Chapter. 11

탐욕 알고리즘

핵심 유형 문제풀이

FAST CAMPUS
ONLINE
유형별 문제풀이

강사. 나동빈

Chapter. 11

탐욕 알고리즘(핵심 유형 문제풀이)

FAST CAMPUS
ONLINE
나동빈 강사.

혼자 힘으로 풀어 보기

센서: https://www.acmicpc.net/problem/2212

문제 제목: 센서

문제 난이도: 하(Easy)

문제 유형: 그리디

추천 풀이 시간: 30분

센서: https://www.acmicpc.net/problem/2212

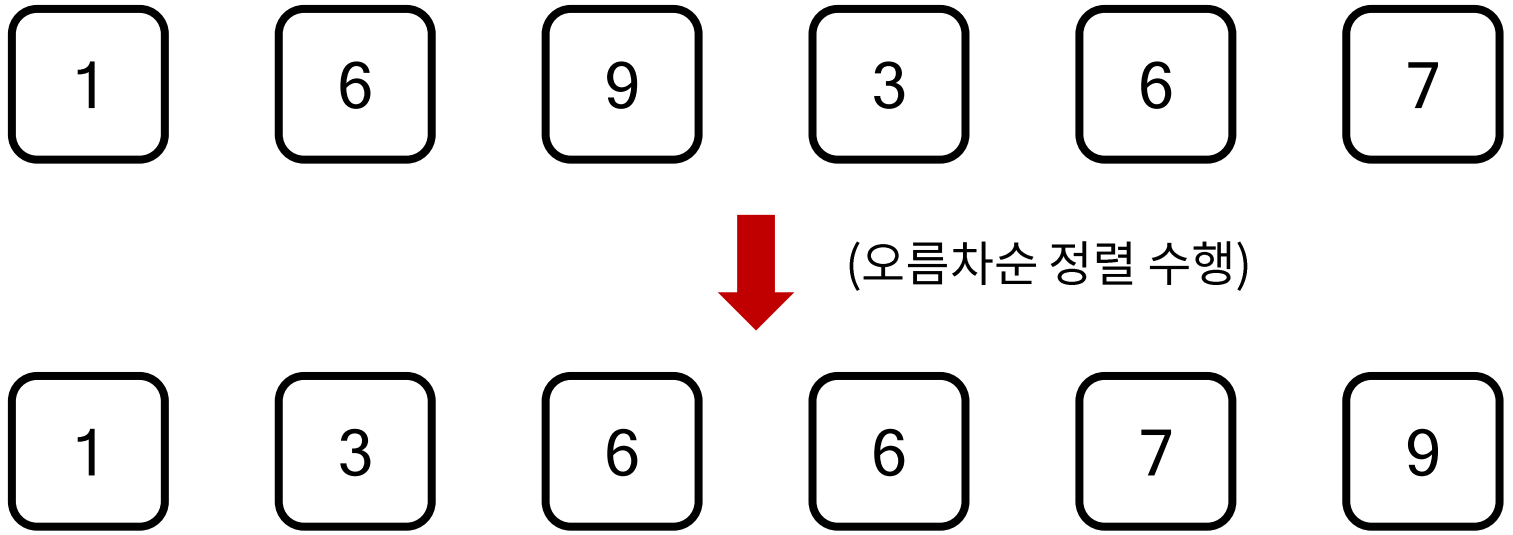
I 문제 풀이 핵심 아이디어

- 최대 K개의 집중국을 설치해야 합니다.
- 집중국들의 수신 가능 영역의 길이의 합을 최소화하는 것이 목표입니다.
- 사실상 정렬만 수행하면 되므로 $\boldsymbol{O(NlogN)}$으로 문제를 해결할 수 있습니다.

