#고급반 2주차

DAG와 위상 정렬

<DAG, Topological Sort>

#DAG

<Directed Acyclic Graph>

Directed Acyclic Graph

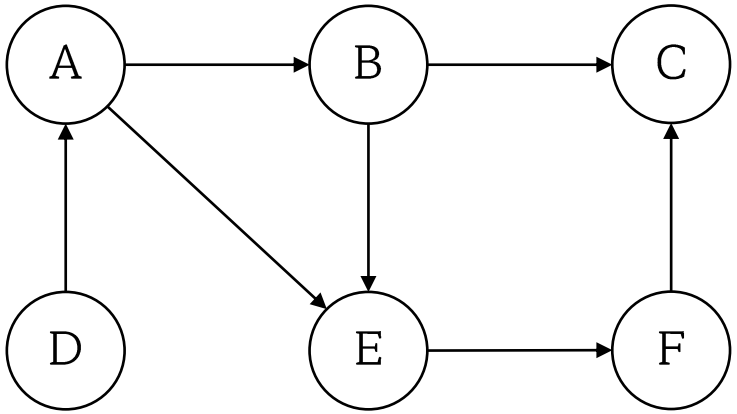
유향 사이클이 없는 그래프

1. 간선에 방향이 있고, 사이클이 없는 그래프를 의미합니다.
2. 한국어로는 '방향 비순환 그래프' 라고 하는 거 같은데, 보통 그냥 DAG라고 부릅니다.

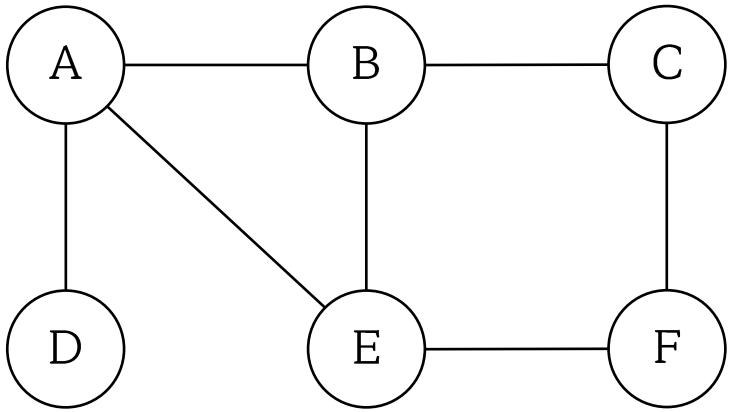

DAG
8
8
오후 7:46
뭐라고 읽음?
피자 먹다 자는 무지
디에이지
8
오후 7:47
튜브낀 튜브
대그
8
오후 7:50
경례하는 프로도
보통 대그라고 읽음
8
오후 7:51

