#10-1. 위상 정렬

나정휘

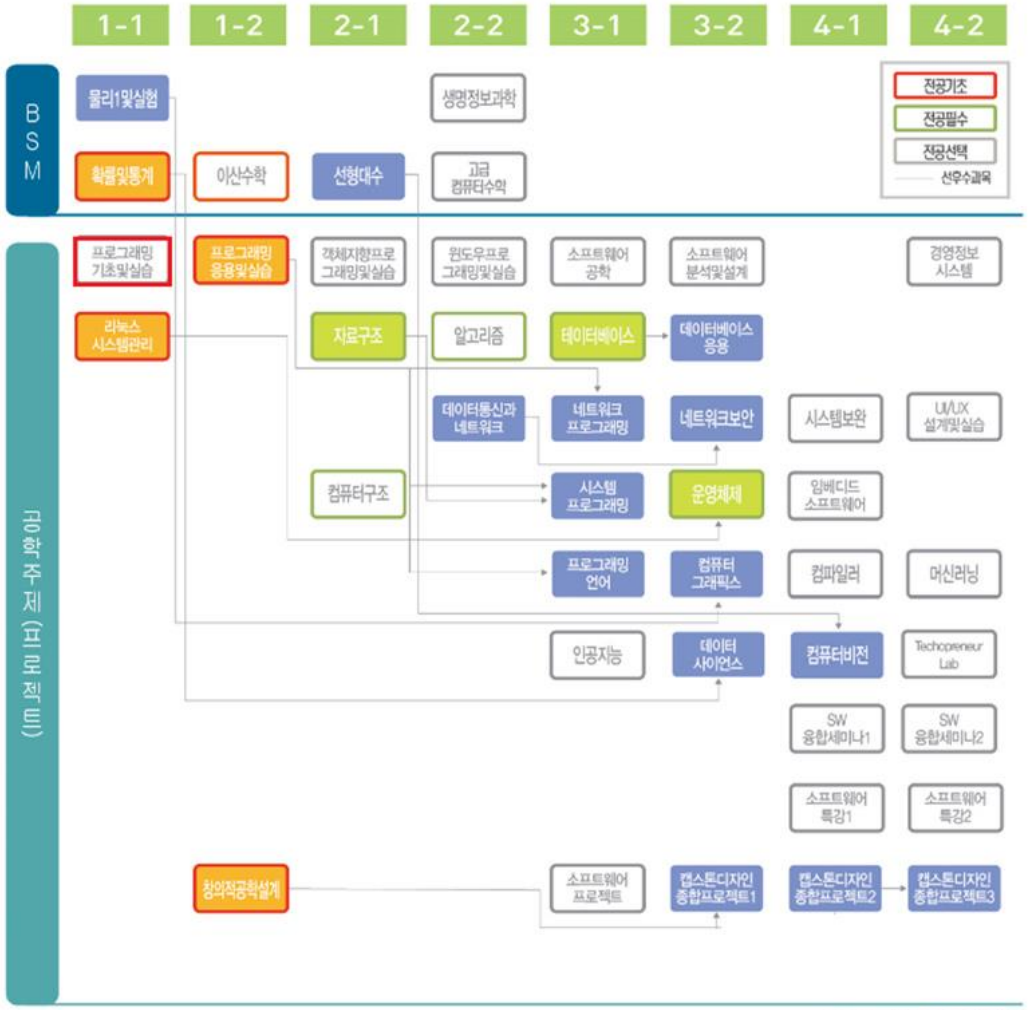
https://justiceHui.github.io/

위상 정렬

위상 정렬

- 간선의 방향을 거스르지 않는 순서대로 정점을 나열하는 것
  - A → B 간선이 있다면 A는 B보다 먼저 나와야 함
  - 위상 정렬 순서대로 정점을 나열하면 간선은 모두 오른쪽을 향함
  - DAG(사이클 없는 방향 그래프)에서만 가능
    - Directed Acyclic Graph