센서: https://www.acmicpc.net/problem/2212

문제 풀이 핵심 아이디어

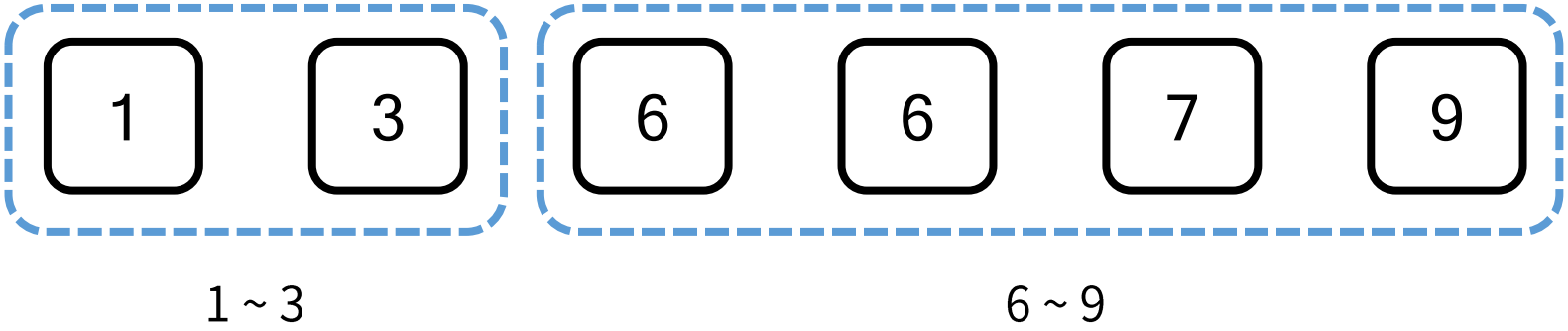
- 각 센서들을 위치를 기준으로 오름차순 정렬을 수행합니다.



센서: https://www.acmicpc.net/problem/2212

I 문제 풀이 핵심 아이디어

- 문제의 요구사항은, 정렬된 센서들을 최대 K개의 영역으로 나누는 것과 동일합니다.
- K = 2일 때, 각 집중국의 수신 가능 영역은 다음과 같습니다.

