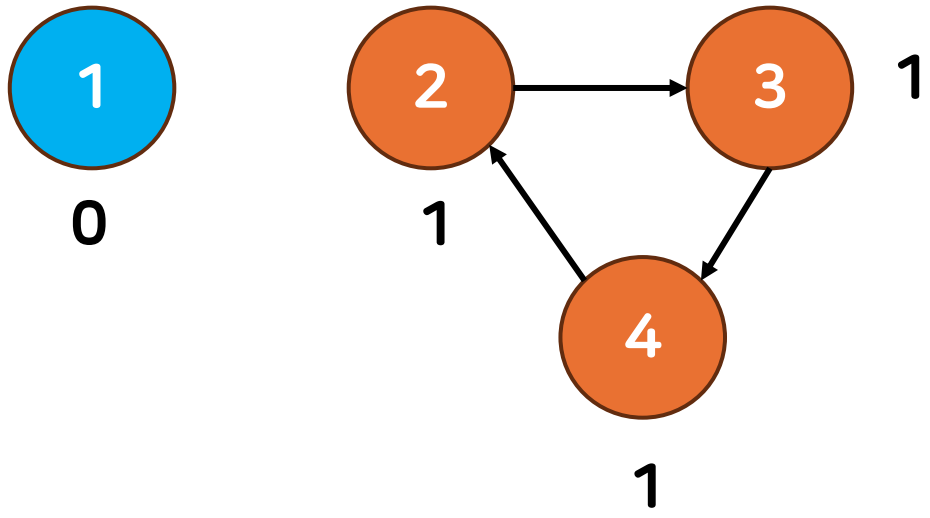
줄 세우기 / 2252

사이클이 존재 할 때 위상 정렬을 하면?



줄 세우기 / 2252

사이클이 존재 할 때 위상 정렬을 하면?

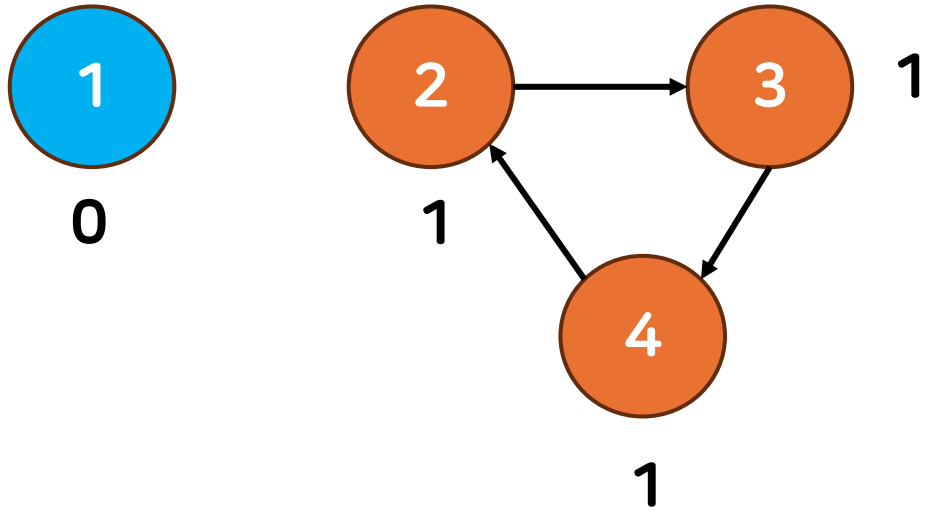


줄 세우기 / 2252

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?

ACM Craft / 1005

백준 1005 / https://www.acmicpc.net/problem/1005

이번 게임에서는 다음과 같이 건설 순서 규칙이 주어졌다. 1번 건물의 건설이 완료된다면 2번과 3번의 건설을 시작할수 있다. (동시에 진행이 가능하다) 그리고 4번 건물을 짓기 위해서는 2번과 3번 건물이 모두 건설 완료되어야지만 4번건물의 건설을 시작할수 있다.

따라서 4번건물의 건설을 완료하기 위해서는 우선 처음 1번 건물을 건설하는데 10초가 소요된다. 그리고 2번 건물과 3번 건물을 동시에 건설하기 시작하면 2번은 1초뒤에 건설이 완료되지만 아직 3번 건물이 완료되지 않았으므로 4번 건물을 건설할 수 없다. 3번 건물이 완성되고 나면 그때 4번 건물을 지을수 있으므로 4번 건물이 완성되기까지는 총 120초가 소요된다.