- 예시
  - 전공 과목 선후수과목 관계
  - 스타크래프트 건물 건설 순서

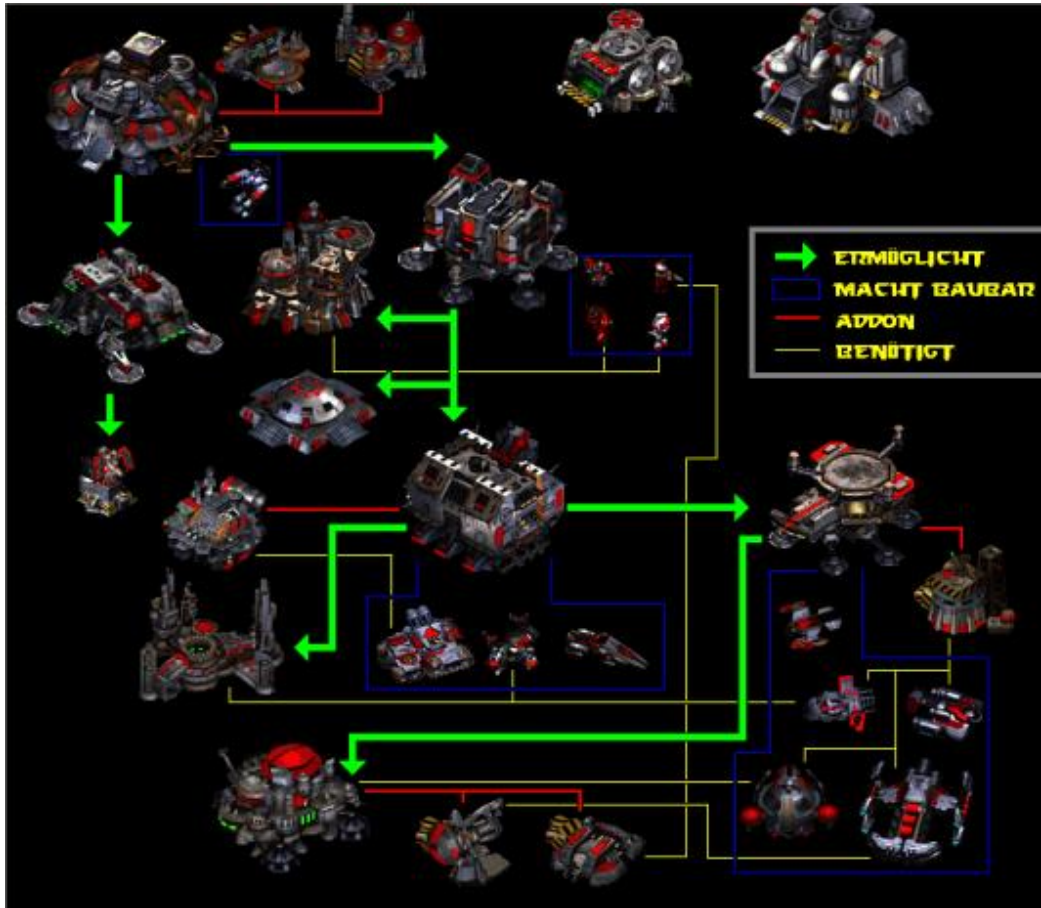


위상 정렬

위상 정렬

- 간선의 방향을 거스르지 않는 순서대로 정점을 나열하는 것
  - A → B 간선이 있다면 A는 B보다 먼저 나와야 함
  - 위상 정렬 순서대로 정점을 나열하면 간선은 모두 오른쪽을 향함
  - DAG(사이클 없는 방향 그래프)에서만 가능
    - Directed Acyclic Graph

- 예시
  - 전공 과목 선후수과목 관계
  - 스타크래프트 건물 건설 순서



위상 정렬

위상 정렬을 구하는 방법

- BFS를 응용하는 방법(Kahn's Algorithm)
- DFS를 응용하는 방법

위상 정렬의 성질

- 가장 앞에 오는 정점은 항상 in-degree가 0임
  - 만약 0이 아니라면 그 정점보다 다른 정점이 먼저 나와야 함
- in-degree가 0이면서 맨 앞이 아닌 정점은 앞으로 옮겨도 됨

