Chapter 02.
고급 자료구조

핵심 유형 문제풀이

핵심 유형 문제풀이 | 다양한 문제를 접하며 코딩 테스트에 익숙해지기

강사 나동빈

Chapter 02. 고급 자료구조

핵심 유형 문제풀이

주사위: https://www.acmicpc.net/problem/1233

혼자 힘으로 풀어보기

문제 제목: 주사위

문제 난이도: 하(Easy)

문제 유형: 기초 문법, 완전 탐색, 배열, 사전(dictionary) 자료형

추천 풀이 시간: 30분

주사위: https://www.acmicpc.net/problem/1233

문제 풀이 핵심 아이디어

- 주사위 세 개를 던지는 "모든 경우의 수"를 고려한다.
- 따라서 3중 반복문을 이용하여 주사위 눈금의 합을 계산한다.
- 이때 각 합에 대하여 **"카운트"**하여 가장 많이 발생하는 합을 계산한다.

주사위: https://www.acmicpc.net/problem/1233

소스 코드


```python
import sys
# 빠른 입력 함수 사용
input = sys.stdin.readline

# 세 주사위의 면 개수 각각 입력받기
s1, s2, s3 = map(int, input().split())
counter = dict()

# 3중 반복문을 이용해 모든 경우의 수 계산
for i in range(1, s1 + 1):
    for j in range(1, s2 + 1):
        for k in range(1, s3 + 1):
            summary = i + j + k # 눈금의 합
            # 각 합의 등장 횟수 "카운트"
            if summary not in counter:
                counter[summary] = 1
            else:
                counter[summary] += 1
```

```python
# 가장 많이 등장한 합 찾기
max_count = -1
answer = int(1e9)
for (key, value) in counter.items():
    # "등장 횟수(빈도수)"가 더 많은 경우를 찾은 경우
    if max_count < value:
        max_count = value
        answer = key
    elif max_count == value:
        # 빈도수가 동일한 것이 여러 개라면, 가장 작은 것 출력
        answer = min(answer, key)
print(answer)
```

강의실: https://www.acmicpc.net/problem/1374

혼자 힘으로 풀어보기

문제 제목: 강의실

문제 난이도: 중(Medium)

문제 유형: 자료구조, 그리디, 정렬, 우선순위 큐

추천 풀이 시간: 50분

강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

- 본 문제는 우선순위 큐 자료구조를 이용해 해결할 수 있는 문제입니다.
- 파이썬의 $heapq$ 라이브러리를 이용해 **우선순위 큐**를 활용해 해결합니다.

