Chapter. 07

고급 탐색 알고리즘

핵심 유형 문제풀이

FAST CAMPUS
ONLINE
유형별 문제풀이

강사. 나동빈

Chapter. 07

고급 탐색 알고리즘(핵심 유형 문제풀이)

FAST CAMPUS
ONLINE
나동빈 강사.

I 혼자 힘으로 풀어 보기

최소 힙: https://www.acmicpc.net/problem/1927

문제 제목: 최소 힙

문제 난이도: 하(Easy)

문제 유형: 힙, 자료구조

추천 풀이 시간: 20분

최소 힙: https://www.acmicpc.net/problem/1927

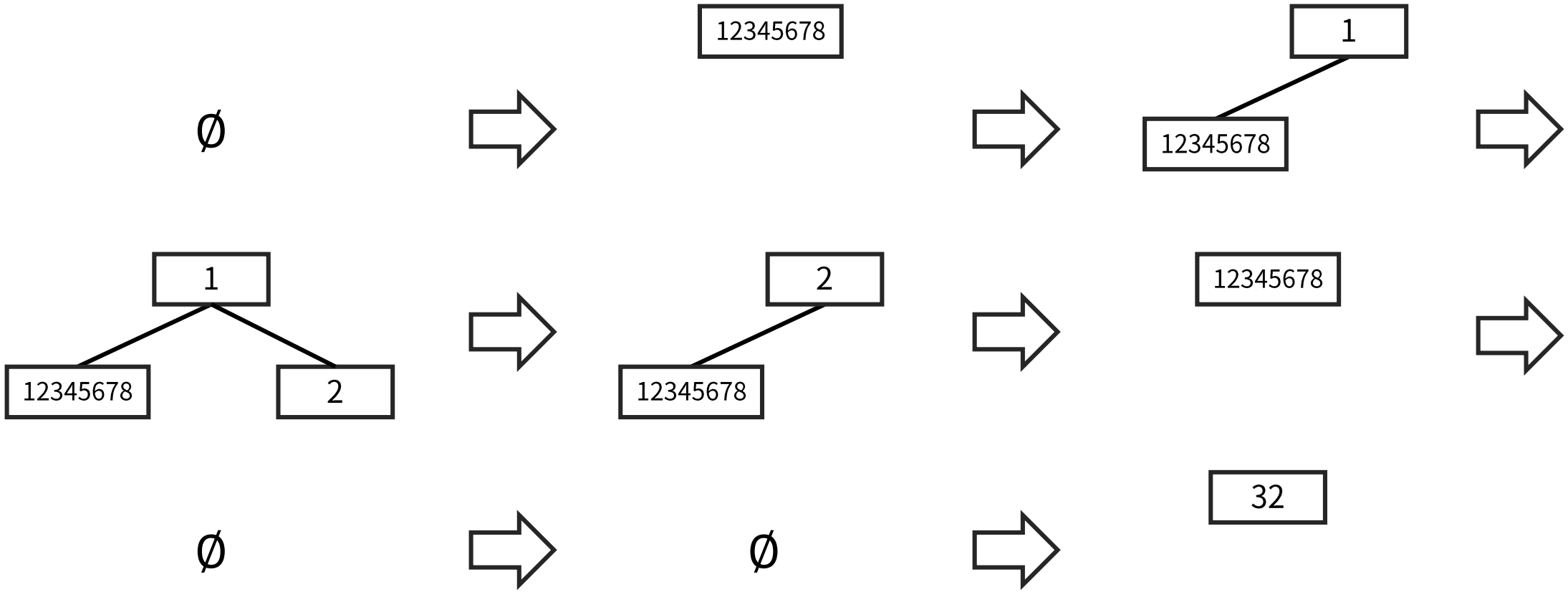
I 문제 풀이 핵심 아이디어

- 최소 힙의 기본적인 기능을 구현합니다.
- 파이썬에서 **heapq 라이브러리**를 이용하면 간단히 힙을 구현할 수 있습니다.