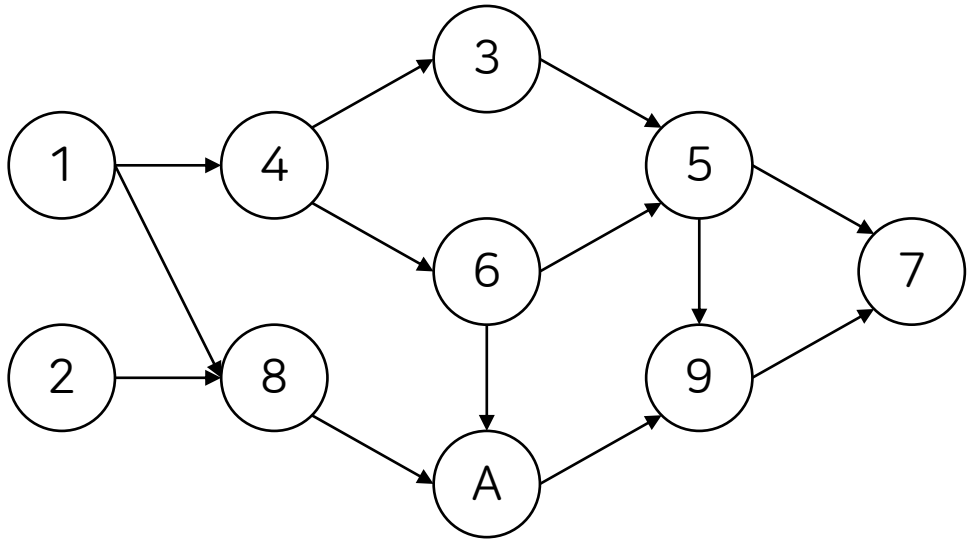
위상 정렬

Kahn's Algorithm

- in-degree가 0인 정점을 큐에 삽입
- 큐가 비어 있지 않으면 아래 과정을 반복
  - 큐에서 정점 v를 제거
  - v를 정답에 추가
  - v에서 나가는 간선을 모두 제거, 해당 정점의 in-degree 1 감소
  - in-degree가 0이 된 정점을 큐에 삽입

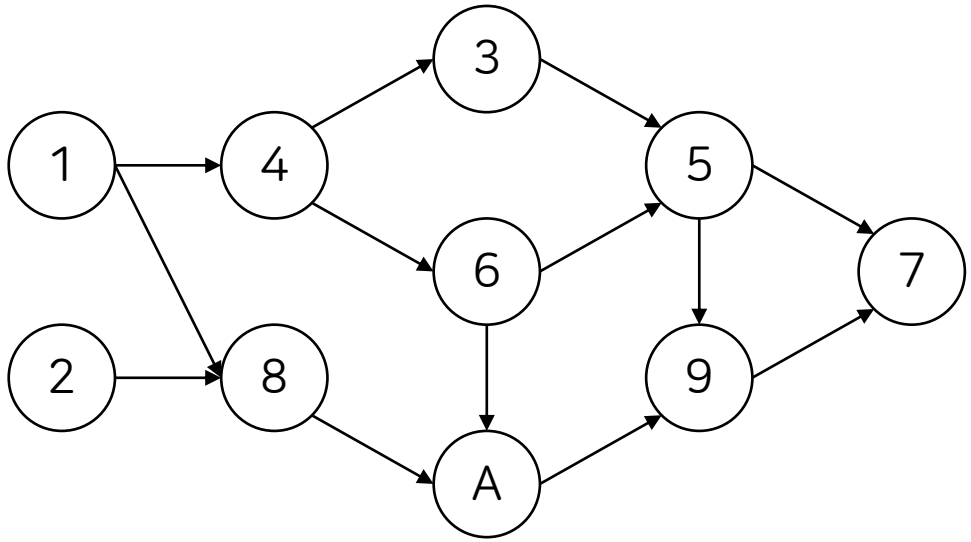
- 사이클이 존재하면 모든 정점을 방문하기 전에 종료됨

위상 정렬



- Q = {1, 2}
- V = 1 Q = {2, 4}
- V = 2 Q = {4, 8}
- V = 4 Q = {8, 3, 6}
- V = 8 Q = {3, 6}
- V = 3 Q = {6}
- V = 6 Q = {A, 5}
- V = A Q = {5}
- V = 5 Q = {9}
- V = 9 Q = {7}
- V = 7 Q = {}
- 1 2 4 8 3 6 A 5 9 7