#DAG

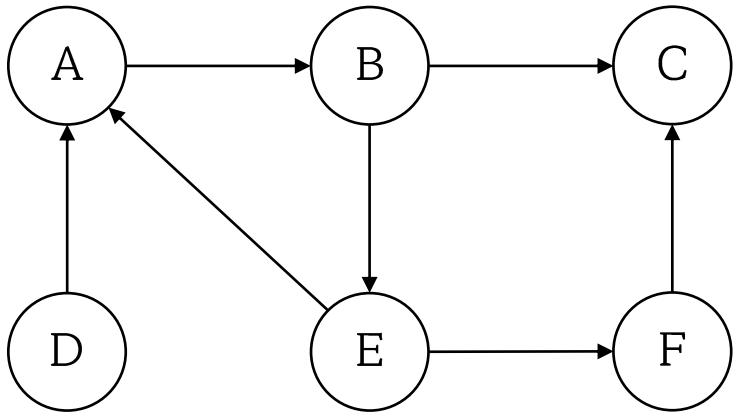
<Directed Acyclic Graph>



DAG



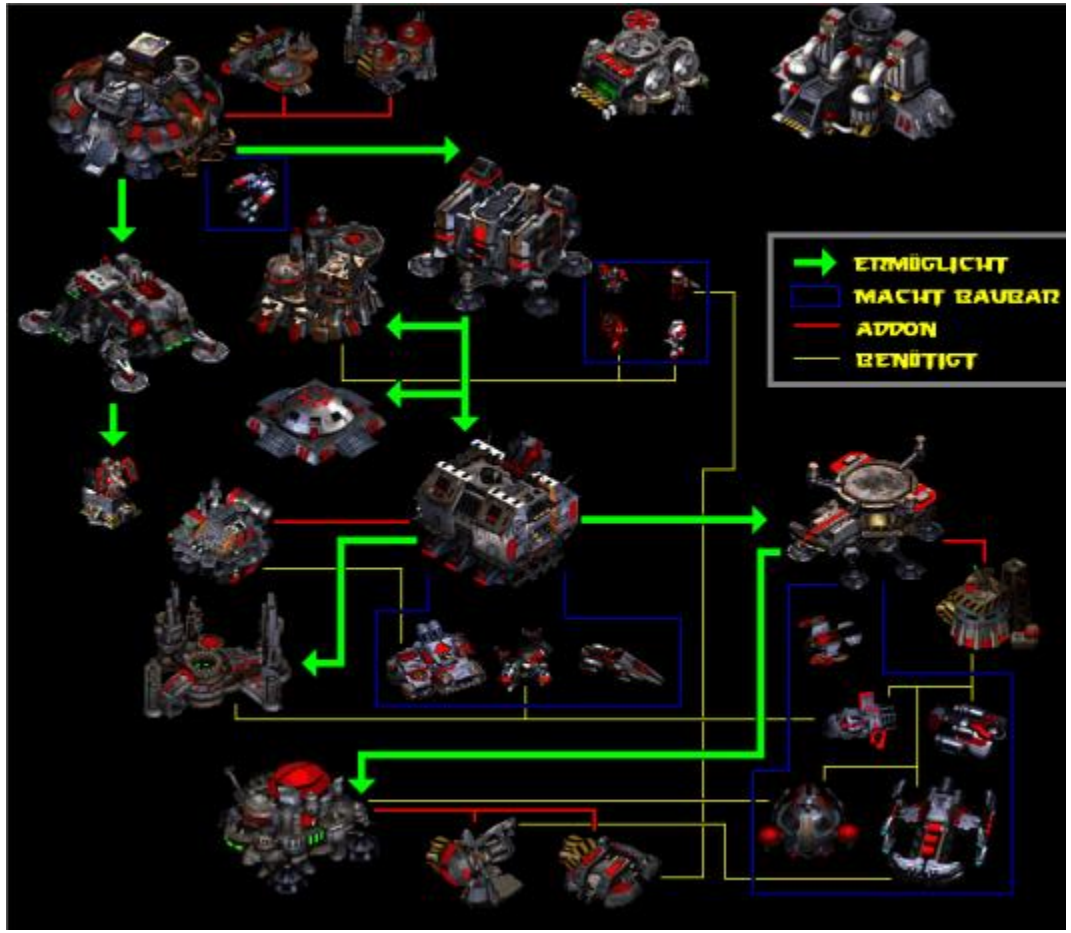
DAG X - 간선에 방향이 없음



DAG X – 사이클이 있음

#DAG

<Directed Acyclic Graph>

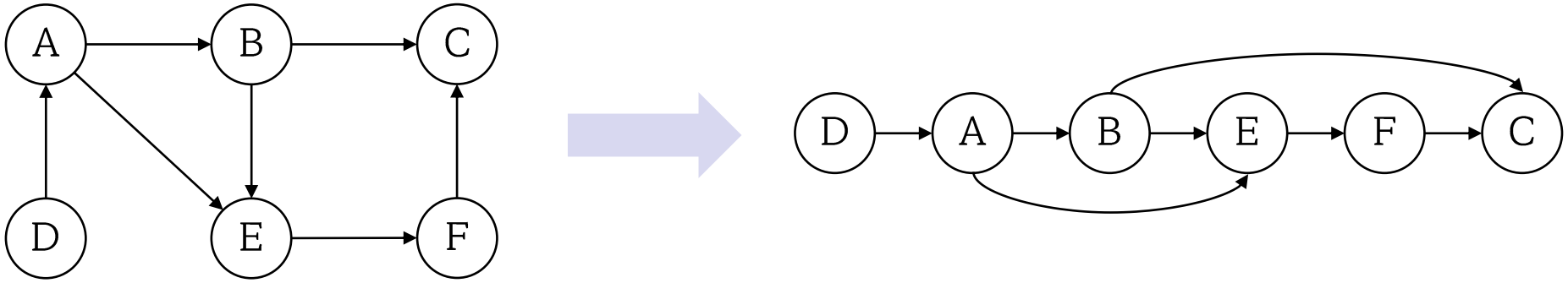


스타크래프트 건물 건설 순서는 DAG를 이룹니다.