최소 힙: https://www.acmicpc.net/problem/1927

문제 풀이 핵심 아이디어

- 최소 힙의 기본적인 기능을 구현합니다.



입력: 0 12345678 1 2 0 0 0 0 32

출력: 0 1 2 12345678 0

최소 힙: https://www.acmicpc.net/problem/1927

| 소스코드

```python
import heapq

n = int(input())
heap = []
result = []

for _ in range(n):
    data = int(input())
    if data == 0:
        if heap:
            result.append(heapq.heappop(heap))
        else:
            result.append(0)
    else:
        heapq.heappush(heap, data)

for data in result:
    print(data)
```

I 혼자 힘으로 풀어 보기

카드 정렬하기: https://www.acmicpc.net/problem/1715

문제 제목: 카드 정렬하기

문제 난이도: 하(Easy)

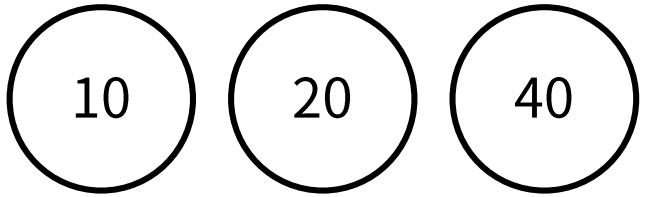
문제 유형: 힙, 자료구조, 그리디

추천 풀이 시간: 20분

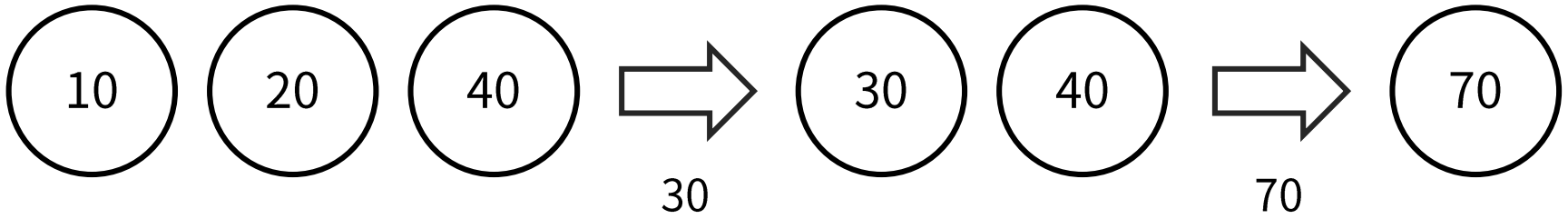
카드 정렬하기: https://www.acmicpc.net/problem/1715

I 문제 풀이 핵심 아이디어

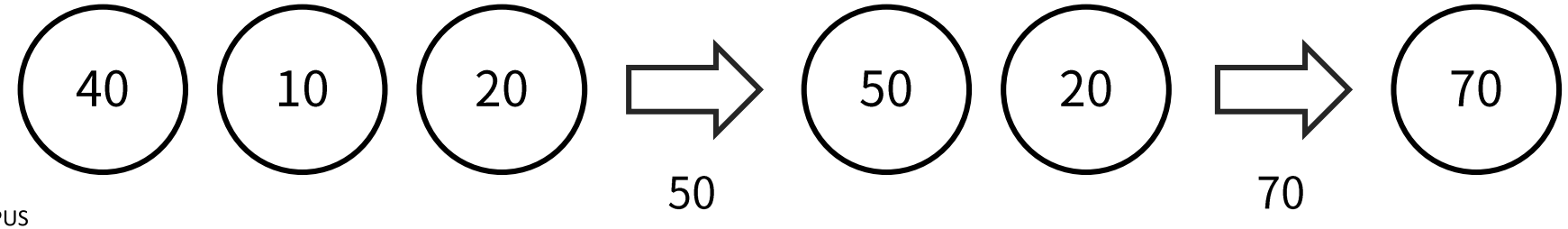
- 가장 크기가 작은 숫자 카드 묶음들을 먼저 합쳤을 때, 비교 횟수가 가장 적습니다.