프로게이머 최백준은 애인과의 데이트 비용을 마련하기 위해 서강대학교배 ACM크래프트 대회에 참가했다! 최백준은 화려한 컨트롤 실력을 가지고 있기 때문에 모든 경기에서 특정 건물만 짓는다면 무조건 게임에서 이길 수 있다. 그러나 매 게임마다 특정건물을 짓기 위한 순서가 달라지므로 최백준은 좌절하고 있었다. 백준이를 위해 특정건물을 가장 빨리 지을 때까지 걸리는 최소시간을 알아내는 프로그램을 작성해주자.

입력

첫째 줄에는 테스트케이스의 개수 T가 주어진다. 각 테스트 케이스는 다음과 같이 주어진다. 첫째 줄에 건물의 개수 N과 건물간의 건설순서 규칙의 총 개수 K이 주어진다. (건물의 번호는 1번부터 N번까지 존재한다)

둘째 줄에는 각 건물당 건설에 걸리는 시간 D_1, D_2, ..., D_N이 공백을 사이로 주어진다. 셋째 줄부터 K+2줄까지 건설순서 X Y가 주어진다. (이는 건물 X를 지은 다음에 건물 Y를 짓는 것이 가능하다는 의미이다)

마지막 줄에는 백준이가 승리하기 위해 건설해야 할 건물의 번호 W가 주어진다.

출력

건물 W를 건설완료 하는데 드는 최소 시간을 출력한다. 편의상 건물을 짓는 명령을 내리는 데는 시간이 소요되지 않는다고 가정한다.

건설순서는 모든 건물이 건설 가능하도록 주어진다.

ACM Craft / 1005

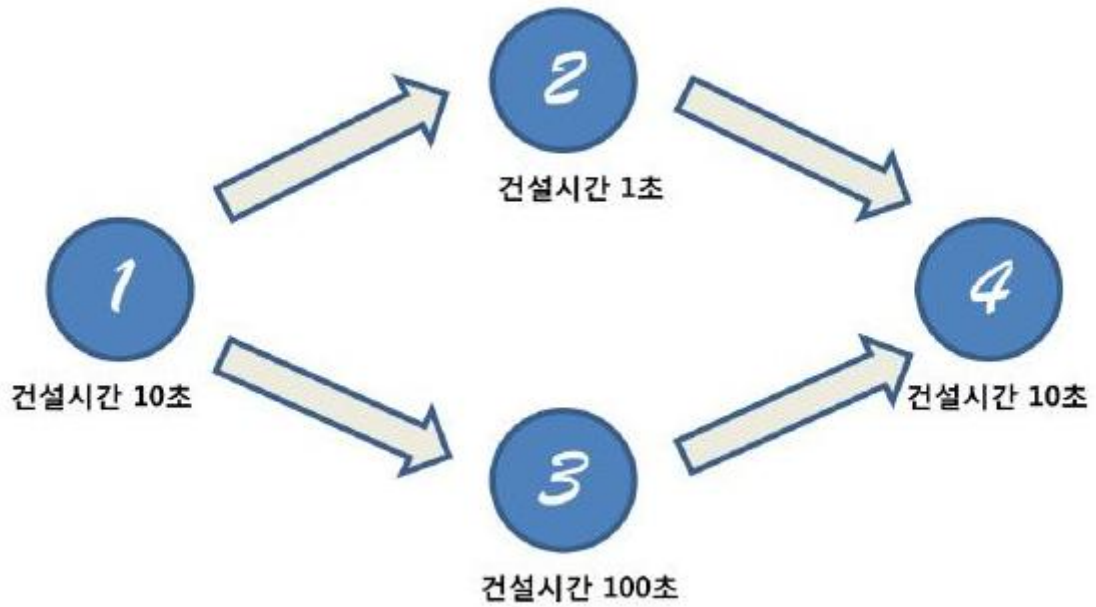
어떤 건물을 짓기 위해서는 미리 지어야 하는 순서가 존재함
K를 짓기 위해서 필요한 최소 시간을 구하는 문제

