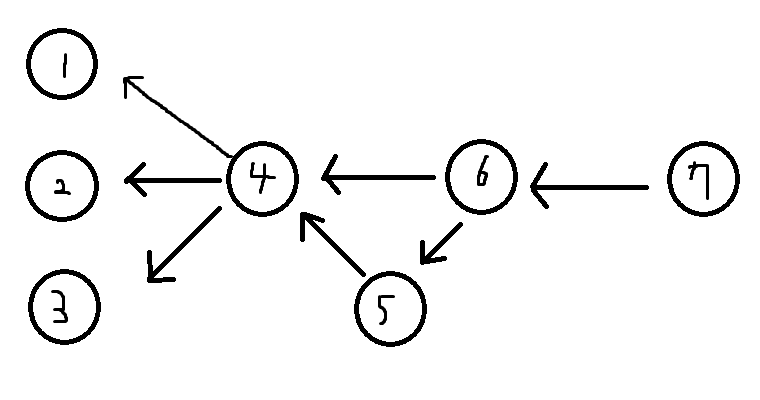
목차

[위상정렬 1](#_Toc126618902)

위상정렬(Topological sort)

* 위상정렬이란?
  + 사이클이 없는 방향 그래프(DAG: Directed Acyclic Graph)에서 정점 사이의 선후 관계를 위배하지 않고 정렬하는 알고리즘.
  + Ex) 수강순서를 정하는 문제, 키 순서대로 서는 문제
* 위상정렬의 시간복잡도
  + O(V+E) V는 정점의 수, E는 간선의 수
* 알고리즘
  + 내가 이해한 위상정렬의 핵심은. 일을 마치기 전까지는 자신의 순번이 오지 않는다는 것이다. 나는 이를 Workout이라 표하기로 했다. 그래프들은 각각의 간선을 가지고 있고 사이클은 존재하지 않는다.



이러한 그래프를 기준으로 설명한다.

* 설명

1. 각 노드들의 workout은 자신이 가리키고 있는 간선의 개수를 뜻한다.

1,2,3 – 0개 4 – 3개 7 – 1개

5 – 1개 6 – 2개

1. Queue를 준비한다. (재정렬 할 컨테이너)
2. 먼저 전체를 탐색하며 Workout이 0인 노드들을 Queue에 넣는다.
3. Queue에서 하나를 뺀 값을 보관한다 (정렬된 데이터가 들어갈 곳임)
4. Queue에서 뺀 값의 Child들(현재 노드를 가리키는 노드들)의 workout을 1개 감소시킨다. 이는 child노드가 할 일이 1개 줄었다는 것을 의미한다.
5. 만약 4번에서 Child노드의 Workout이 줄어서 0이 됐다면, Queue에 넣는다.
6. 4번으로 돌아가고 Queue에 모든 요소가 없어질때까지 반복한다.

예외 : 만약 이 루프가 끝났는데 정렬되지 않은 노드들이 있다면 그것은 사이클 형태를 띄고있는 것이다.

* 소스코드 (백준 2252번 줄 세우기)

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

class Line {

public:

vector<Line\*> childs;

int workout;

int value;

Line():

workout(0)

,value(0)

{

}

void Push(Line\* child)

{

childs.push\_back(child);

child->workout++;

}

};

int main()

{

std::ios\_base::sync\_with\_stdio(false);

std::cin.tie(NULL);

int N, M;

cin >> N >> M;

Line\* arr;

arr = new Line[N] {};

for (int i = 0; i < M; ++i)

{

int a, b;

cin >> a >> b;

arr[a - 1].Push(&arr[b - 1]);

}

queue<Line\*> que;

for (int i = 0; i < N; ++i)

{

arr[i].value = i + 1;

if (0 == arr[i].workout)

que.push(&arr[i]);

}

while (!que.empty())

{

auto p = que.front();

que.pop();

cout << p->value << " ";

for (auto& child : p->childs)

{

child->workout--;

if (0 == child->workout)

que.push(child);

}

}

delete[] arr;

return 0;

}