[예시 1] 총 비교 횟수: **100**



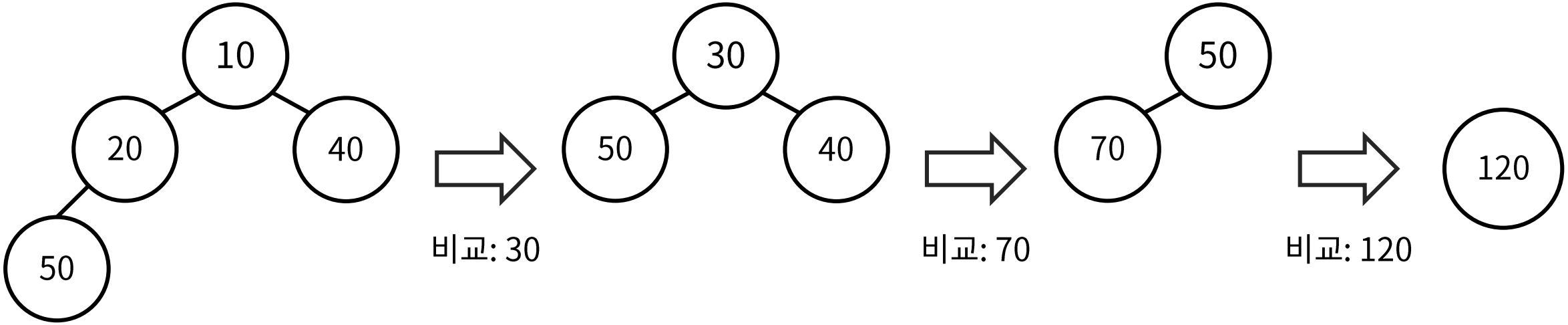
[예시 1] 총 비교 횟수: **120**



카드 정렬하기: https://www.acmicpc.net/problem/1715

문제 풀이 핵심 아이디어

- 가장 크기가 작은 숫자 카드 묶음들을 2개씩 합치기 위해, 힙 자료구조를 이용합니다.



최소 비교 횟수: 220

| 소스코드

카드 정렬하기: https://www.acmicpc.net/problem/1715

```python
import heapq

n = int(input())
heap = []

for i in range(n):
    data = int(input())
    heapq.heappush(heap, data)

result = 0

while len(heap) != 1:
    one = heapq.heappop(heap)
    two = heapq.heappop(heap)
    sum_value = one + two
    result += sum_value
    heapq.heappush(heap, sum_value)

print(result)
```

I 혼자 힘으로 풀어 보기

문제집: https://www.acmicpc.net/problem/1766

문제 제목: 문제집

문제 난이도: 중(Medium)

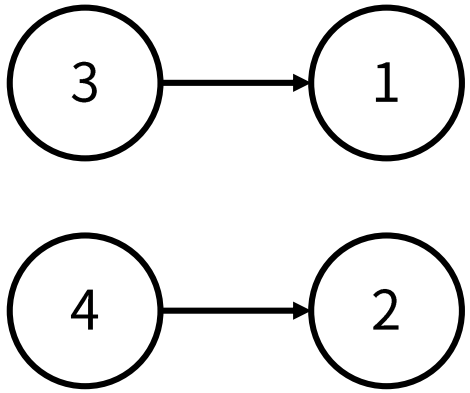
문제 유형: 힙, 위상 정렬

추천 풀이 시간: 40분

문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

- 본 문제는 전형적인 위상 정렬 문제입니다.
- 위상 정렬은 **순서가 정해져 있는 작업**을 차례로 수행해야 할 때, 순서를 결정해주는 알고리즘입니다.
- 위상 정렬의 시간 복잡도는 $O(V+E)$로 문제를 해결할 수 있습니다.



