Chapter 02.
위상 정렬

Clip 01 | [2252] 줄 세우기
위상 정렬 이론

Clip 02 | [2623] 음악프로그램
그래프 가공과 위상 정렬 예외처리

Clip 03 | [1005] ACM Craft
위상정렬과 DP

Clip 04 | [1766] 문제집
우선순위 큐를 이용한 위상정렬

Ch02. 위상 정렬

1. [2252] 줄 세우기

BOJ2252: 줄 세우기

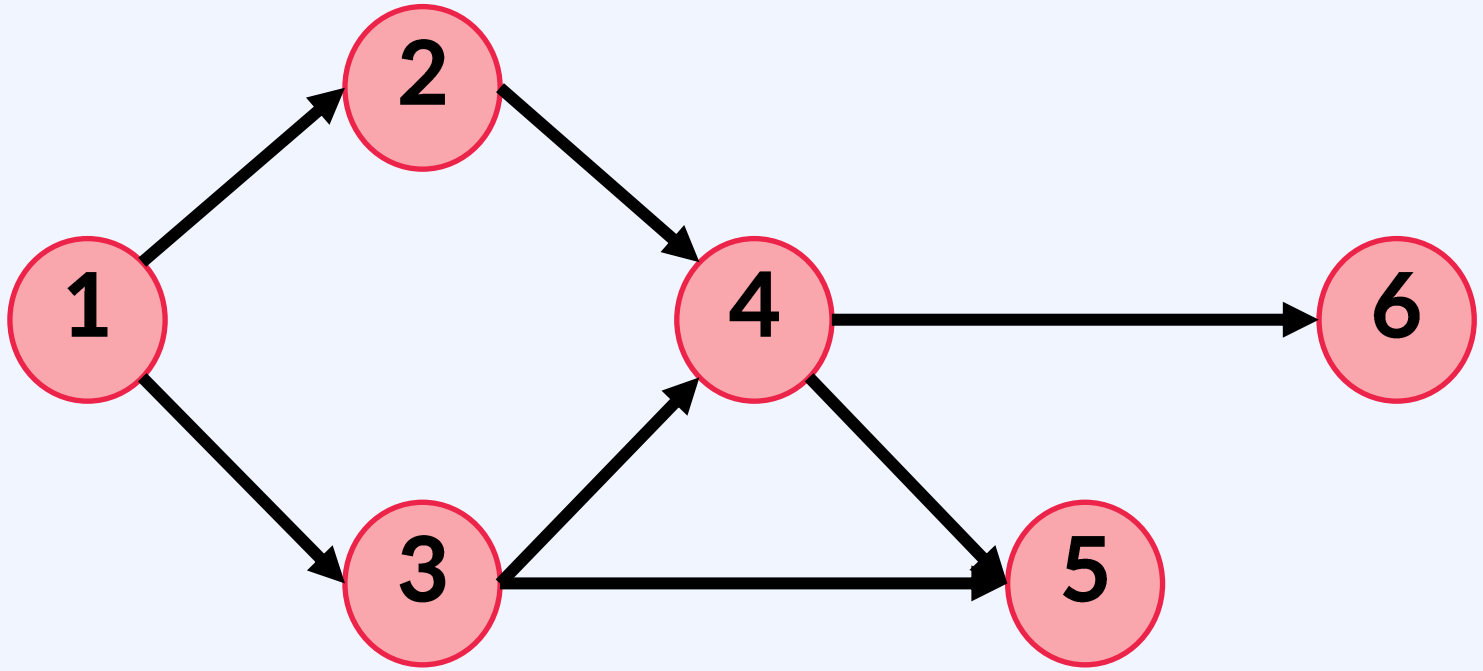
위상 정렬(Topological Sort)

- DAG에서 정점을 나열하는 알고리즘

- DAG?
  - Directed Acyclic Graph
  - 방향이 있고, 사이클이 없는 그래프

BOJ2252: 줄 세우기

위상 정렬(Topological Sort)