강의실: https://www.acmicpc.net/problem/1374

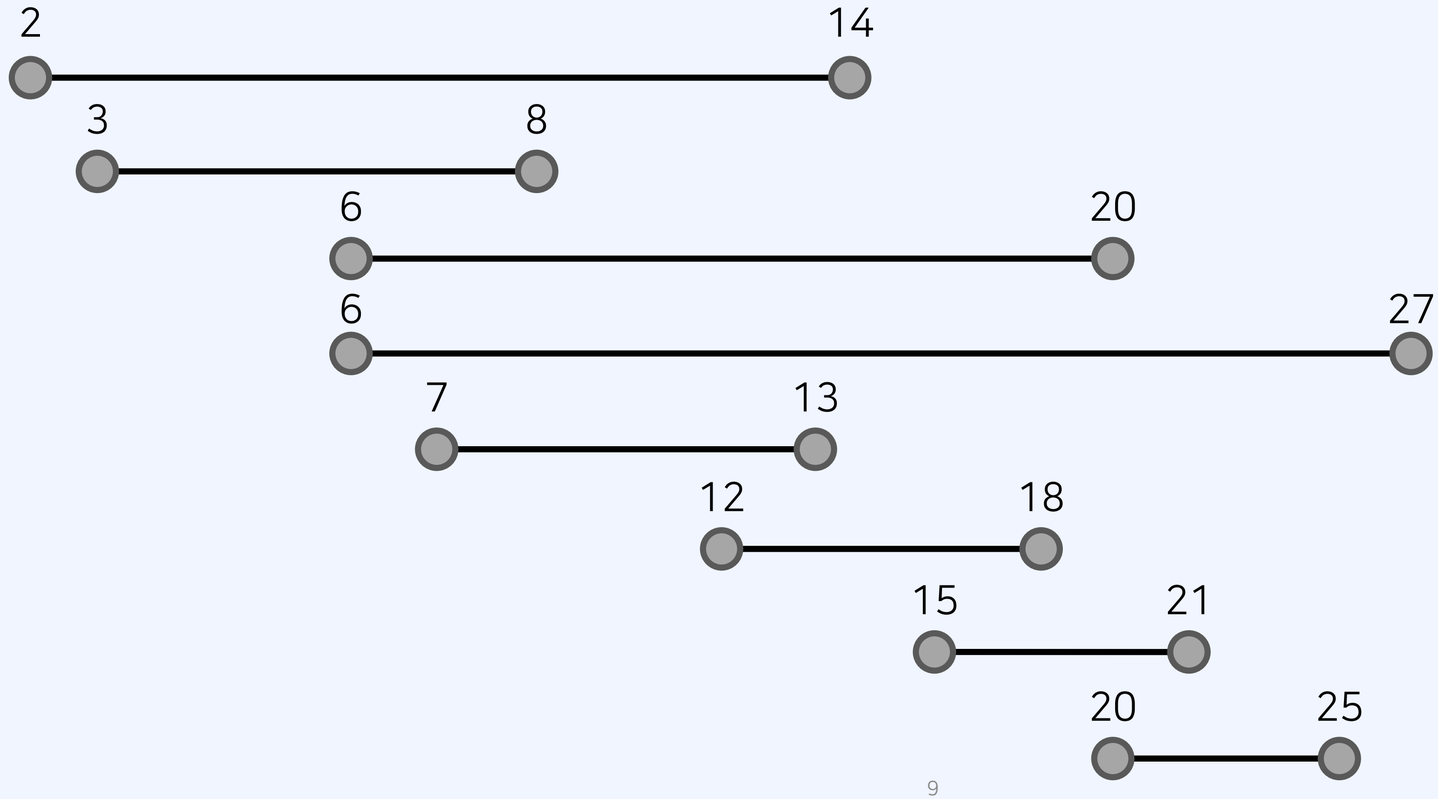
문제 풀이 핵심 아이디어

- 문제의 예시와 같이 8개의 강의가 존재한다고 가정하자.
  - **[강의 목록]**
  - 시작 시간: 15, 종료 시간: 21
  - 시작 시간: 20, 종료 시간: 25
  - 시작 시간: 3, 종료 시간: 8
  - 시작 시간: 2, 종료 시간: 14
  - 시작 시간: 6, 종료 시간: 27
  - 시작 시간: 7, 종료 시간: 13
  - 시작 시간: 12, 종료 시간: 18
  - 시작 시간: 6, 종료 시간: 20

강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

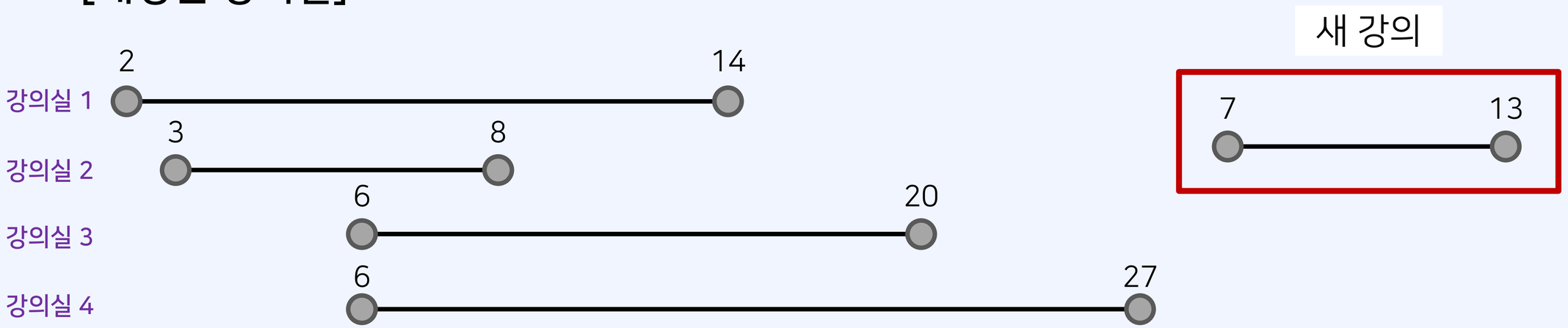
- 가장 먼저 **[강의 목록]**의 각 강의를 “시작 시간”을 기준으로 정렬한다.
- 따라서 강의를 하나씩 확인할 때 **먼저 시작하는 강의부터 확인**할 수 있다.