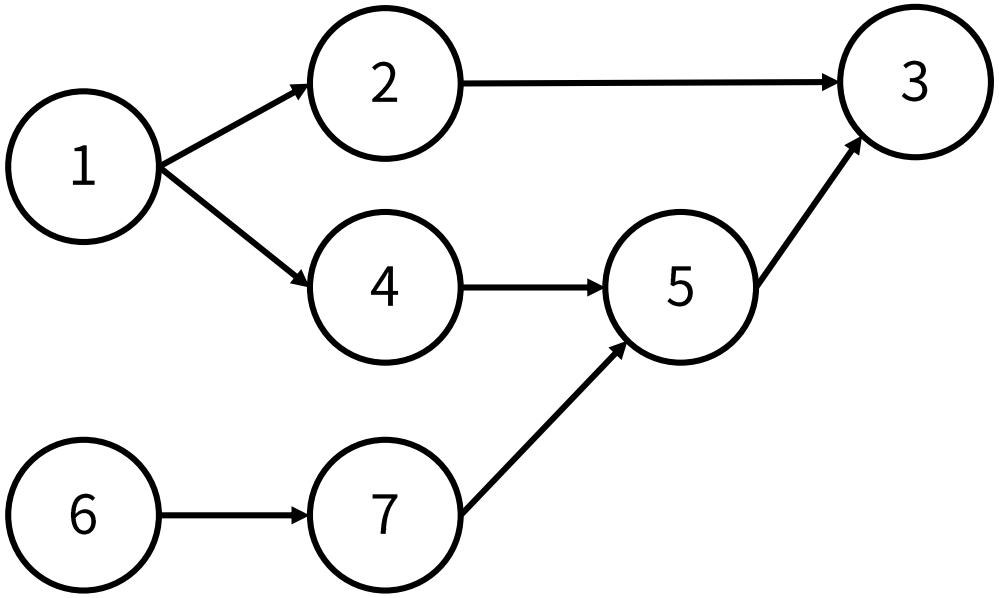
정답 예시: 3 – 1 – 4 – 2

문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.



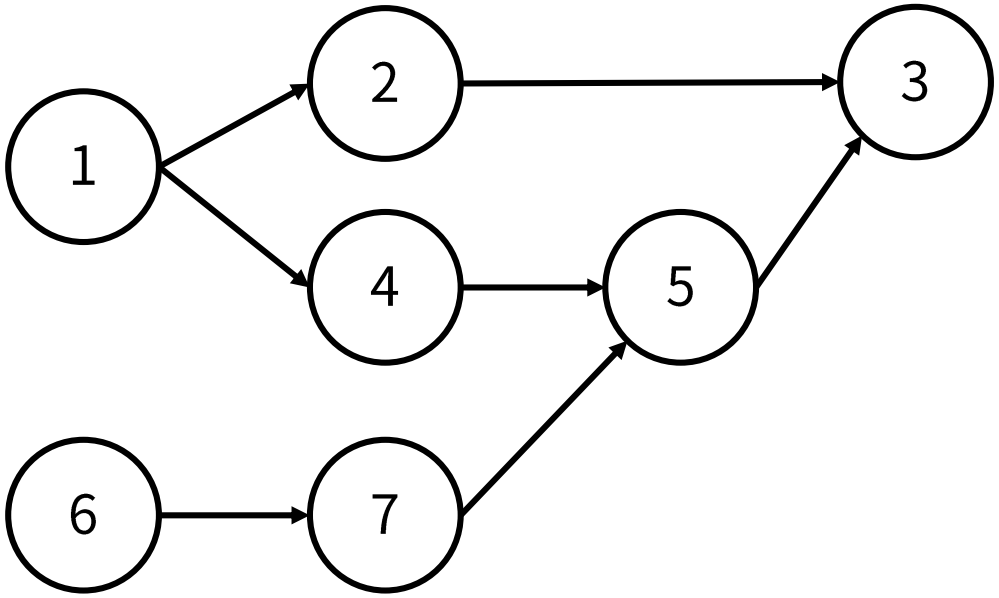
- 모든 원소를 방문하기 전에 큐가 빈다면 사이클이 존재하는 것입니다.
- 모든 원소를 방문했다면 큐에서 꺼낸 순서가 위상 정렬의 결과입니다.