예제 입력 1 복사

```
4 4
10 1 100 10
1 2
1 3
2 4
3 4
```

예제 출력 1 복사

```
120
```



ACM Craft / 1005

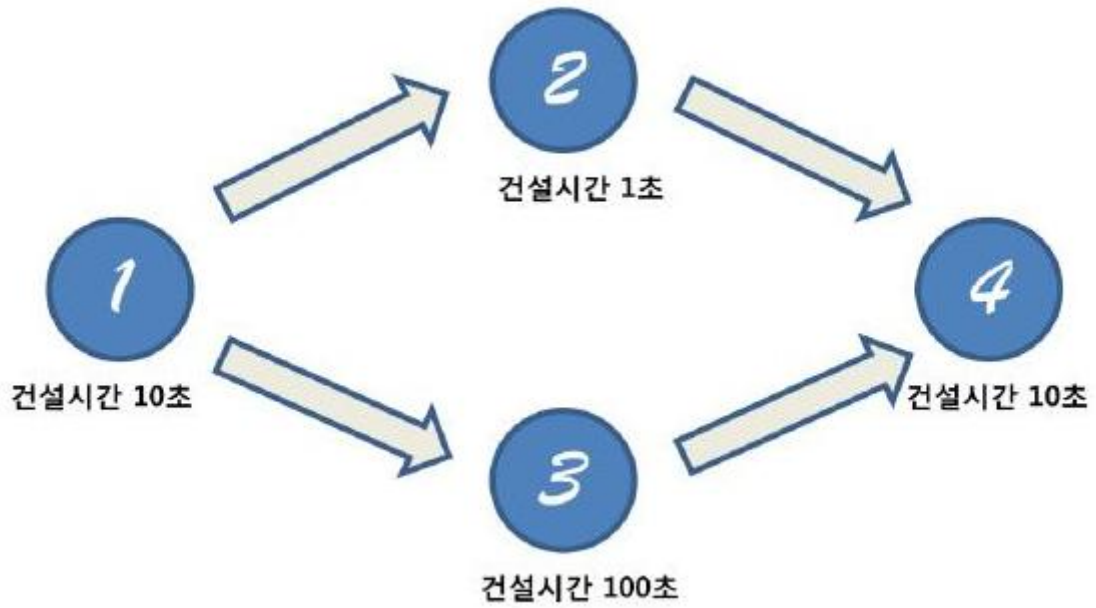
건물 4를 짓기 위해서는 건물 2와 건물 3을 지어야 함
건물 2와 건물 3을 짓기 위해서는 건물 1을 지어야 함

예제 입력 1 복사

```
4 4
10 1 100 10
1 2
1 3
2 4
3 4
```

예제 출력 1 복사

```
120
```



ACM Craft / 1005

건설순서는 모든 건물이 건설 가능하도록 주어진다.

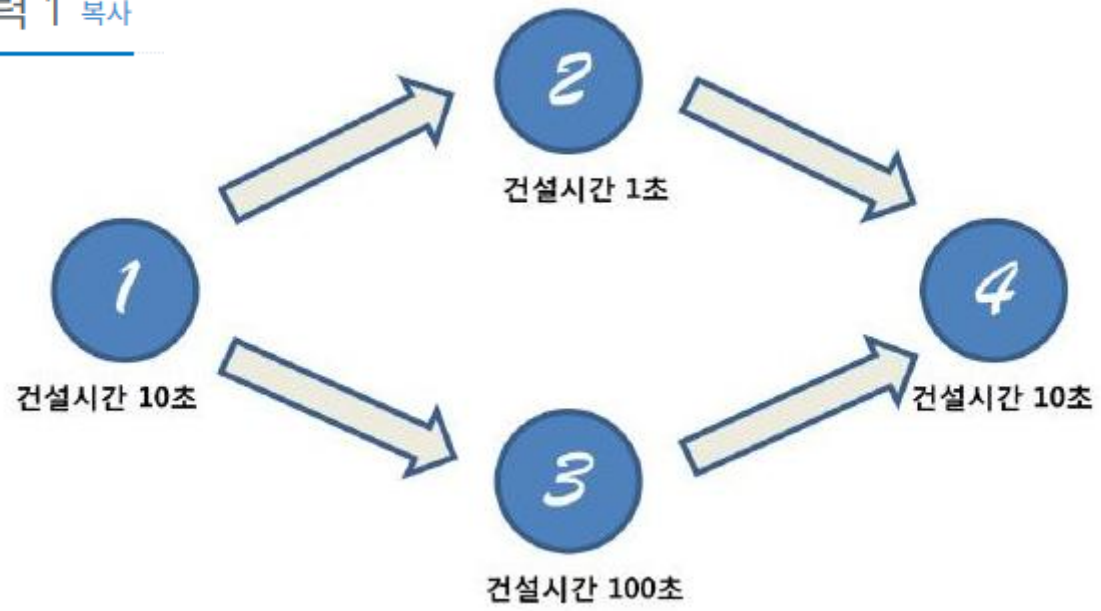
위상 정렬이 가능하다 -> 사이클이 없는 그래프
DP가 가능하다 -> 사이클이 없는 그래프