위상 정렬



- Q = {2, 1}
- V = 2 Q = {1}
- V = 1 Q = {8, 4}
- V = 8 Q = {4}
- V = 4 Q = {6, 3}
- V = 6 Q = {3, A}
- V = 3 Q = {A, 5}
- V = A Q = {5, 9}
- V = 5 Q = {9}
- V = 9 Q = {7}
- V = 7 Q = {}
- 2 1 8 4 6 3 A 5 9 7

위상 정렬

Kahn's Algorithm

- 시간 복잡도: O(|V| + |E|)
- 큐에 있는 원소들의 출력 순서는 상관 없음
  - 큐에 들어간 순간 언제든지 위상 정렬 결과에 추가할 수 있음
  - 편의상 앞에 있는 원소부터 사용하는 것
  - BOJ 1766 문제집: 큐에서 가장 작은 원소부터 꺼내야 함

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

int N, M, In[32323];
vector<int> G[32323];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1; i<=M; i++){
        int s, e; cin >> s >> e;
        G[s].push_back(e);
        In[e] += 1;
    }
    queue<int> Q;
    for(int i=1; i<=N; i++) if(!In[i]) Q.push(i);
    while(!Q.empty()){
        int v = Q.front(); Q.pop();
        cout << v << " ";
        for(auto i : G[v]) if(!--In[i]) Q.push(i);
    }
}
```

질문?

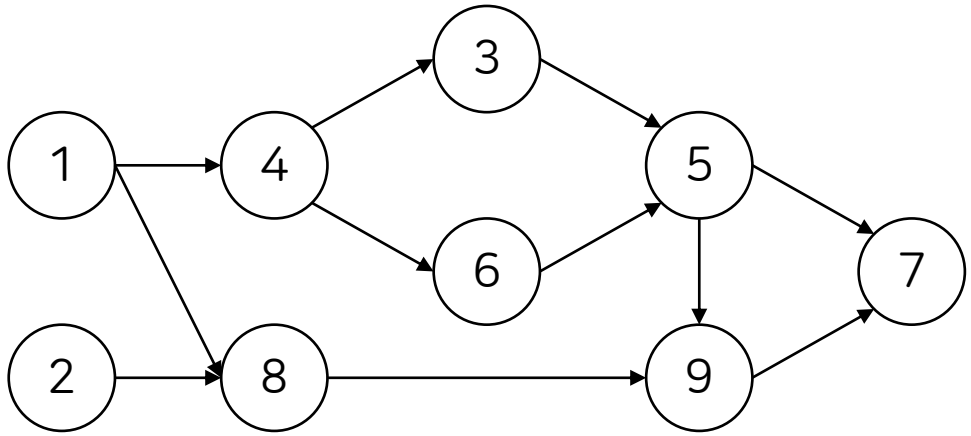
위상 정렬

깊이 우선 탐색

- DFS를 하면서, 정점을 빠져나오는 순서대로 기록
- 그 순서의 역순은 위상 정렬임

- 간선 A → B가 있을 때
- A를 B보다 먼저 방문하면 항상 B를 먼저 탈출함
  - B를 탈출한 다음에 A를 탈출해야 함
- B를 A보다 먼저 방문하면 항상 B를 먼저 탈출함
  - B를 탈출한 다음에 A를 방문할 수 있음
  - B를 탈출하기 전에 A를 방문하면 사이클

위상 정렬



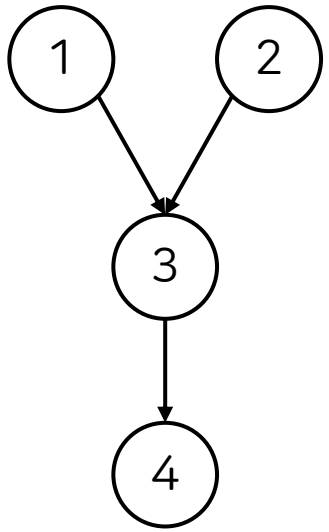
- 1 진입
- 4 진입
- 6 진입
- 5 진입
- 7 진입
- 7 탈출(5)
- 9 진입
- 9 탈출(5)
- 5 탈출(6)
- 7 9 5 6 3 4 8 1 2
- 2 1 8 4 3 6 5 9 7

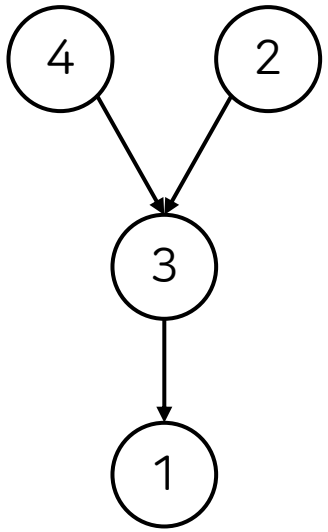
6 탈출(4)
3 진입
3 탈출(4)
4 탈출(1)
8 진입
8 탈출(1)
1 탈출
2 진입
2 탈출

위상 정렬

깊이 우선 탐색

- 시간 복잡도: O(|V| + |E|)
- 방문하지 않은 모든 정점에 대해 DFS를 호출해야 함
  - 아래 그래프에서 DFS(1)만 호출하면 모든 정점을 볼 수 없음





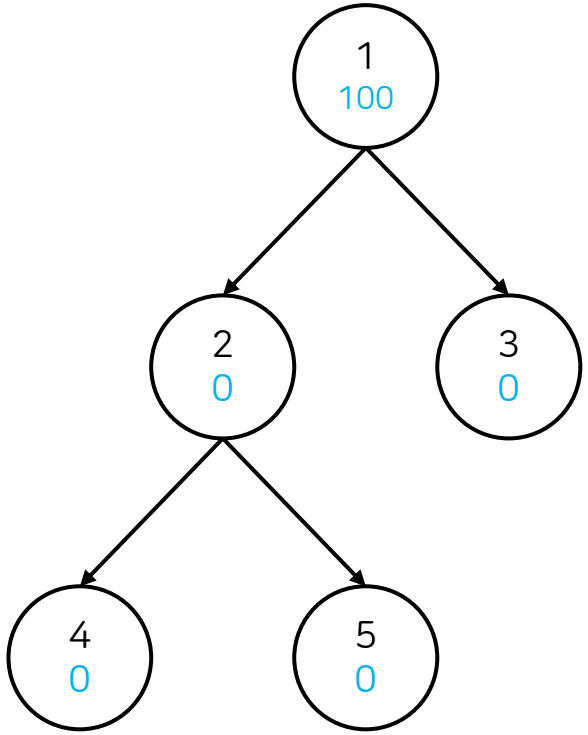

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, M, C[32323];
vector<int> G[32323], V;

void DFS(int v){
    C[v] = 1;
    for(auto i : G[v]) if(!C[i]) DFS(i);
    V.push_back(v);
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1; i<=M; i++){
        int s, e; cin >> s >> e;
        G[s].push_back(e);
    }
    for(int i=1; i<=N; i++) if(!C[i]) DFS(i);
    reverse(V.begin(), V.end());
    for(auto i : V) cout << i << " ";
}
```