- DAG?
  - Directed Acyclic Graph
  - 방향이 있고, 사이클이 없는 그래프

BOJ2252: 줄 세우기

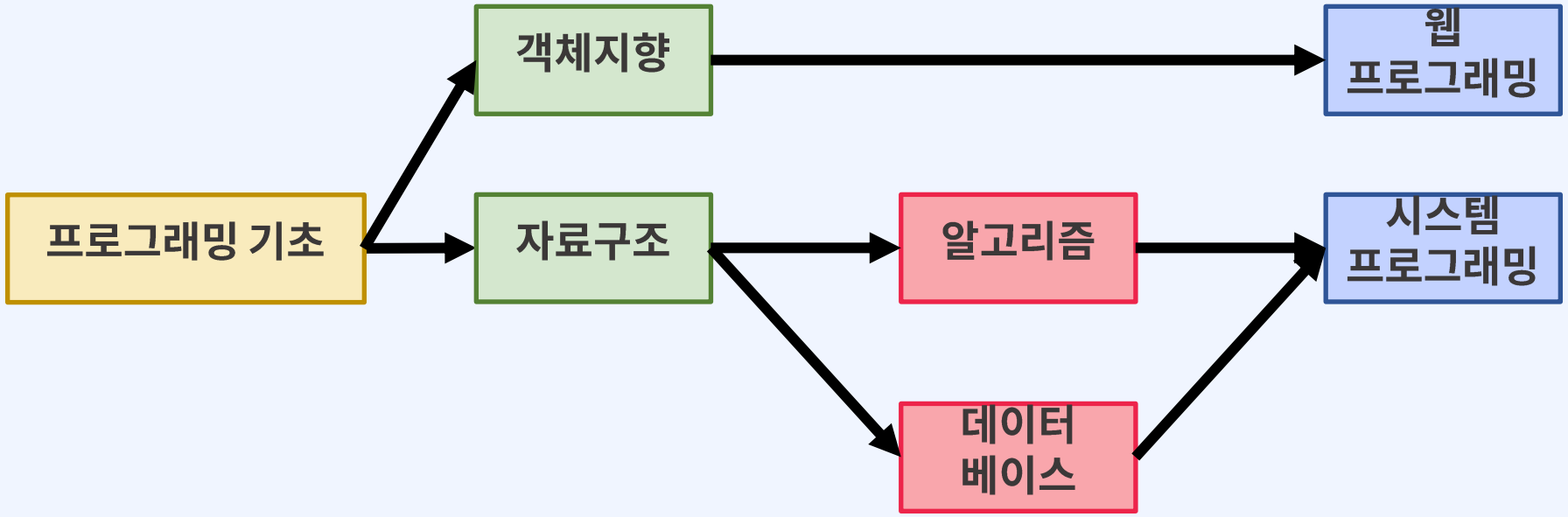
위상 정렬(Topological Sort)

- DAG에서 정점을 나열하는 알고리즘
- 노드간 선 후 관계를 나타내기 위해 사용함
  - ex. 작업 공정, 스케줄링,
    - B 작업은 A작업이 선행되어야 진행할 수 있다
- 사이클이 존재하는 그래프는 위상 정렬을 할 수 없다

BOJ2252: 줄 세우기

위상 정렬(Topological Sort)

ex. 선 이수 과목



알고리즘은 자료구조를 수강한 뒤 들을 수 있다

BOJ2252: 줄 세우기

위상 정렬(Topological Sort)

구현 방법

1. 진입 차수가 0인 정점 선택
2. 선택된 정점을 위상 정렬된 대상으로 출력
3. 해당 정점과 연결된 간선을 제거 (진입 차수를 감소)
4. 모든 노드를 다 정렬할 때 까지 (1) ~ (3)을 반복

BOJ2252: 줄 세우기

문제 요약

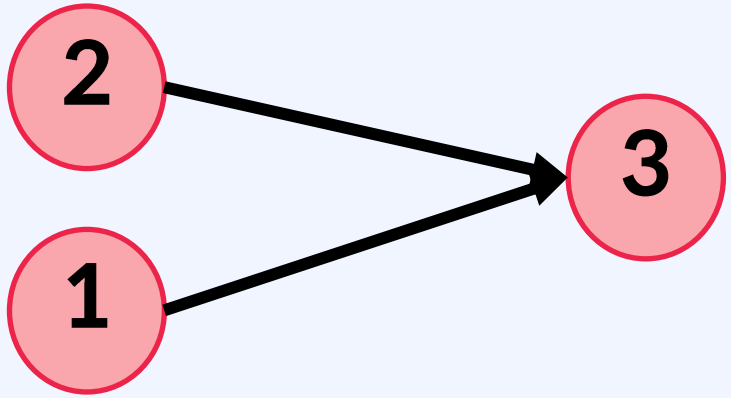
- 학생들을 키 순서대로 줄을 세움
- 일부 학생들간 대소 관계만 주어질 때,
  키 순서대로 줄을 세우기
- 학생 수 N : (1 <= N < = 32,000)
- 대소 관계 수: (1 <= M <= 100,000)