강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

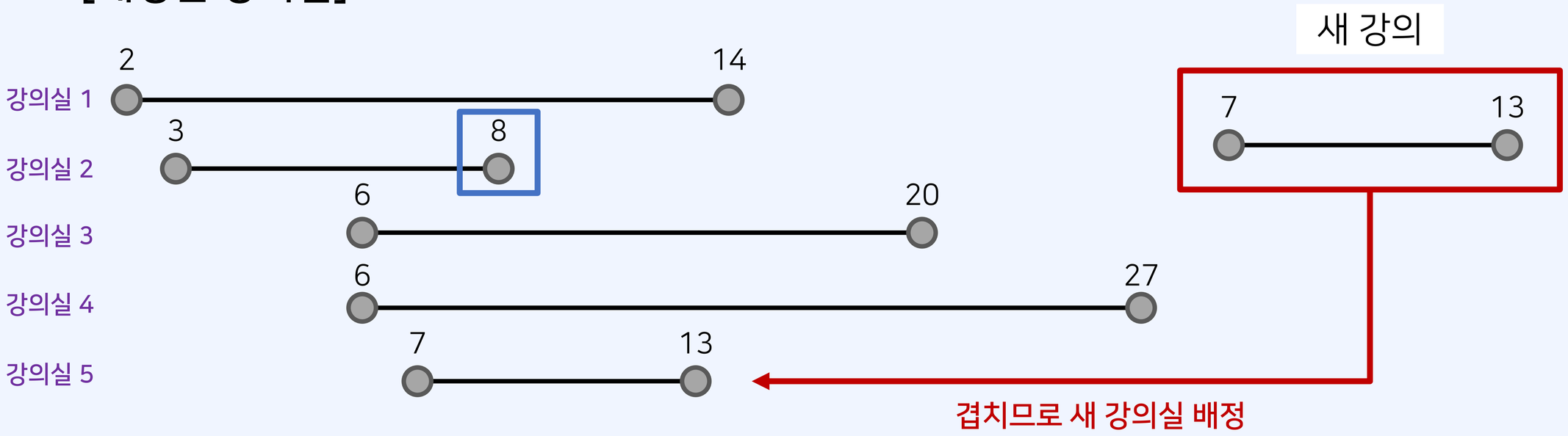
- [강의 목록]에서 새 강의를 하나씩 확인한다.
- 이때, “시작 시간”을 현재 배정된 강의실에서 “가장 빠른 끝나는 시간“과 비교한다.
- 강의 시간이 겹친다면 새 강의실을 배정하고, 겹치지 않는다면 기존 강의실에 넣는다.
- [배정된 강의실]