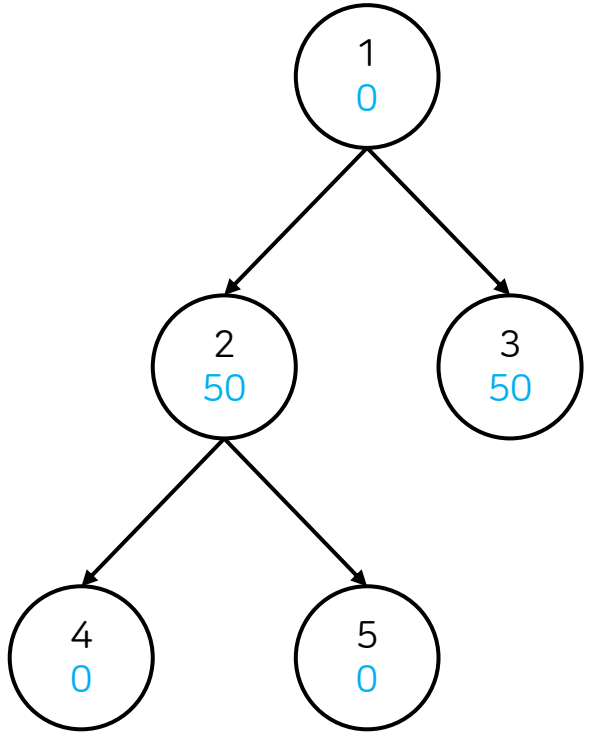
질문?