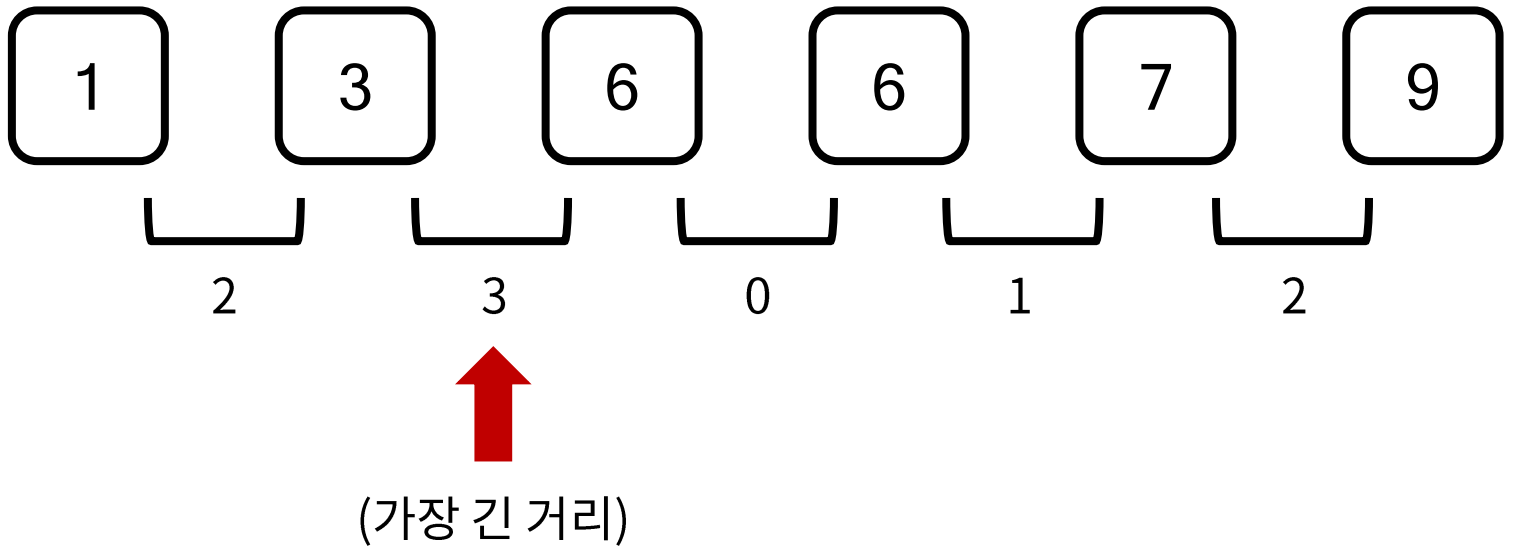


수신 가능 영역의 길이의 합의 최솟값: **5**

센서: https://www.acmicpc.net/problem/2212

I 문제 풀이 핵심 아이디어

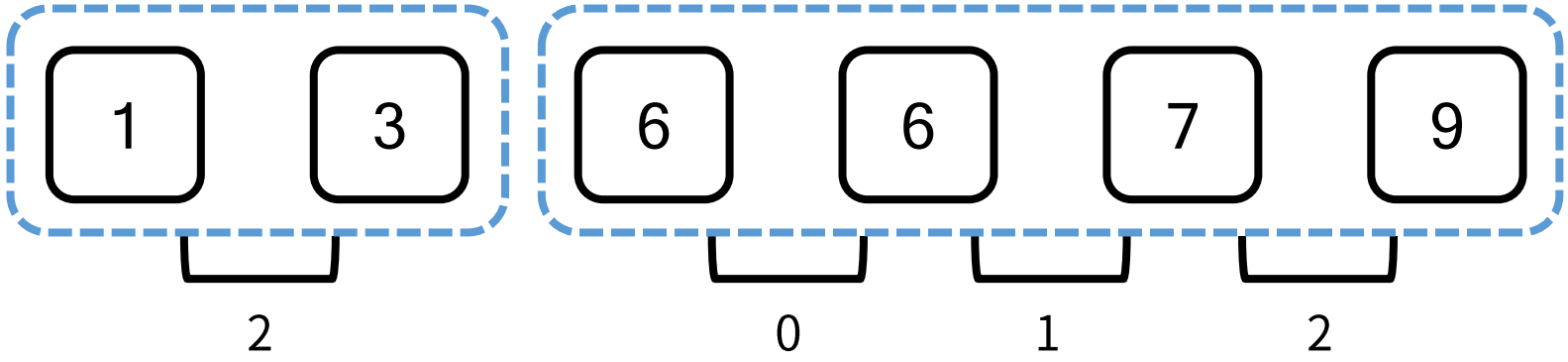
- 따라서, 문제의 알고리즘은 다음과 같습니다.
  - 각 센서를 오름차순 정렬합니다.
  - 각 센서 사이의 거리를 계산합니다.
  - 가장 거리가 먼 순서대로 K - 1개의 연결 고리를 제거합니다.



센서: https://www.acmicpc.net/problem/2212

I 문제 풀이 핵심 아이디어

- 따라서, 문제의 알고리즘은 다음과 같습니다.
  - 각 센서를 오름차순 정렬합니다.
  - 각 센서 사이의 거리를 계산합니다.
  - 가장 거리가 먼 순서대로 K - 1개의 연결 고리를 제거합니다.



➡ 남아있는 모든 거리들의 합 (정답): **5**

센서: https://www.acmicpc.net/problem/2212

소스코드

```python
import sys

n = int(input())
k = int(input())

# 집중국의 개수가 n 이상일 때
if k >= n:
    print(0) # 각 센서의 위치에 설치하면 되므로 정답은 0
    sys.exit()

# 모든 센서의 위치를 입력 받아 오름차순 정렬
array = list(map(int, input().split(' ')))
array.sort()

# 각 센서 간의 거리를 계산하여 내림차순 정렬
distances = []
for i in range(1, n):
    distances.append(array[i] - array[i - 1])
distances.sort(reverse=True)

# 가장 긴 거리부터 하나씩 제거
for i in range(k - 1):
    distances[i] = 0
print(sum(distances))
```

혼자 힘으로 풀어 보기

도서관: https://www.acmicpc.net/problem/1461

문제 제목: 도서관

문제 난이도: 중(Medium)