예제 입력 1 복사

```
4 4
10 1 100 10
1 2
1 3
2 4
3 4
```

예제 출력 1 복사

```
120
```



ACM Craft / 1005

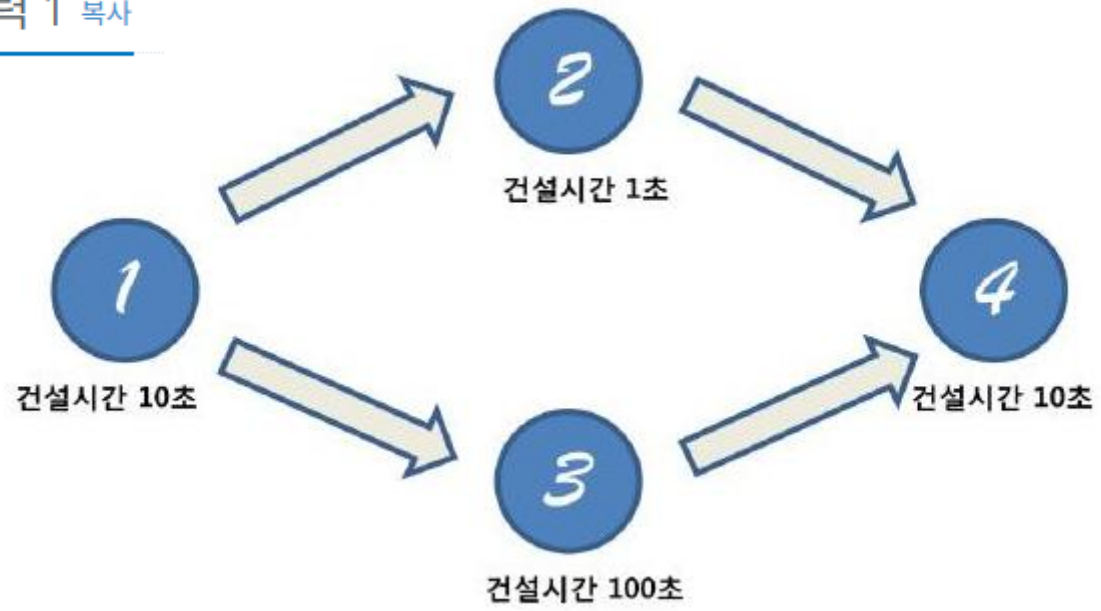
어떤 건물 K를 지을 수 있는 최단 시간
K를 짓기 이전에 지어야 하는 건물 중
최대로 걸리는 시간 + 건물 K를 짓는 데 걸리는 시간

예제 입력 1 복사

```
4 4
10 1 100 10
1 2
1 3
2 4
3 4
```

예제 출력 1 복사

```
120
```



ACM Craft / 1005

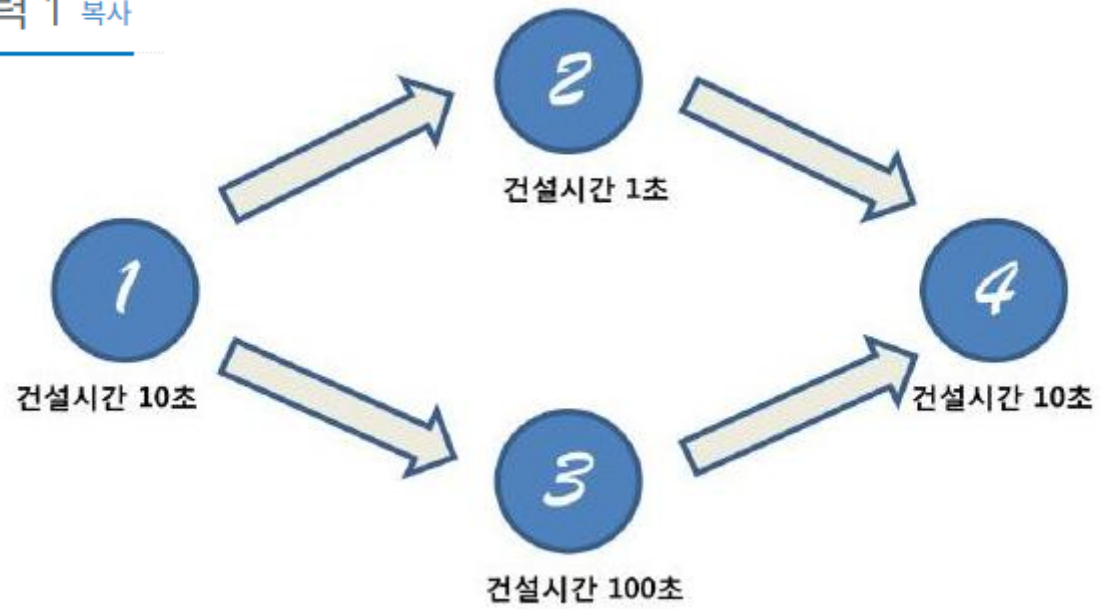
DP[i] = i 번째 건물을 짓는데 걸리는 최소 시간
DP[4] = max(DP[2], DP[3]) + 10