BOJ2252: 줄 세우기

문제 요약

입력 데이터
3 2 1 3 2 3

출력 데이터
1 2 3



BOJ2252: 줄 세우기

문제 분석

- [학생]: 정점
- [앞 / 뒤로 줄 서는 관계]: 간선

- 학생들을 줄세우는 위상정렬 문제가 된다

- 이 연결관계는 DAG를 만족할까?
  - 학생들의 관계가 {앞} – {뒤} 로 구분된다
  - A가 B의 뒤에 있으면서 앞에 있는 경우는 존재하지 않는다

BOJ2252: 줄 세우기

문제 분석

- 정점과 간선을 연결리스트로 만들면서
  진입 차수(Indegree)를 계산한다
- 진입 차수가 0인 정점을 큐에 넣고, 연결된 정점들
  의 진입 차수를 1 감소시킨다
- 다시 진입 차수가 0인 정점을 찾아서 반복한다

BOJ2252: 줄 세우기

구현

```
List<Integer>[] list = new List[n + 1];
for(int i = 1; i <= n; i++) {
    list[i] = new ArrayList<>();
}
int[] indegree = new int[n + 1];
int[] check = new int[n + 1];
for(int i = 0; i < m; i++) {
    int a = sc.nextInt();
    int b = sc.nextInt();
    list[a].add(b);
    indegree[b]++;
}
```

인접리스트 초기화

학생들의 연결관계 입력

인접 리스트에 추가

진입 차수 1 증가

BOJ2252: 줄 세우기

구현

```
Queue<Integer> q = new LinkedList<>();
for(int i = 1; i <= n; i++) {
    if(indegree[i] == 0) q.offer(i);
}
```

진입 차수가 0 인 정점을 찾아서 큐에 넣기

BOJ2252: 줄 세우기

구현

```
while(!q.isEmpty()) {
    int now = q.poll();
    check[now] = 1;
    System.out.print(now + " ");
    for(int next : list[now]) {
        if(check[next] == 1) continue;
        indegree[next]--;
        if(indegree[next] == 0) q.offer(next);
    }
}
```

큐에서 뽑고, 데이터 체크

이미 출력한 데이터는 스킵

다음 방문 노드 진입 차수 감소

진입차수가 0이된
노드 큐에 추가

Ch02. 위상 정렬

2. [2623] 음악 프로그램

BOJ2623: 음악 프로그램

문제 요약

- 방송 출연할 가수의 순서를 정하기
- PD 마다 정해 둔 출연 순서가 있다
- 모든 PD를 만족하는 출연이 가능하면 순서를 출력
- 불가능하다면 0을 출력

BOJ2623: 음악 프로그램

문제 분석

- 보조PD의 리스트를 하나의 그래프로 가공해야 한다

- PD가 정한 A → B → C → D의 순서가 있다고 하면 아래와 같이 간선 분리가 가능하다
  - A → B
  - B → C
  - C → D

BOJ2623: 음악 프로그램

문제 분석

- 위상 정렬을 할 수 없는 경우?
  - 위상 정렬이 성립하려면 그래프가 DAG로 주어져야 한다 (방향성이 있고, 사이클이 없는 그래프)
- 이 문제에서는 출연 순서에 대한 간선의 방향성은 정해져 있다
- 그렇다면, 사이클이 발생하는 케이스가 있는지 찾으면 된다

BOJ2623: 음악 프로그램

문제 분석

- 사이클을 찾는 방법?
  - 위상 정렬은 순서상에 모든 노드가 1번씩 등장해야 한다
  - 만약 사이클이 있다면, 특정 노드가 2번 이상 등장한다

- 위상 정렬을 하는데 N개의 정점을 모두 출력하였는데,
  여전히 큐에 정렬 대상이 남아 있다면?
  - 적어도 1개 이상의 정점에서 사이클이 발생했다
    (비둘기 집의 원리)

BOJ2623: 음악 프로그램

문제 분석

- 사이클을 찾는 방법?
  - 출력하는 정점의 개수를 센다
  - N개의 정점을 출력했는데, 여전히 큐에 정점이 남으면 사이클이다

- 정점이 N개라고 무조건 사이클이 아니라는 보장도 없다
  - 정점 정렬 여부를 기록하는 check[] 에 모두 표시가 되었는지 검증도 진행해야 한다

BOJ2623: 음악 프로그램

구현

```
List<Integer>[] list = new ArrayList[n + 1];
for (int i = 1; i <= n; i++) {
    list[i] = new ArrayList<>();
}
Queue<Integer> q = new LinkedList<>();
int[] indegree = new int[n + 1];
int[] check = new int[n + 1];
```

