


그래프 (Graph)

- 위상정렬
- 에지에 가중치가 있는(weighted) DAG에서 최장경로(longest path)문제

5. 위상정렬 (topological sort)

- 그래프의 위상순서(topological order): DAG $G = (V, E)$ 의 정점들에 다음과 같이 1부터 n (정점 수)까지 번호를 부여: $(v, w) \in E(G)$ 이면 v 의 번호가 w 의 번호보다 작다.
- DAG $G = (V, E)$ 의 위상정렬(topological sort): V 의 정점들을 다음 조건을 만족하면서 일렬로 나열하는 것: $(v, w) \in E(G)$ 이면 v 가 w 보다 앞서 나와야 한다.
 방향은 있는데 사이클은 X인 그래프
- 방향그래프 G 가 사이클을 가지고 있으면 G 는 위상정렬을 할 수 없다

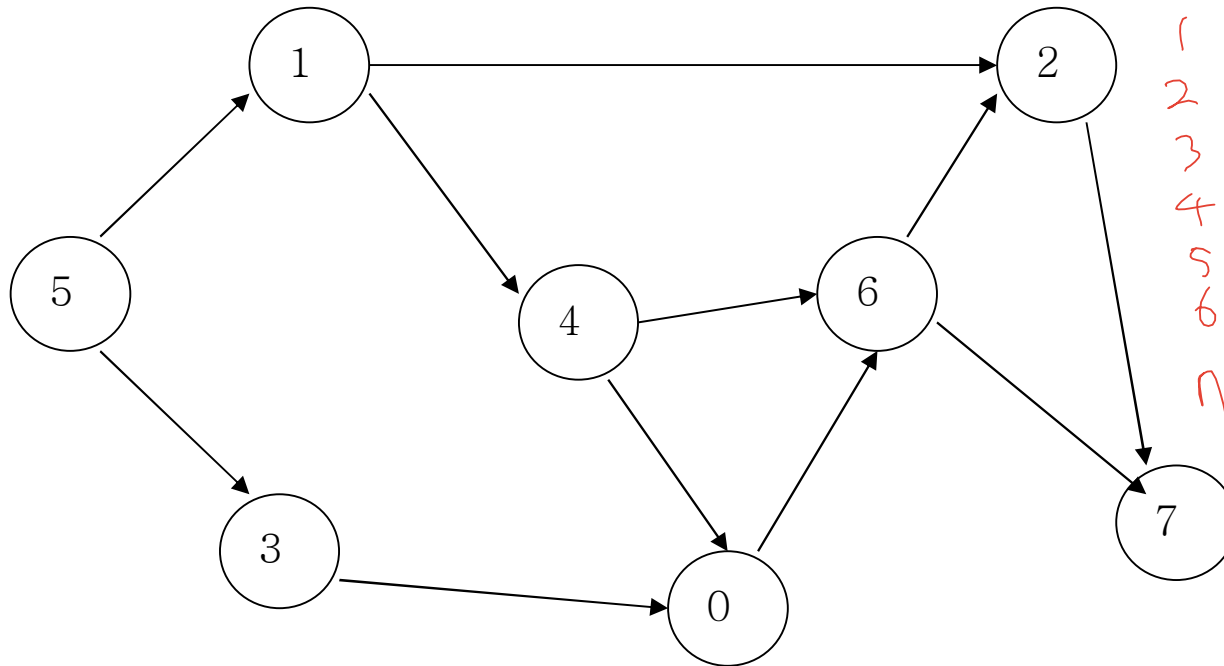
종속 태스크 스케줄링

- 종속 태스크 스케줄링
n개의 태스크들과 이들 태스크들 사이의 종속관계가 주어져 있다: 두 태스크 t_1, t_2 에 대하여 t_2 가 t_1 에 종속된다 $\leftrightarrow t_1$ 이 끝나야만 t_2 를 시작할 수 있다.
- 태스크들의 종속관계를 만족하는 태스크들의 수행 순서를 구하라.

태스크	태스크 번호	종속 태스크
옷 고르기	1	9
옷 입기	2	1, 8
아침식사	3	5, 6, 7
출발	4	2, 3
커피 만들기	5	9
토스트 만들기	6	9
주스 따르기	7	9
샤워하기	8	9
잠깨기	9	—

위상정렬 예

- 위상정렬의 예



위상정렬결과: 5, 1, 3, 4, 0, 6, 2, 7

위상정렬결과는 여러 가지가 나올 수 있다.