예제 입력 1 복사

```
4 4
10 1 100 10
1 2
1 3
2 4
3 4
```

예제 출력 1 복사

```
120
```



ACM Craft / 1005

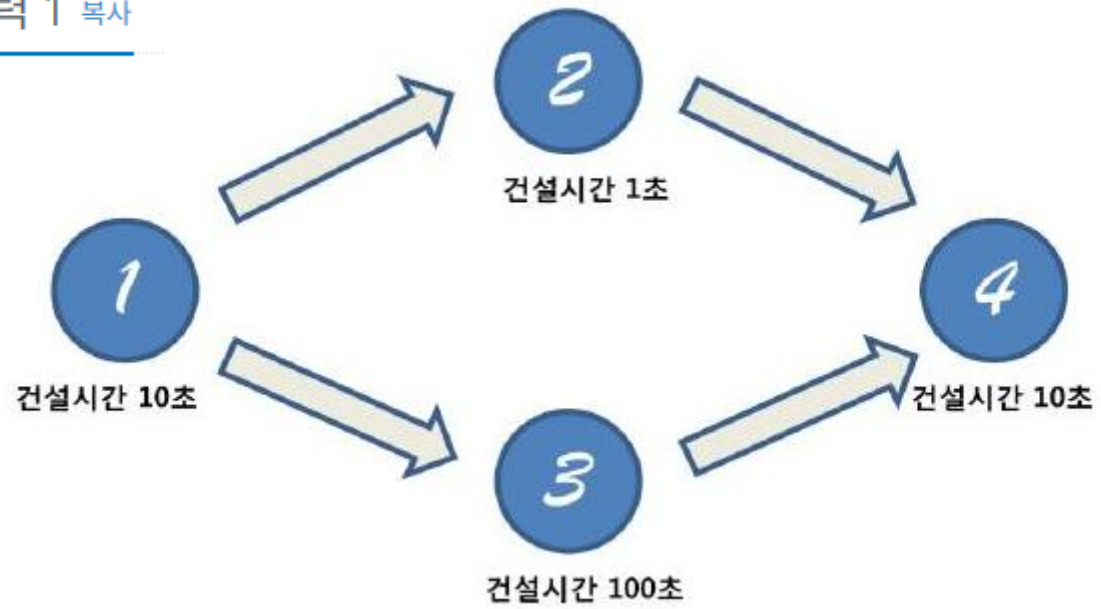
DP[4] = max(DP[2], DP[3]) + 10
DP[4]를 채우기 위해서 DP[2], DP[3] 값을 알아야 함
이 순서는 위상 정렬 순서와 같음