#DAG

<Directed Acyclic Graph> - 위상 정렬

- DAG에서는 위상 정렬이 가능합니다!
  - 간선의 방향을 거스르지 않는 순서대로 정점을 나열하는 것을 말합니다.
  - u → v 간선이 있다면 u가 v보다 먼저 나와야 합니다.
  - 위상 정렬 순서대로 정점을 나열했을 때, 모든 간선은 오른쪽을 향하고 있겠죠.



#DAG

<Directed Acyclic Graph> - 구현 w/ BFS

위상 정렬을 구현하는 첫번째 방법으로, BFS를 응용할 수 있습니다.

- indegree가 0인 정점들을 큐에 삽입하고, 아래 과정을 반복합니다.
  1. 큐에서 정점 v를 pop
  2. v를 위상 정렬 결과에 append
  3. v에서 나가는 간선들을 제거, 연결된 정점들의 indegree 감소
  4. indegree가 0이 된 정점들을 큐에 삽입

만약 DAG가 아니라면, 모든 정점을 방문하지 않은 채로 위 과정이 종료됩니다.

위상 정렬 결과에서 가장 앞에 오는 정점의 indegree는 항상 0입니다. 만약 그렇지 않다면 다른 정점이 무조건 먼저 나왔어야 하겠죠. 따라서 indegree가 0인 정점을 위상 정렬 결과에 추가하고, 해당 정점을 삭제한 뒤 재귀적으로 반복하면 올바른 위상 정렬 결과를 얻을 수 있습니다.

#DAG

<Directed Acyclic Graph> - 구현 w/ BFS

`adj` : 인접 리스트로 나타낸 그래프

`ind` : 각 정점의 indegree

`res` : 위상 정렬 결과

만약 DAG가 아니라면 모든 정점을 방문하기 전에 BFS가 종료되므로, res의 크기가 N인지 확인해 주면 되겠죠.

시간복잡도 : O(|V| + |E|)

```
vector<int> topological_sort(int N, vector<vector<int>> adj) {
    vector<int> ind(N + 1, 0);
    for (auto i : adj) {
        for (auto j : i) {
            ind[j]++;
        }
    }
    queue<int> q;
    for (int i = 1; i <= N; i++) {
        if (ind[i] == 0) {
            q.push(i);
        }
    }
    vector<int> res;
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        res.push_back(v);
        for (auto i : adj[v]) {
            ind[i]--;
            if (ind[i] == 0) {
                q.push(i);
            }
        }
    }
    return res;
}
```

#DAG

<Directed Acyclic Graph> - 구현 w/ DFS

위상 정렬을 구현하는 두번째 방법으로, DFS를 응용할 수 있습니다.

- DFS를 진행하면서, 정점을 빠져나오는 순서대로 기록합니다.
- 기록된 순서의 역순이 위상 정렬 결과입니다.

왜 why? 간선 u → v 가 있을 때,

- 항상 v를 먼저 탈출합니다.
  - u를 먼저 방문한 경우, v를 탈출해야 u를 탈출할 수 있습니다.
  - v를 먼저 방문한 경우, v를 탈출해야 u를 방문할 수 있습니다.
    - 만약 v를 탈출하기 전에 u를 방문했다면 사이클이 존재합니다.

#DAG

<Directed Acyclic Graph> - 구현 w/ DFS

`adj` : 인접 리스트로 나타낸 그래프

`res` : 위상 정렬 결과

`vis` : 각 정점의 방문 상태

0 : 미방문 / 1 : 방문, 탈출 X / 2 : 탈출 완료

DFS 도중 `vis[now]` 가 1인 정점에 방문한다면 이는 DAG가 아님을 의미합니다.

모든 정점에 대해 DFS를 수행하고, 마지막에 res 배열을 뒤집으면 됩니다.

시간복잡도 : O(|V| + |E|)

```
vector<int> res;
int vis[100010];
void dfs(int now) {
    if (vis[now] == 1) {
        // cycle detected !!
        return;
    }
    if (vis[now] == 2) {
        return;
    }
    vis[now] = 1;
    for (auto i : adj[now]) {
        dfs(i);
    }
    res.push_back(now);
    vis[now] = 2;
}

int main() {
...
    for (int i = 1; i <= N; i++) {
        dfs(i);
    }
    reverse(res.begin(), res.end());
...
}
```

#DAG

<Directed Acyclic Graph> - 위상 정렬 연습문제

3 줄 세우기 (BOJ #2252)

<문제 설명>

- N명의 학생들을 키 순서대로 줄을 세우려고 한다.
- 두 학생의 키를 비교한 결과가 M개 주어진다.
- 가능한 키 순서의 결과를 아무거나 출력한다.