2의 인접 분지수는 0, 1, 2, 3, 4, 5, 6, 7

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	1	0
1	0	0	1	0	1	0	0	0
2								
3								
4								
5								
6								
7								



진입분지수를 이용한 위상정렬

위상정렬 문제

위상정렬

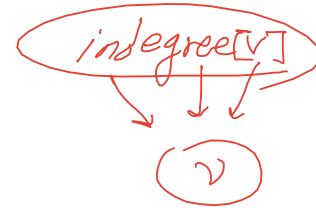
- indegree[v]: 정점 v로 들어오는 에지의 수
정점 v의 진입분지수(indegree)
- 각 정점의 진입분지수(indegree)를 구하여 0인 정점을 v를 출력하고 v에서 나가는 모든 에지 (v,w)에 대하여 w의 진입분지수를 1 감소
- 위 과정을 반복

무향 그래프

v와 인접한 정점들

인접행렬 $O(n^2)$

인접 리스트 $O(n+m)$



```
#include <queue>
```

```
using namespace std;
```

```
....
```

```
queue<int> Q;
```

```
for each vertex v ∈ V
```

```
{
```

```
    v의 indegree[v]를 계산;
```

```
    if(indegree[v] == 0)
```

```
        Q.push(v);
```

```
}
```

```
while(!Q.empty())
{
```

```
    v = Q.front();
```

```
    Q.pop();
```

```
    output v;
```

```
    for each vertex w adjacent from v
```

```
{
```

```
        indegree[w]--;
```

```
        if(indegree[w] == 0)
```

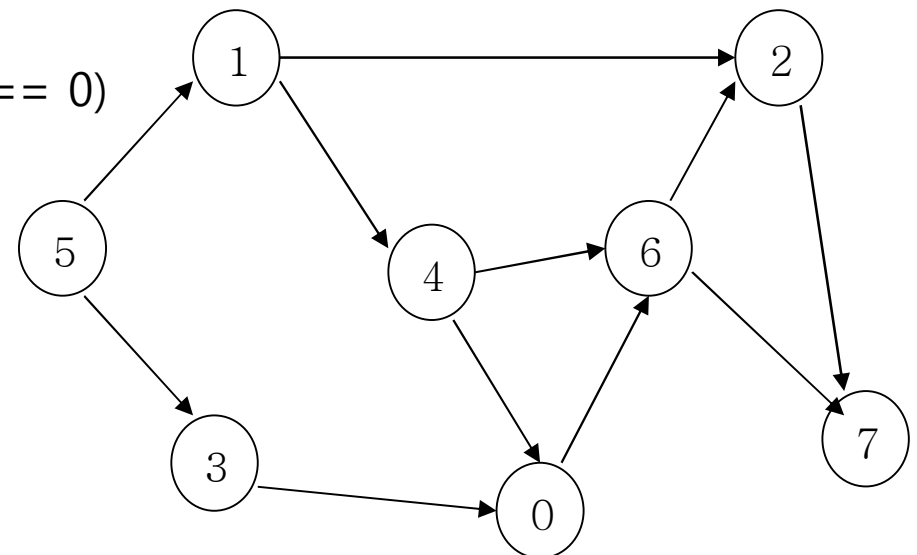
```
            Q.push(w);
```

```
}
```

```
}
```

들어오는 에지 수

0	1	2	3	4	5	6	7
2	1	2	1	0	2	2	



모든 vertex들이 출력되지 않으면 cycle이 있음

깊이우선탐색을 이용한 위상정렬

- **Depth First** 골격을 이용하여 위상정렬 역순으로 정점을 출력하는 프로그램

Algorithm `dfs(g, v, visited)` // `g`는 그래프, `v`는 출발정점
`visited[v] = true` // visit `v`

`v`와 인접한 각 정점 `w`에 대하여

`if(!visited[w])` // if `w` is unvisited
`dfs(g, w, visited)`

`v`를 출력 (`v`와 인접정점을 모두 방문 후 출력)
→ 위상정렬의 역순으로 정점을 정렬해야 함.