문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.

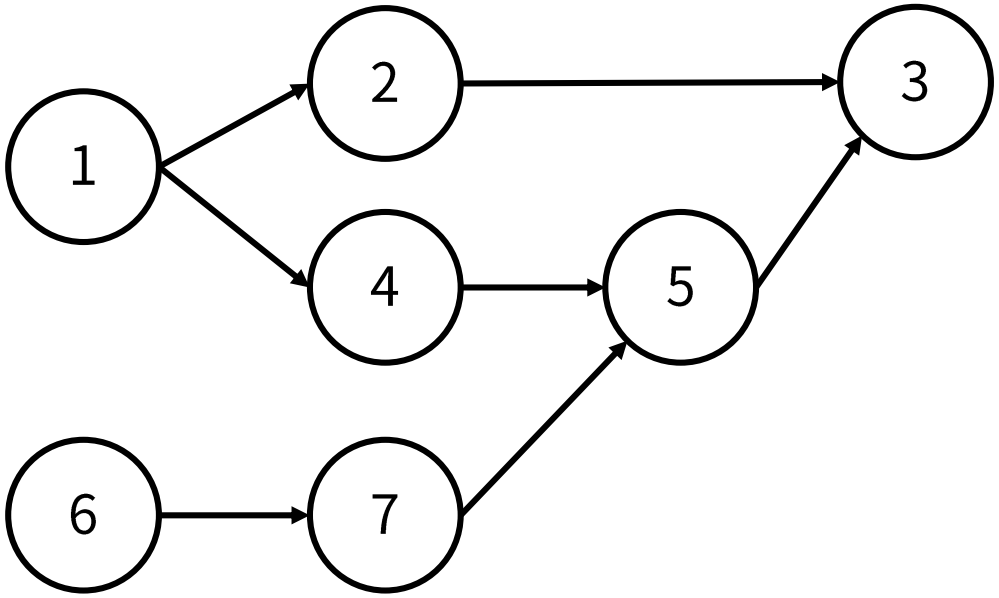


문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.



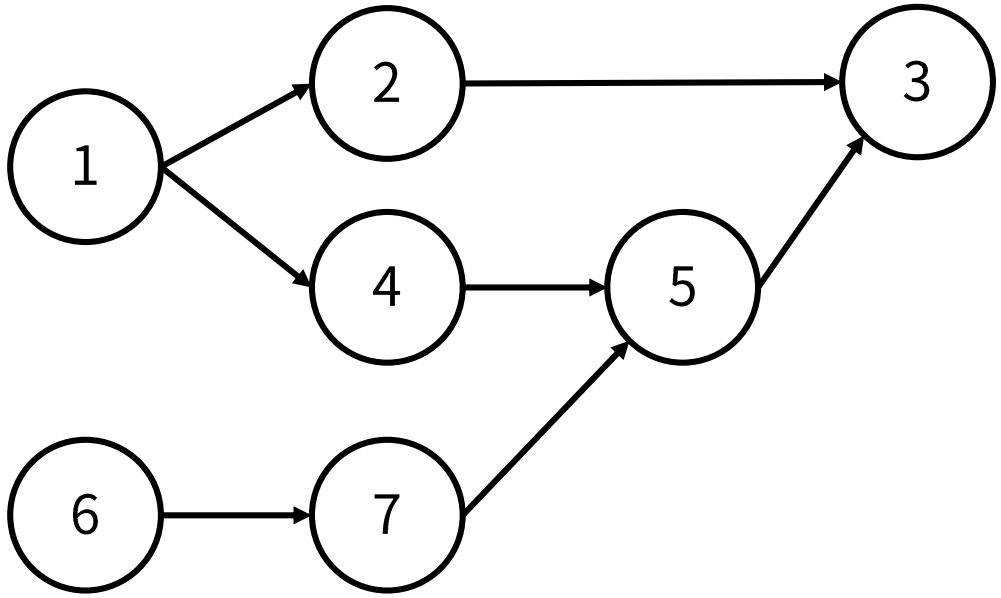
- 정답:

문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.