정점의 개수를 이용한 공간 초기화

BOJ2623: 음악 프로그램

구현

```
for (int i = 0; i < m; i++) {
    int cnt = sc.nextInt();
    int front = sc.nextInt();

    for (int j = 2; j <= cnt; j++) {
        int back = sc.nextInt();
        list[front].add(back);
        indegree[back]++;
        front = back;
    }
}
```

각 PD별 뽑은 출연자 수

첫번째로 방송하는 출연자

이후의 입력을 받으며
연결관계를 순환한다
진입차수도 함께 기록한다

BOJ2623: 음악 프로그램

구현

```
while (!q.isEmpty()) {
    int now = q.poll();
    check[now] = 1;
    answer.add(now);
    if (answer.size() > n) {
        System.out.println(0);
        return;
    }
    for (int next : list[now]) {
        if (check[next] == 1) continue;
        indegree[next]--;
        if (indegree[next] == 0) q.offer(next);
    }
}
```

방송에 출연하였으면 체크한다

만약 N번 이상 정답이 나온다면
누군가 2번 이상 출연-> 사이클

출연했다면 스킵

진입 차수 감소와
다음 탐색 준비

BOJ2623: 음악 프로그램

구현

```
for(int i = 1; i <= n; i++) {
    if(check[i] == 0) {
        System.out.println(0);
        return;
    }
}
```

모든 사람이 출연했는지 검사
아니라면 0 을 출력

Ch02. 위상 정렬

3. [1005] ACM Craft

BOJ1005: ACM Craft

문제 요약

- 정해진 순서에 따라 건물을 건설
- 단 건물별로 건설에 일정 시간이 소요된다
- 특정 건물을 가장 빠르게 짓기 위한 최소 시간을 구하는 문제
- 건물의 개수 N : (2 <= N <= 1,000)
- 건물의 건설 순서 K : (1 <= K <= 100,000)