강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

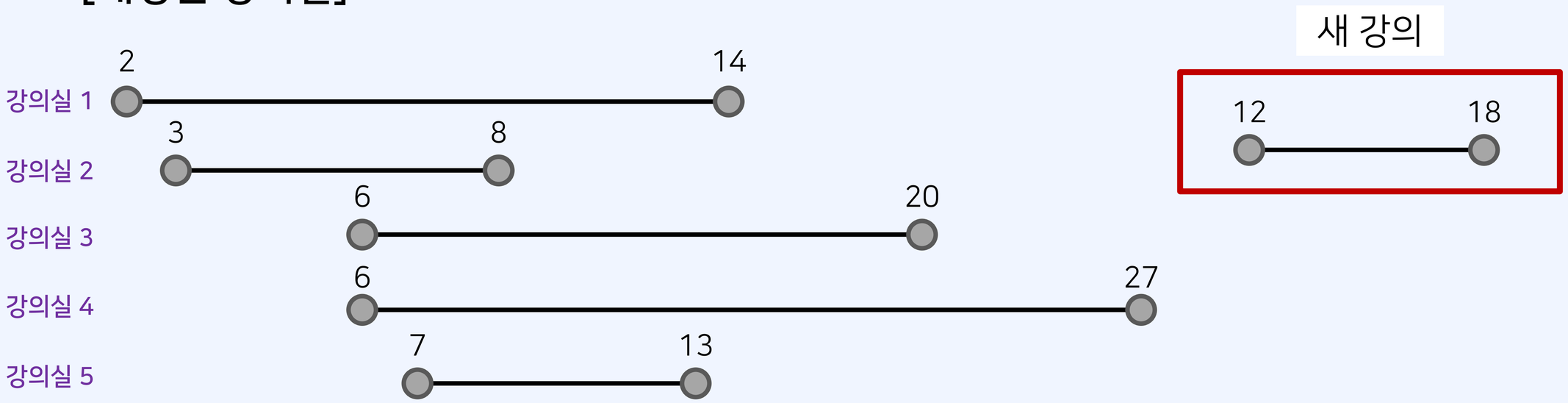
- [강의 목록]에서 새 강의를 하나씩 확인한다.
- 이때, "시작 시간"을 현재 배정된 강의실에서 "가장 빠른 끝나는 시간"과 비교한다.
- 강의 시간이 겹친다면 새 강의실을 배정하고, 겹치지 않는다면 기존 강의실에 넣는다.
- [배정된 강의실]



강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

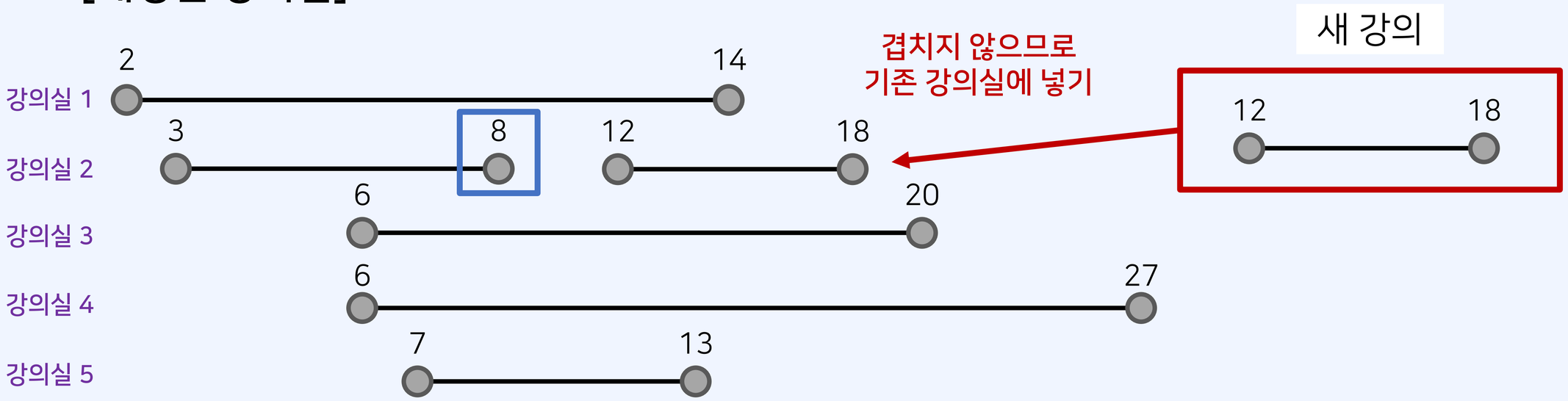
- [강의 목록]에서 새 강의를 하나씩 확인한다.
- 이때, “시작 시간”을 현재 배정된 강의실에서 “가장 빠른 끝나는 시간“과 비교한다.
- 강의 시간이 겹친다면 새 강의실을 배정하고, 겹치지 않는다면 기존 강의실에 넣는다.
- [배정된 강의실]