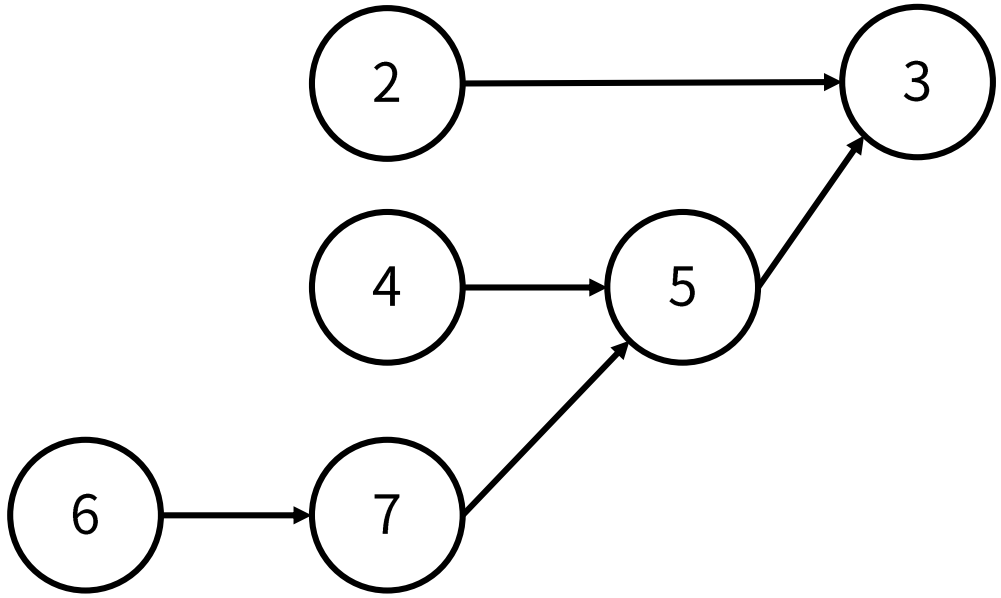
- 정답: ①

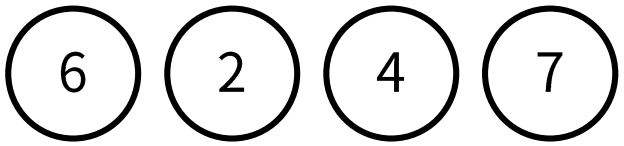
문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.