<제약 조건>

- $1 \leq N \leq 32,000$
- $1 \leq M \leq 100,000$

#DAG

<Directed Acyclic Graph> - 위상 정렬 연습문제

3 줄 세우기 (BOJ #2252)

<문제 해설>

위상 정렬의 조건 'u → v 간선이 있다면 u가 v보다 먼저 나온다'는 문제에서 이야기하는 조건과 정확히 동일한 조건입니다.

'학생 A가 학생 B보다 앞에 서야 한다' 는 조건이 주어지면 A → B 간선을 그래프에 추가합니다.

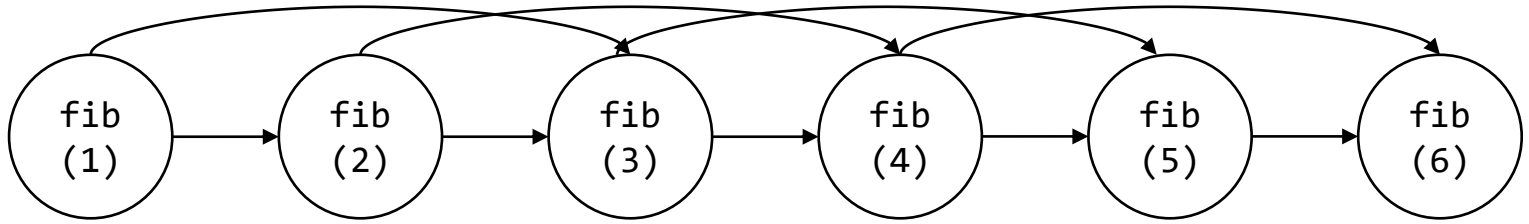
해당 그래프를 위상 정렬 한 결과를 출력하면 맞았습니다!! 를 받습니다.

#DAG

<Directed Acyclic Graph> - DAG와 DP

DAG와 DP는 밀접한 관련이 있습니다.

- DP는 큰 문제를 작은 부분 문제들로 나누어 해결한 다음, 이를 합치는 방법입니다.
- 이때 큰 문제와 작은 부분 문제 사이의 관계를 그래프로 나타내면, 이는 DAG로 표현됩니다.
- 따라서 해당 DAG의 위상 정렬 순서대로 해결하면, 올바른 순서로 부분 문제를 해결해 큰 문제의 답을 계산할 수 있겠죠.



피보나치 수 DP는 위와 같은 DAG로 나타낼 수 있습니다.

#DAG

<Directed Acyclic Graph> - DAG와 DP

이는 달리 말하면 DAG에 대한 문제는 DP를 활용해 해결할 수 있다는 뜻입니다.

그래프에 사이클이 없기 때문에, 위상 정렬 순서 상으로 답을 계산해 나가면 부분 문제들이 이미 계산되어 있기 때문에 항상 올바른 답을 얻을 수 있습니다.

- 예를 들어, 그래프 상의 최장 경로 문제는 일반적으로 NP-Hard지만, DAG에서는 DP를 이용해 풀 수 있습니다.

`dp[i] := i`에서 끝나는 최장 경로의 길이

로 정의하면, 오른쪽과 같이 답을 계산할 수 있습니다.

```
for (auto i : tpl_order) {
    for (auto [nxt, cost] : adj[i]) {
        dp[nxt] = max(dp[nxt], dp[i] + cost);
    }
}
```

#DAG

<Directed Acyclic Graph> - DAG와 DP 연습문제

ACM Craft (BOJ #1005)

<문제 설명>

- N개의 건물이 있고, i번 건물은 건설하는데 D_i 가 소요된다.
- '건물 X를 지어야 건물 Y를 지을 수 있다' 는 규칙 K개가 있다.
- 목표 건물의 번호 W가 주어질 때, W번 건물을 짓는 데 걸리는 최소 시간을 구해라.

<제약 조건>

- $1 \le N \le 1{,}000$
- $1 \le K \le 100{,}000$

#DAG

<Directed Acyclic Graph> - DAG와 DP 연습문제

ACM Craft (BOJ #1005)

<문제 해설>

주어진 건설 규칙을 그래프로 나타내면 DAG가 됩니다. '`dp[i] := i`번 건물을 짓는 최소시간' 으로 정의하면 아래와 같은 점화식이 성립합니다.

$$dp[i] = \begin{cases} D[i] & \text{(먼저 지어야 하는 건물이 없는 경우)} \\ \max(dp[j]) + D[i] & (j \to i \text{ 규칙이 있는 } j\text{에 대해}) \end{cases}$$

위상 정렬 순서대로 dp 배열을 계산하면 맞았습니다!! 를 받습니다.