강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

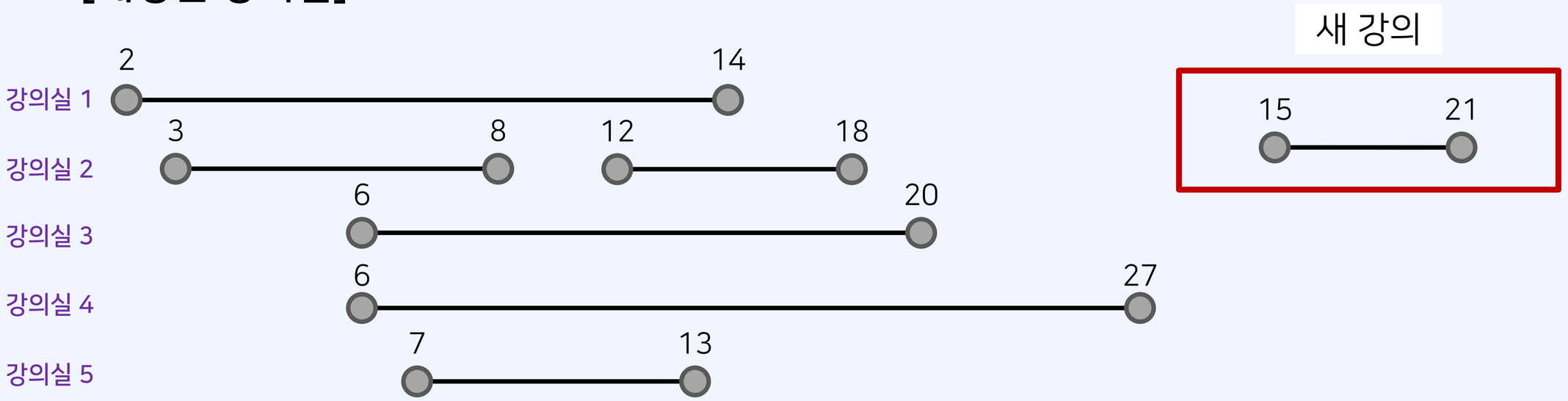
- [강의 목록]에서 새 강의를 하나씩 확인한다.
- 이때, "시작 시간"을 현재 배정된 강의실에서 "가장 빠른 끝나는 시간"과 비교한다.
- 강의 시간이 겹친다면 새 강의실을 배정하고, 겹치지 않는다면 기존 강의실에 넣는다.
- [배정된 강의실]



강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

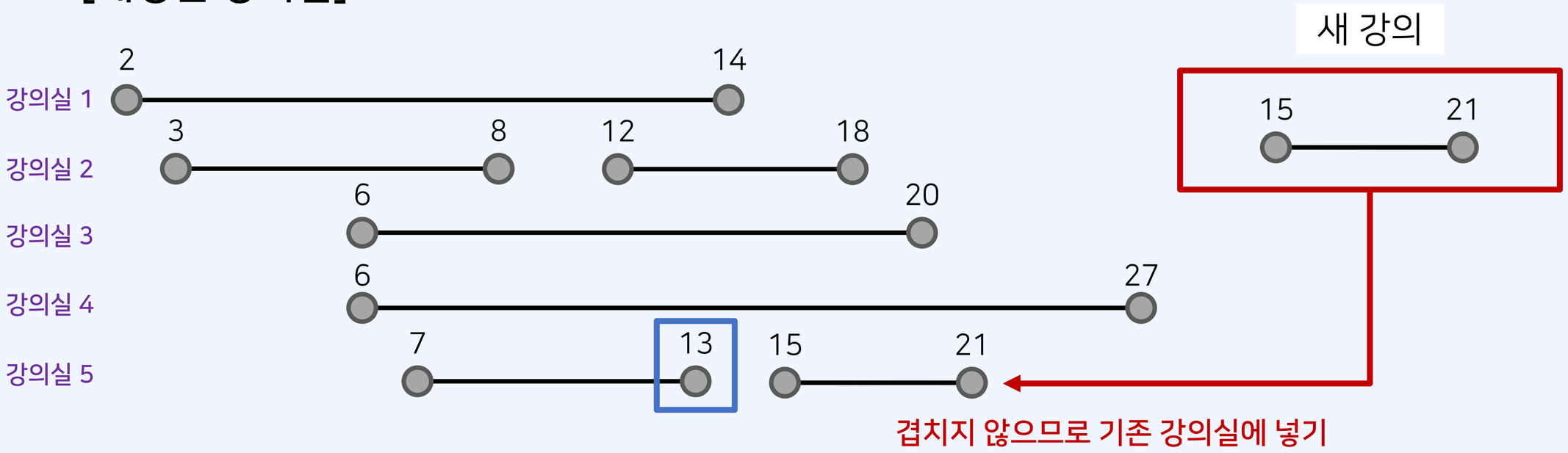
- [강의 목록]에서 새 강의를 하나씩 확인한다.
- 이때, "시작 시간"을 현재 배정된 강의실에서 "가장 빠른 끝나는 시간"과 비교한다.
- 강의 시간이 겹친다면 새 강의실을 배정하고, 겹치지 않는다면 기존 강의실에 넣는다.
- [배정된 강의실]