BOJ1005: ACM Craft

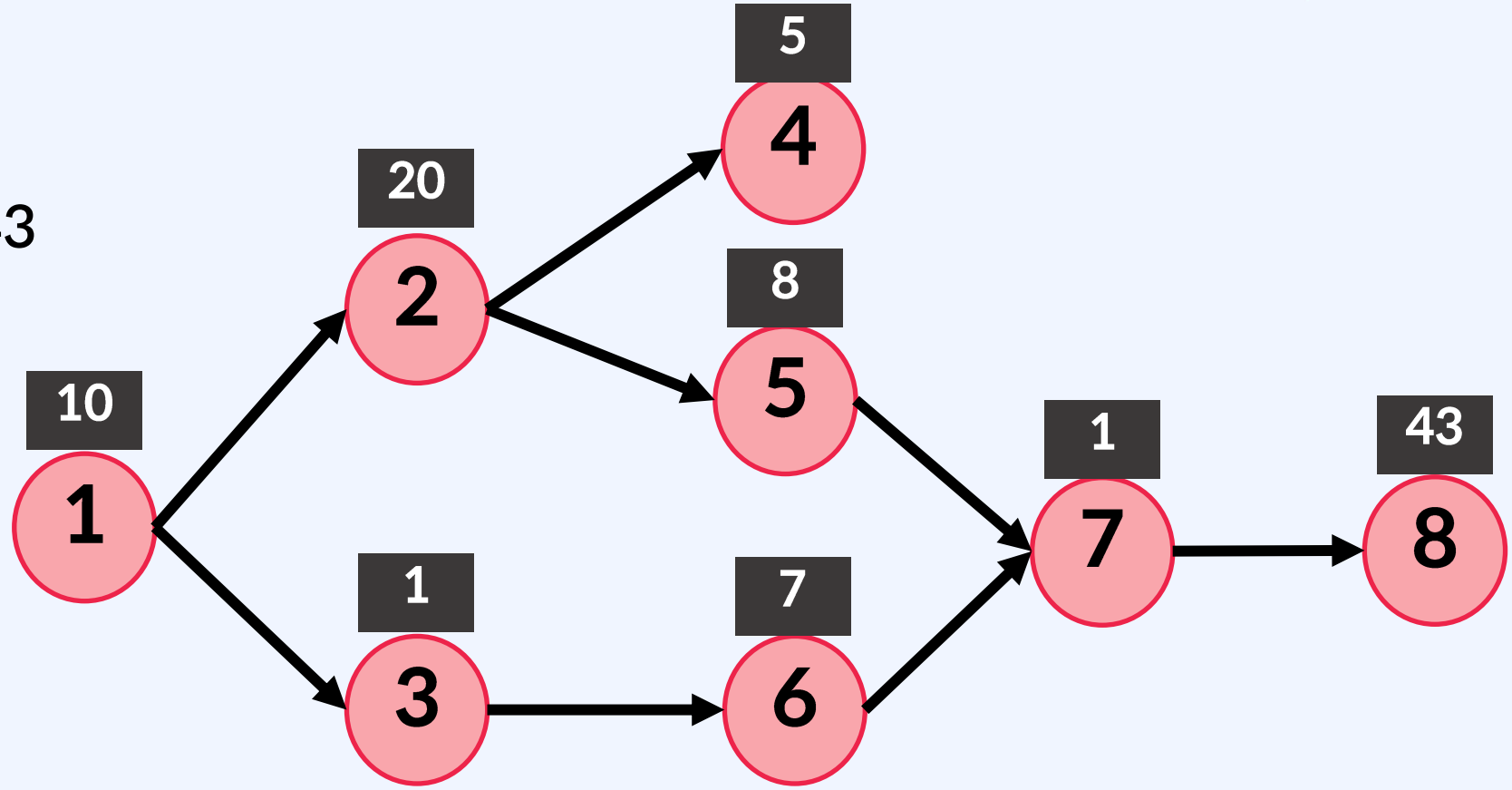
문제 분석

- {i} 번째 건물이 완성되는데 걸리는 시간은?
  - max(먼저 지어야 하는 건물의 완성 시간) + time[i]
- DP와 위상 정렬을 동시에 수행할 필요가 있다

BOJ1005: ACM Craft

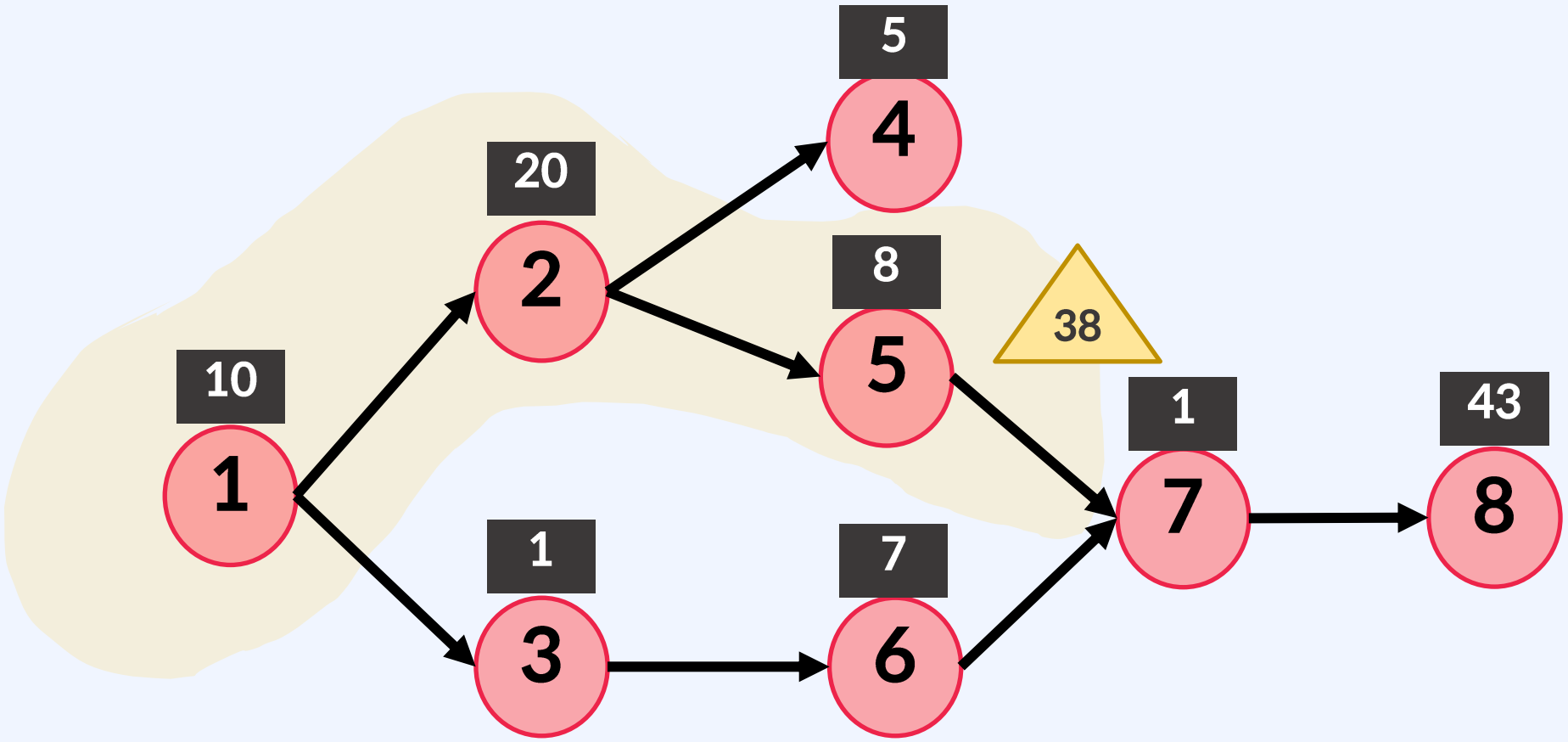
문제 분석

```
8 8
10 20 1 5 8 7 1 43
1 2
1 3
2 4
2 5
3 6
5 7
6 7
7 8
```