#DAG

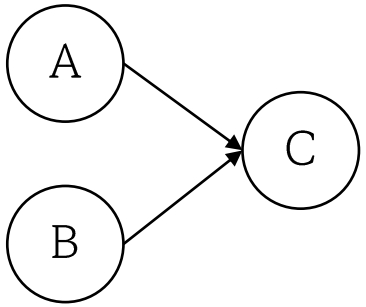
<Directed Acyclic Graph> - DAG와 DP

Tip

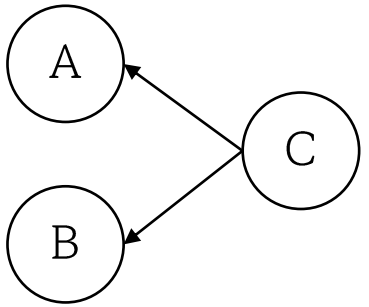
DAG 상에서 DP를 계산할 때, 명시적으로 위상 정렬을 할 필요가 없는 경우가 대부분입니다.

간선의 방향을 뒤집은 뒤, top-down 방식으로 문제를 해결하면 됩니다.

위상 정렬을 하는 이유는, 상태들의 올바른 계산 순서를 정하고 botton-up으로 구하기 위함입니다. 그런데 memoization과 함께 top-down 방식으로 코드를 작성하면 계산 순서를 신경 쓸 필요가 없겠죠.



A와 B를 먼저 계산하고
C를 계산

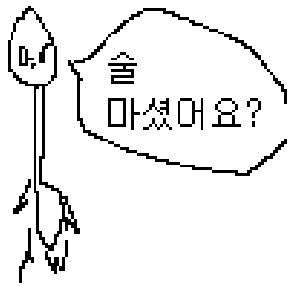


solve(C)를 호출하고, 뒤집힌 간선을 따라
solve(A), solve(B) 호출

#DAG

<Directed Acyclic Graph> - DAG와 DP

```
for (int i = 1; i <= N; i++) {
    for (auto j : edge[i]) {
        ind[j]++;
    }
}
queue<int> q;
for (int i = 1; i <= N; i++) {
    if (ind[i] == 0) {
        q.push(i);
    }
}
vector<int> res;
while (!q.empty()) {
    int v = q.front();
    q.pop();
    res.push_back(v);
    for (auto i : edge[v]) {
        ind[i]--;
        if (ind[i] == 0) {
            q.push(i);
        }
    }
}
for (auto i : res) {
    if (dp[i] == -1) {
        dp[i] = d[i];
    }
    for (auto j : edge[i]) {
        dp[j] = max(dp[j], dp[i] + d[j]);
    }
}
```




```
int solve(int x) {
    if (dp[x] != -1) {
        return dp[x];
    }
    int res = 0;
    for (auto i : edge[x]) {
        res = max(res, solve(i) + d[i]);
    }
    return dp[x] = res;
}
```

#연습 문제 도전

2 문제집 (BOJ #1766)

사전 순으로 가장 빠른 위상 정렬 결과는?

1 최종 순위 (BOJ #3665)

간선 정의를 어떻게 할까요?

2 24461 그래프의 줄기 (BOJ #24461)

HCPC 2021 기출문제.

5 임계경로 (BOJ #1948)

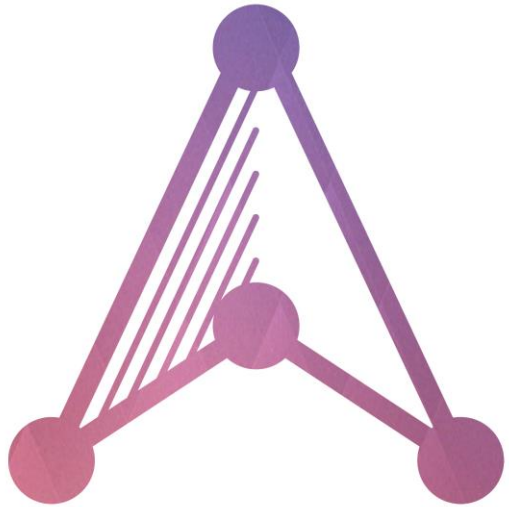
걸어서 세계 속으로

3 그래서 팩 주냐? (BOJ #17227)

돌게임 재밌나요?

5 Dumae (BOJ #16182)

자취를 하면 고기를 먹을 일이 잘 없는 것 같습니다.



ALOHA

The algorithm club.