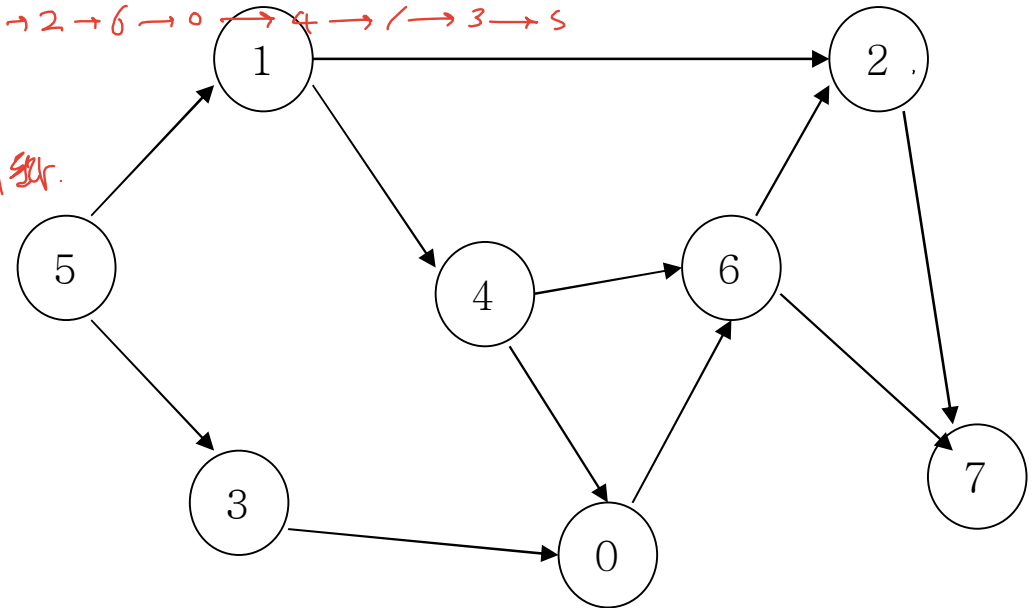
Algorithm `reverseTopologicalSort(g)`

for each vertex `v` in `V`:

if `visited[v] == false`:

`dfs(v)`

dfs 5 → 1 → 2 → 7 → 4 → 6 → 0
출력 7 → 2 → 6 → 0 → 4 → 1 → 3 → 5



6. Weighted DAG에서 최장경로 문제

- 에지에 가중치가 있는 DAG에서 두 정점 s와 t에 대하여 s로부터 t까지의 최장(가장 길이가 긴) 경로를 구하는 문제
- 일반적인 weighted 그래프에서 정점 s로부터 vertex t까지의 최장경로를 찾는 것은 매우 어려운 문제이다.
- DAG에서 vertex s로부터 vertex t까지의 최장(가장 길이가 긴) 경로를 찾는 것은 쉬운 문제이다: **동적계획법을 이용**

$L[u]$: u로부터 부터 정점 t까지 가는 최장경로의 길이 (재귀적으로 구함)

$Out(u) = \{ v \mid (u, v) \in E \}$ // u로부터 인접한 정점들의 집합

$L[u] = \max \{ L[v] + \text{weight}(u,v) \}$

$v \in Out(u)$

$L[]$ 를 **동적계획법**으로 구한다.

$u \rightarrow v$

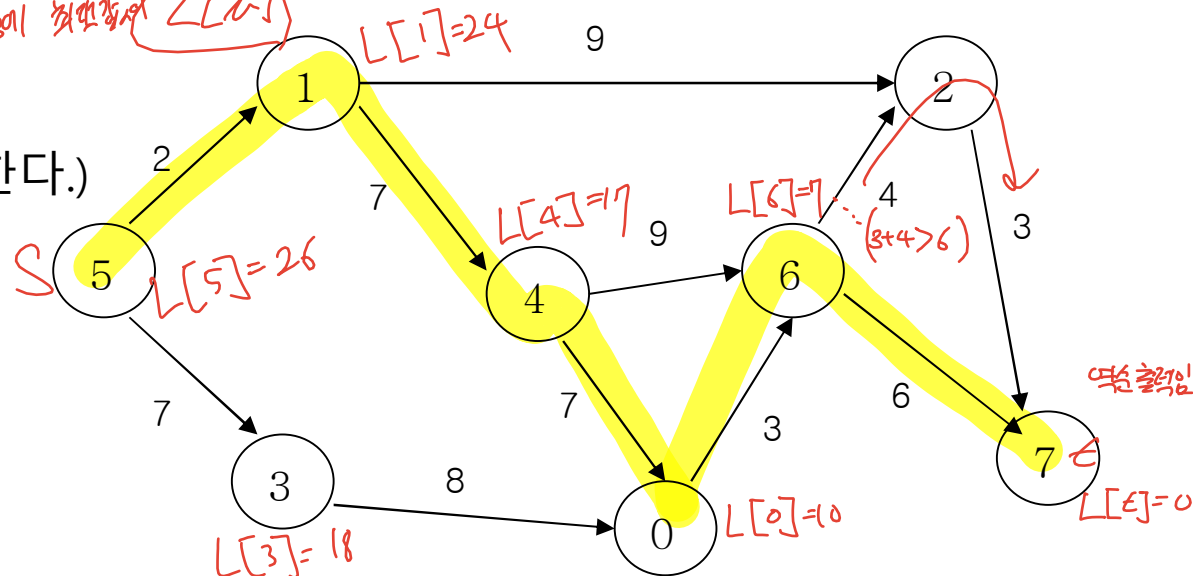
$Out(u)$ 에 있는 v를 모두 확인하고 생각.
각각에 최장경로 $L[v]$

$L[t] = 0$ 로 초기화 한다.