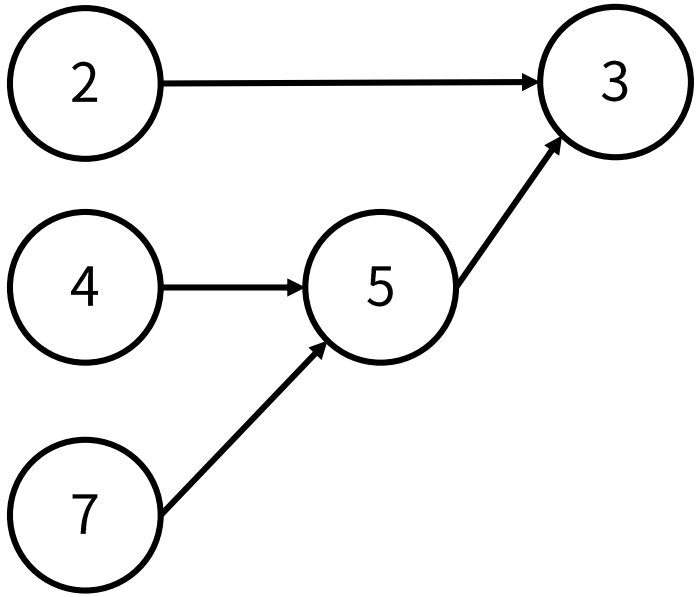
- 정답: ① ⑥

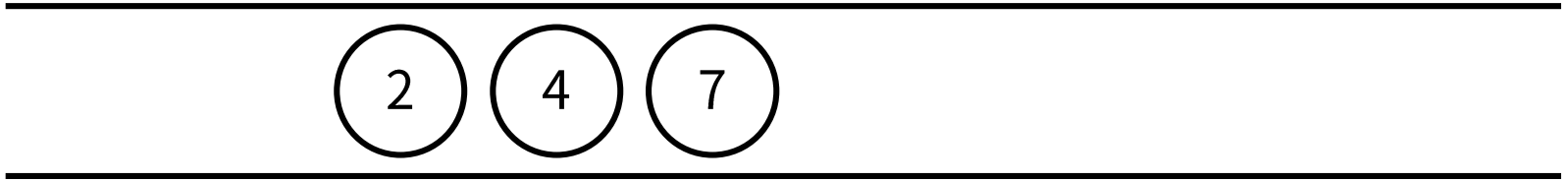
문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.





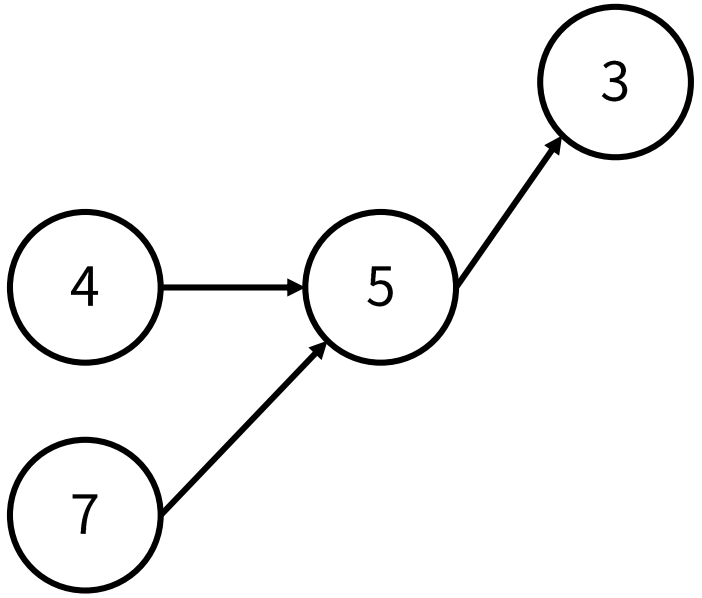
- 정답: ① ⑥ ②

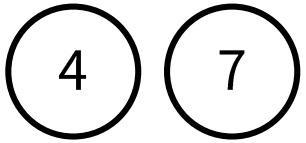
문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.