강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

- [강의 목록]에서 새 강의를 하나씩 확인한다.
- 이때, "시작 시간"을 현재 배정된 강의실에서 "가장 빠른 끝나는 시간"과 비교한다.
- 강의 시간이 겹친다면 새 강의실을 배정하고, 겹치지 않는다면 기존 강의실에 넣는다.
- [배정된 강의실]



강의실: https://www.acmicpc.net/problem/1374

문제 풀이 핵심 아이디어

[알고리즘 요약]

- (시작 시간, 종료 시간) 형태로 **[강의 목록]**에 삽입한 뒤에, **정렬**한다.
  - 강의를 하나씩 꺼내서 확인할 때 시작 시간을 기준으로 확인할 수 있다. (우선순위 큐를 사용)
- 배정된 강의실을 처리하기 위한 우선순위 큐를 하나 더 준비한다. (강의 종료 시간을 삽입)
- 먼저 첫 번째 강의를 꺼내어 우선순위 큐에 삽입하고, 아래를 반복한다.
  - "강의 목록"에서 **[새 강의]**를 꺼내어 확인한다.
  - "배정된 강의실(우선순위 큐)"에서 하나의 **[기존 강의]**를 꺼내어 확인한다.

1. 강의 시간이 겹치면 새로운 강의실이 필요하다. (새 강의와 기존 강의 모두 넣기)
2. 강의 시간이 겹치지 않으면 기존 강의실에 새 강의를 배정한다. (새 강의 넣기)

강의실: https://www.acmicpc.net/problem/1374

Ch2. 고급 자료구조
핵심 유형 문제풀이

소스 코드

```python
import sys
# 빠른 입력 함수 사용
input = sys.stdin.readline
import heapq

n = int(input()) # 강의 개수 N 입력받기
lectures = []
for i in range(n):
    # 강의 번호, 시작 시간, 종료 시간 입력받기
    id, start, end = map(int, input().split())
    # (시작 시간, 종료 시간)을 기준으로 힙에 삽입
    heapq.heappush(lectures, (start, end))
    # 우선순위 큐를 이용하므로, 자동적으로 정렬

heap = [] # 배정된 강의실
# 첫 번째 강의가 끝나는 시간을 우선순위 큐에 삽입
end = heapq.heappop(lectures)[1]
heapq.heappush(heap, end)
answer = 1 # 필요한 강의실 개수(최종 정답)
for i in range(n - 1):
    # 강의 목록에서 [새 강의] 꺼내기
    new_start, new_end = heapq.heappop(lectures)
    # [기존 강의] 중에서 가장 일찍 끝나는 강의 꺼내기
    end = heapq.heappop(heap)

    if new_start < end: # 강의 시간이 겹치면
        heapq.heappush(heap, end) # 기존 강의실 유지
        heapq.heappush(heap, new_end) # 새로운 강의실 추가
        answer += 1
    else:
        heapq.heappush(heap, new_end) # 기존 강의실에 배정

print(answer)
```

중앙값 구하기: https://www.acmicpc.net/problem/2696

혼자 힘으로 풀어보기

문제 제목: 중앙값 구하기

문제 난이도: 중상

문제 유형: 자료구조, 우선순위 큐

추천 풀이 시간: 60분

중앙값 구하기: https://www.acmicpc.net/problem/2696

문제 풀이 핵심 아이디어

- 본 문제는 우선순위 큐 자료구조를 이용해 해결할 수 있는 문제입니다.
- 파이썬의 $heapq$ 라이브러리를 이용해 **우선순위 큐**를 활용해 해결합니다.