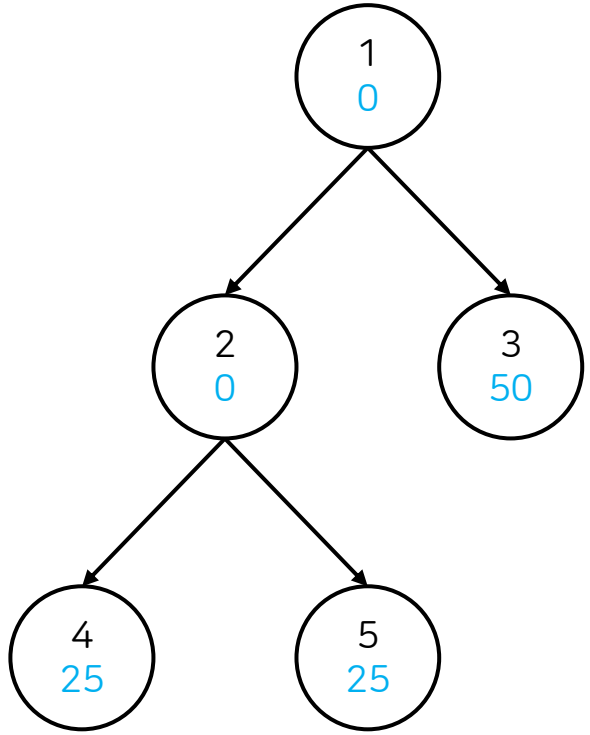
위상 정렬

BOJ 28360 양동이 게임

- 양동이가 DAG 형태로 연결되어 있음 (모든 간선은 번호가 작은 정점에서 큰 정점으로 가는 방향)
- 양동이의 out-degree가 0 초과면 양동이에 담긴 물이 연결되어 있는 양동이로 균등하게 이동함
- 양동이에 담긴 물의 양의 최댓값을 구하는 문제







위상 정렬

BOJ 28360 양동이 게임

- 양동이가 DAG 형태로 연결되어 있음 (모든 간선은 번호가 작은 정점에서 큰 정점으로 가는 방향)
- 양동이의 out-degree가 0 초과면 양동이에 담긴 물이 연결되어 있는 양동이로 균등하게 이동함
- 양동이에 담긴 물의 양의 최댓값을 구하는 문제

- 번호가 작은 정점에서 큰 정점으로 가는 간선만 존재하므로 그래프에 사이클 없음
- 정점 번호 자체가 위상 정렬 순서임

- 번호가 큰 정점에서 작은 정점으로 물이 흘러가지 않으므로
- 번호가 작은 정점부터 차례대로 물을 흘려보내면 됨

- out-degree가 0일 때 0으로 나누지 않도록 조심

위상 정렬

BOJ 28360 양동이 게임

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

int N, M;
double D[55];
vector<int> G[55];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M; D[1] = 100;
    for(int i=1,s,e; i<=M; i++) cin >> s >> e, G[s].push_back(e);
    for(int i=1; i<=N; i++){
        if(G[i].empty()) continue;
        double nxt = D[i] / G[i].size();
        for(auto j : G[i]) D[j] += nxt;
        D[i] = 0;
    }
    cout << fixed << setprecision(20) << *max_element(D+1, D+N+1);
}
```

위상 정렬

BOJ 28360 양동이 게임

- 작은 정점 → 큰 정점 조건이 없으면 직접 위상 정렬 구해야 함
  - 처음에 Q.push(1); 만 하면 안 됨
  - 1이 아닌 in-degree = 0 정점 존재할 수 있음

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

int N, M, In[55];
double D[55];
vector<int> G[55];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M; D[1] = 100;
    for(int i=1,s,e; i<=M; i++) cin >> s >> e, G[s].push_back(e), In[e]++;

    queue<int> Q;
    for(int i=1; i<=N; i++) if(!In[i]) Q.push(i);
    while(!Q.empty()){
        int v = Q.front(); Q.pop();
        for(auto i : G[v]){
            D[i] += D[v] / G[v].size();
            if(!--In[i]) Q.push(i);
        }
        if(!G[v].empty()) D[v] = 0;
    }
    cout << fixed << setprecision(20) << *max_element(D+1, D+N+1);
}
```

질문?