(나머지 정점들의 $L[]$ 은 $-\infty$ 로 초기화 한다.)

깊이우선탐색을 이용하여 구한다.

예: $s = 5, t = 7$



Weighted DAG에서 최장경로 문제

$$\{$$

```
{
    if (a > b)
        return a;
    else
        return b;
}
```

$$\}$$

```
void findLongestPath(Node** adjList, int n, int v, int t,
                    vector<bool> &visited, vector<int> &longestLength)
```

 $\{$

```
visited[v] = true; // v를 방문하였다고 표시함
```

```
if (v == t){
```

```
longestLength[v] = 0;
```

```
//      cout << v << " ";
return;
```

}

```
Node* ptr = adjList[v];
```

```
while (ptr != NULL){
```

```
int w = ptr->vertex;
```

```
if (not visited[w]){
```

```
findLongestPath(adjList, n, w, t, visited, longestLength);
```

```
longestLength[v] = maximum(longestLength[v], longestLength[w]+ ptr->weight);
```

}

else

```
longestLength[v] = maximum(longestLength[v], longestLength[w]+ ptr->weight);
```

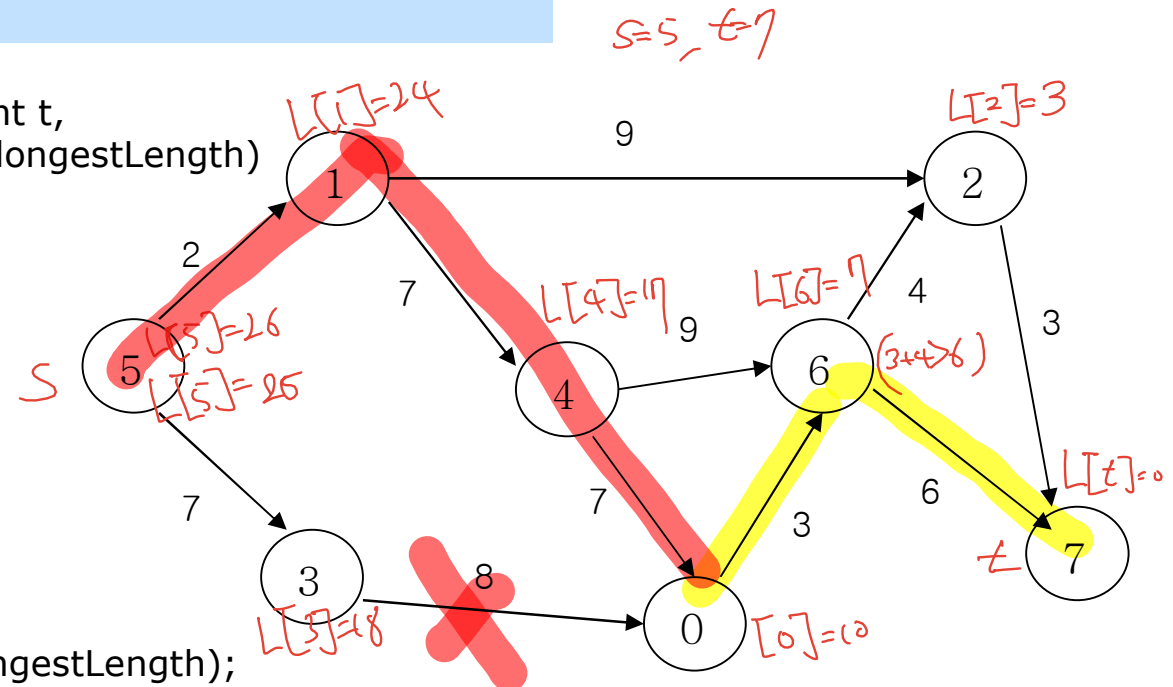
```
ptr = ptr->link;
```

}

```
// cout << v << " ";
```

}

```
struct Node {
    int vertex;
    int weight; //
    struct Node *link;
};
```

수행시간: $O(n+m)$