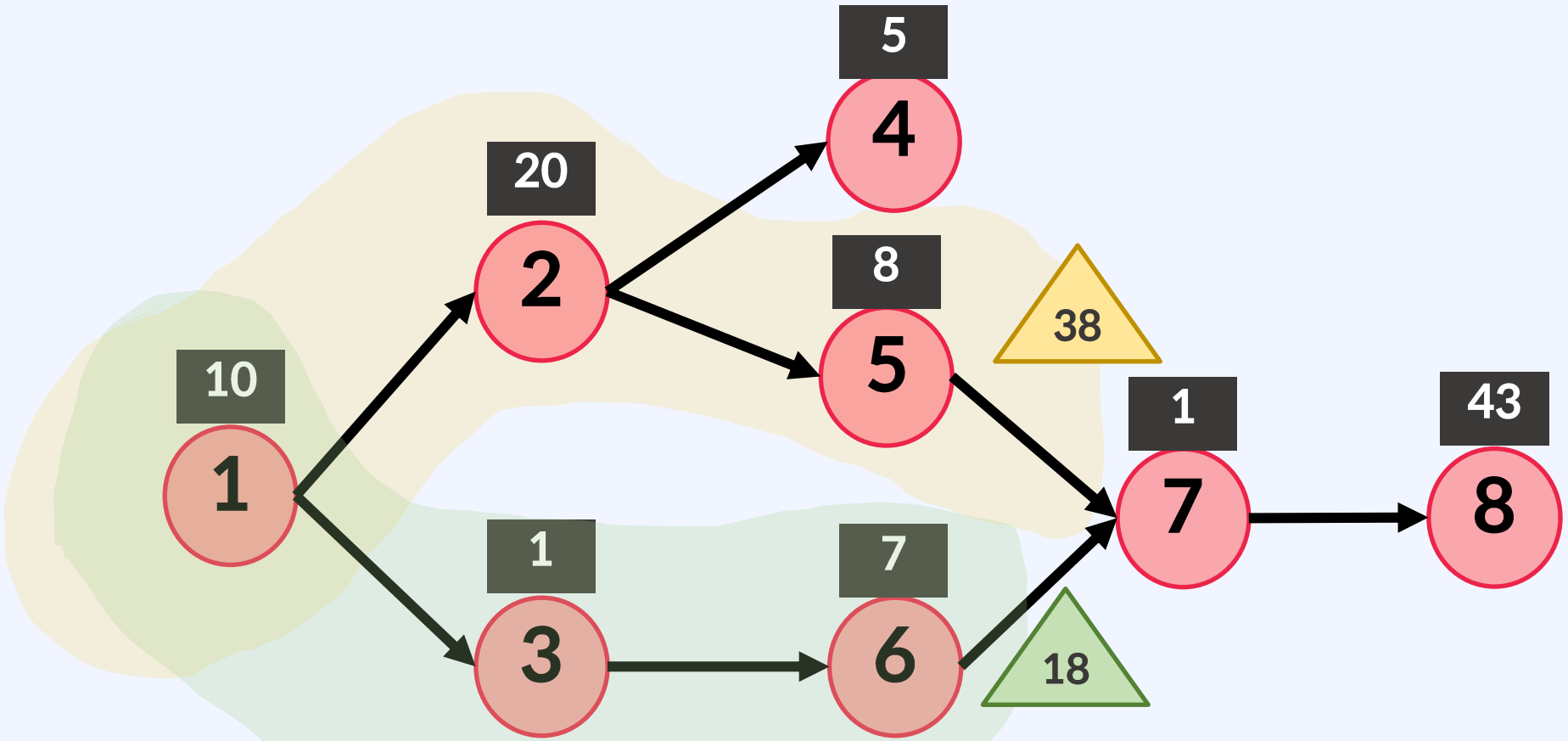
BOJ1005: ACM Craft

문제 분석



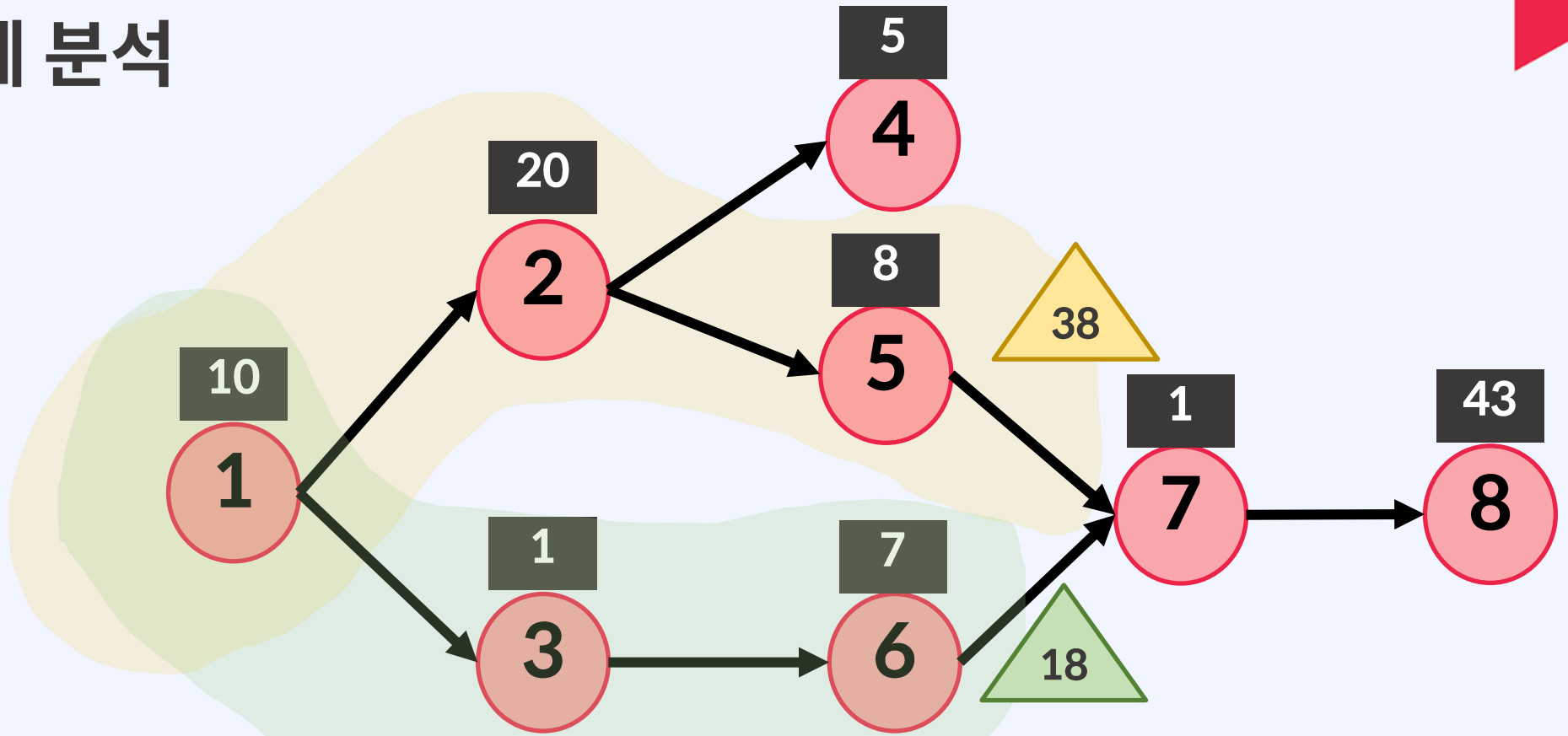
BOJ1005: ACM Craft

문제 분석



BOJ1005: ACM Craft

문제 분석



7을 짓기 위해서는 초록색 / 노란색 영역의 건설이 모두 끝나야 한다
즉, 더 큰 시간 값만큼 기다려야 한다

BOJ1005: ACM Craft

문제 분석

- 문제 요구사항: 특정 건물을 짓는데 걸리는 시간
  - dp[next]: next번째 건물이 완성되는데 필요한 시간

- dp[next] = max(dp[now]) + time[i]
  - now 까지의 건물이 완성되는데 걸리는 시간 +
    {i}번째 건물이 완성되는데 걸리는 시간