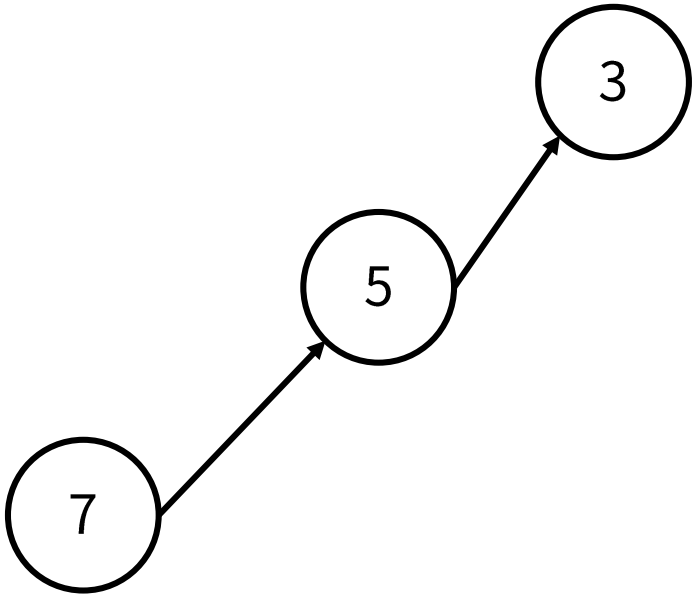
- 정답: ① ⑥ ② ④

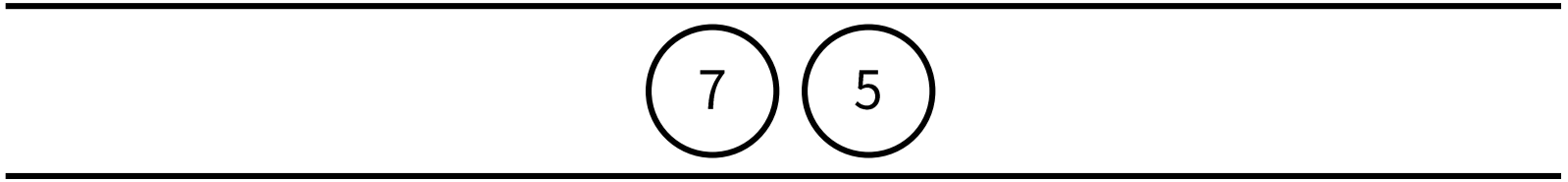
문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.



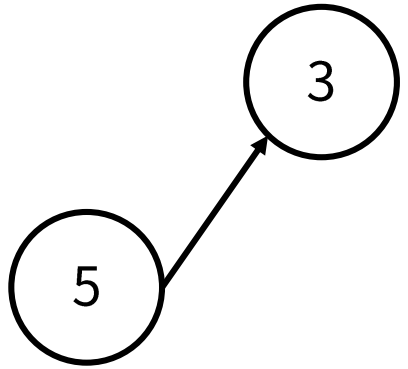


- 정답: ① ⑥ ② ④ ⑦

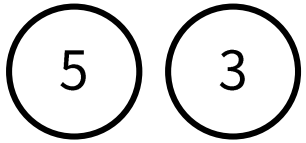
문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘



1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.



- 정답: ① ⑥ ② ④ ⑦ ⑤

문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

3

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.

3

- 정답: ① ⑥ ② ④ ⑦ ⑤ ③

문제집: https://www.acmicpc.net/problem/1766

I 문제 풀이 핵심 아이디어

위상 정렬(Topology Sort) 알고리즘

1) 진입 차수가 0인 정점을 큐에 삽입합니다.
2) 큐에서 원소를 꺼내 해당 원소와 간선을 제거합니다.
3) 제거 이후에 진입 차수가 0이 된 정점을 큐에 삽입합니다.
4) 큐가 빌 때까지 2) - 3) 과정을 반복합니다.

- 정답: ① ⑥ ② ④ ⑦ ⑤ ③

문제집: https://www.acmicpc.net/problem/1766

소스코드

```
import heapq

n, m = map(int, input().split())
array = [[] for i in range(n + 1)]
indegree = [0] * (n + 1)

heap = []
result = []

for _ in range(m):
    x, y = map(int, input().split())
    array[x].append(y)
    indegree[y] += 1

for i in range(1, n + 1):
    if indegree[i] == 0:
        heapq.heappush(heap, i)
```

```
result = []

while heap:
    data = heapq.heappop(heap)
    result.append(data)
    for y in array[data]:
        indegree[y] -= 1
        if indegree[y] == 0:
            heapq.heappush(heap, y)

for i in result:
    print(i, end=' ')
```