예제 입력 1 복사

```
4 4
10 1 100 10
1 2
1 3
2 4
3 4
```

예제 출력 1 복사

```
120
```



ACM Craft / 1005

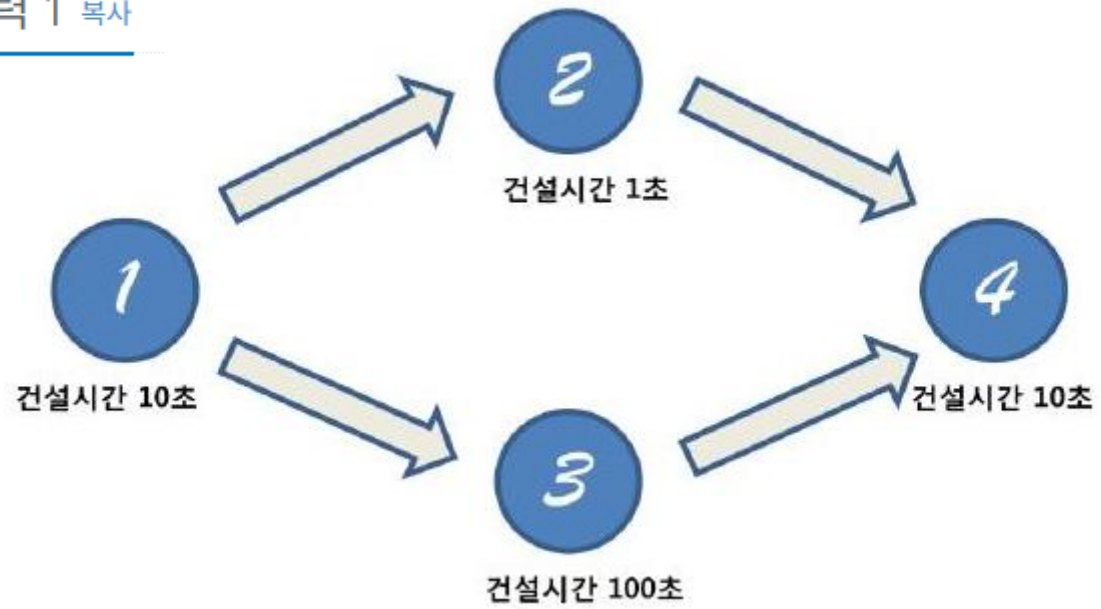
이 순서는 위상 정렬 순서와 같음
-> 위상 정렬을 하면서 DP 테이블을 채워보자

예제 입력 1 복사

```
4 4
10 1 100 10
1 2
1 3
2 4
3 4
```

예제 출력 1 복사

```
120
```



ACM Craft / 1005

C++

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

    ll t; cin >> t;
    while(t--) run();

    return 0;
}
```

```cpp
const ll MAX = 1010;
const ll INF = 1e12;
ll n, m, k, ind[MAX];
ll a[MAX], dp[MAX];
vector <ll> adj[MAX];
queue <ll> q;
```

```cpp
void run(){
    cin >> n >> m;
    // 값 입력 및 초기화
    for(int i = 1;i <= n;i++){
        cin >> a[i];
        dp[i] = ind[i] = 0;
        adj[i].clear();
    }

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

    for(int i = 1;i <= n;i++){
        if(!ind[i]) q.push(i);
    }

    while(!q.empty()){
        ll cur = q.front(); q.pop();
        // 현재 DP값에 현재 건물 짓는 데 걸리는 시간 추가
        dp[cur] += a[cur];
        for(auto& nxt : adj[cur]){
            // 다음 DP 값은 이전의 건물을 짓는 데 드는
            // 시간 중 최댓값으로 갱신
            dp[nxt] = max(dp[nxt], dp[cur]);
            if(!--ind[nxt]) q.push(nxt);
        }
    }

    ll result = 0;
    cout << dp[k] << "\n";
}
```

ACM Craft / 1005

C++

```
t = int(input())
for _ in range(t):
    run()
```

```
MAX = 1010
INF = 10**12

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

```
def run():
    global n, m, k, ind, a, dp, adj, q
    q.clear()
    n, m = map(int, input().split())

    # 값 입력 및 초기화
    for i in range(1, n+1):
        a[i] = int(input())
        dp[i] = ind[i] = 0
        adj[i].clear()

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

    k = int(input())

    for i in range(1, n+1):
        if ind[i] == 0:
            q.append(i)

    while q:
        cur = q.popleft()
        # 현재 DP값에 현재 건물 짓는 데 걸리는 시간 추가
        dp[cur] += a[cur]
        for nxt in adj[cur]:
            # 다음 DP 값은 이전의 건물을 짓는 데 드는
            # 시간 중 최댓값으로 갱신
            dp[nxt] = max(dp[nxt], dp[cur])
            ind[nxt] -= 1
            if ind[nxt] == 0:
                q.append(nxt)

    print(dp[k])
```

질문?

기본 과제

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

ACM Craft - https://www.acmicpc.net/problem/1005

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

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

장난감 조립 - https://www.acmicpc.net/problem/2637

고생하셨습니다