- next와 now의 값은? → 위상정렬을 통해 획득한다

BOJ1005: ACM Craft

문제 분석

- now
  - indegree가 0이라 큐에 들어갔다가
    뽑힌 건물의 번호

- next
  - now와 연결된 다른 건물들 (그래프에서 획득)

BOJ1005: ACM Craft

구현

```
for (int i = 0; i < k; i++) {
    int a = sc.nextInt();
    int b = sc.nextInt();
    adj[a][b] = 1;
    indegree[b]++;
}
```

인접행렬과 진입차수 처리

```
Queue<Integer> q = new LinkedList<>();
for (int i = 1; i <= n; i++) {
    if (indegree[i] == 0) {
        q.offer(i);
        dp[i] = time[i];
    }
}
```

진입차수가 0인 건물 큐와 시간 배열에 추가

BOJ1005: ACM Craft

구현

```
while (!q.isEmpty()) {
    int now = q.poll();
    for (int next = 1; next <= n; next++) {
        if (adj[now][next] == 1) {
            indegree[next]--;
            dp[next] = Math.max(dp[next], dp[now] + time[next]);
            if (indegree[next] == 0) q.offer(next);
        }
    }
}
```

건물을 큐에서 뽑아, 연결된 그래프에 최대시간을 업데이트

Ch02. 위상 정렬

4. [1766] 문제집

BOJ1766: 문제집

문제 요약

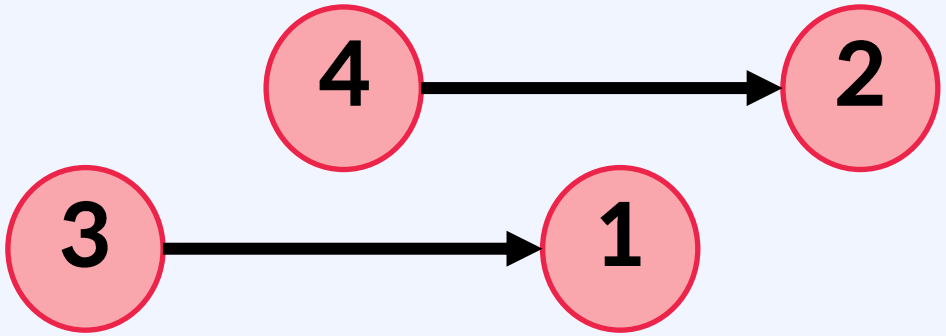
- 1번부터 N번까지의 문제를 풀이
  - (1 <= N <= 32,000)
- 규칙
  - N개의 문제를 모두 풀어야한다
  - 특정 문제는 먼저 풀이 해야 할 문제가 있다
    - 순서쌍 정보 (1 <= M <= 100,00)
  - 가능한 쉬운 문제부터 풀어야한다
    - 문제번호가 낮을수록 쉬운 문제이다

BOJ1766: 문제집

문제 요약

입력 데이터
4 2 4 2 3 1

출력 데이터
3 1 4 2



BOJ1766: 문제집

문제 분석

- 일반적인 위상 정렬 문제일 줄 알았지만
  가능한 쉬운 문제부터 풀어야하는 조건이 있다

- 풀이가 가능한 문제는 큐에 담겨있다
  - 이 중에서 번호가 낮은 문제부터 뽑아야 한다
  - 우선순위 큐?

BOJ1766: 문제집

문제 분석

- 위상 정렬을 관리하는 큐를 최소 힙으로 구현하면 항상 최소값이 먼저 나오는 자료구조의 특성으로

  낮은 번호의 문제부터 풀이할 수 있다

BOJ1766: 문제집

구현

```
PriorityQueue<Integer> pq = new PriorityQueue<>();
for(int i = 1; i <= n; i++) {
    if(indegree[i] == 0) pq.offer(i);
}
while(!pq.isEmpty()) {
    int now = pq.poll();
    System.out.print(now + " ");
    for(int next : list[now]) {
        indegree[next]--;
        if(indegree[next] == 0) pq.offer(next);
    }
}
```

우선순위 큐를 정의하고
indegree가 0인 문제부터 넣는다
(먼저 풀어야하는 의존성이 없다)

연결 관계가 있는 문제는
리스트를 순회하며 우선순위 큐에 추가