중앙값 구하기: https://www.acmicpc.net/problem/2696

문제 풀이 핵심 아이디어

- 가장 쉽게 떠올릴 수 있는 해결 방법은 다음과 같다.
  1. 배열에 원소를 차례대로 넣는다.
  2. 홀수 번째가 입력될 때마다 정렬을 수행한다.
- **한계점**: 단, 정렬은 $O(NlogN)$의 수행 시간을 요구하므로, 매번 정렬을 이용하여 중앙값을 계산하는 것은 비효율적이다.

중앙값 구하기: https://www.acmicpc.net/problem/2696

문제 풀이 핵심 아이디어

[알고리즘 요약]

- **[최대 힙] - [중앙값] - [최소 힙]** 형태의 구조를 갖춘다.
- 새로운 원소가 들어오면 다음과 같이 한다.
  - 중앙값보다 작거나 같은 원소는 **왼쪽의 최대 힙**에 넣는다.
  - 중앙값보다 큰 원소는 **오른쪽의 최소 힙**에 넣는다.
- 홀수 번째 원소가 입력될 때마다 최대 힙과 최소 힙의 크기를 동일하게 맞춘다.
  - 한쪽이 더 크다면, 반대편으로 이동시킨다.

중앙값 구하기: https://www.acmicpc.net/problem/2696

문제 풀이 핵심 아이디어

- 새 원소가 중앙값보다 작거나 같으면 **최대 힙**에, 크다면 **최소 힙**에 넣는다.

① 간단히 1, 5, 4, 3, 2가 있다고 가정하자.

입력: [1, 5, 4, 3, 2]

최대 힙: [2, 1] / **중앙값**: 3 / **최소 힙**: [4, 5]

② 추가적으로 6, 7이 들어왔다면, 다음과 같이 처리된다.

입력: [1, 5, 4, 3, 2, 6, 7]

최대 힙: [2, 1] / **중앙값**: 3 / **최소 힙**: [4, 5, 6, 7]

두 힙의 크기가 다르므로, 최소 힙에서
최대 힙 방향으로 1개씩 이동한다.

최대 힙: [3, 2, 1] / **중앙값**: 4 / **최소 힙**: [5, 6, 7]

중앙값 구하기: https://www.acmicpc.net/problem/2696

소스 코드


```python
import sys
# 빠른 입력 함수 사용
input = sys.stdin.readline
import heapq

def show_result():
    # 출력할 총 원소의 개수
    print(len(result))
    for i in range(len(result)):
        print(result[i], end=' ')
        # 10개 단위로 줄바꿈
        if (i + 1) % 10 == 0:
            print()
    print()
```

```python
for _ in range(int(input())): # 테스트 케이스만큼 반복
    m = int(input()) # 수열의 크기 M(M은 항상 홀수)
    data = []
    for i in range(m // 10 + 1):
        data.extend(list(map(int, input().split()))) # 수열 입력
    left = [] # 왼쪽 최대 힙 (원소를 삽입/삭제할 때 음수 부호)
    right = [] # 오른쪽 최소 힙
    median = data[0] # 중앙값
    result = [median] # 결과 배열
    for i in range(1, m): # 수열의 원소를 하나씩 확인하며
        if data[i] <= median: heapq.heappush(left, -data[i])
        else: heapq.heappush(right, data[i])
        if i % 2 == 0: # 2개 확인할 때마다
            if len(left) > len(right): # 왼쪽에서 오른쪽
                heapq.heappush(right, median)
                median = -heapq.heappop(left)
            elif len(left) < len(right): # 오른쪽에서 왼쪽
                heapq.heappush(left, -median)
                median = heapq.heappop(right)
            result.append(median)
    show_result() # 정답 출력
```