문제 유형: 그리디

추천 풀이 시간: 40분

도서관: https://www.acmicpc.net/problem/1461

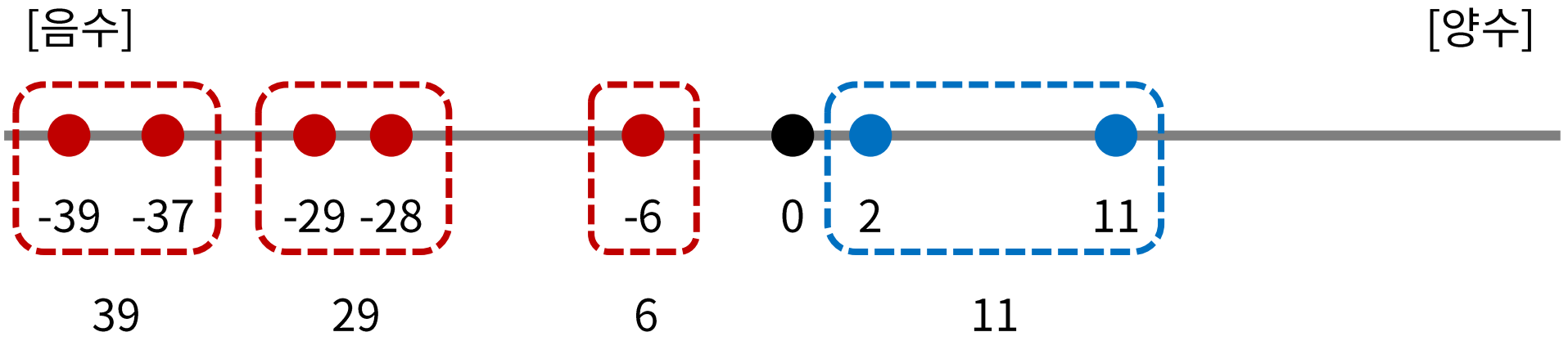
문제 풀이 핵심 아이디어

- 일직선상의 각 책들을 원래의 위치에 놓아야 합니다.
- 0보다 큰 책들과 0보다 작은 책들을 나누어서 처리합니다.
- 두 개의 우선순위 큐를 이용하여 문제를 효과적으로 해결할 수 있습니다.
- 마지막 책을 놓을 때는 다시 0으로 돌아올 필요가 없으므로, 가장 먼 책을 마지막으로 놓습니다.

도서관: https://www.acmicpc.net/problem/1461

I 문제 풀이 핵심 아이디어

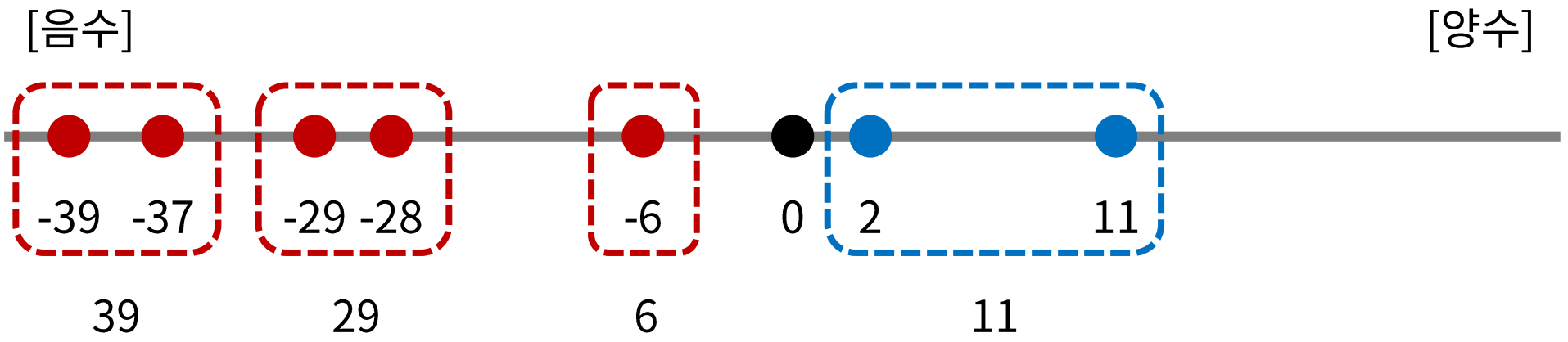
- 책의 개수(N) = 7, 한 번에 들 수 있는 책의 개수(M) = 2
- 음수와 양수에 대하여 개별적으로, M개씩 묶어서 처리합니다.
- M개씩의 묶음 중에서 가장 거리가 먼 책만큼 이동해야 합니다.



도서관: https://www.acmicpc.net/problem/1461

I 문제 풀이 핵심 아이디어

- 책의 개수(N) = 7, 한 번에 들 수 있는 책의 개수(M) = 2
- 음수와 양수에 대하여 개별적으로, M개씩 묶어서 처리합니다.
- M개씩의 묶음 중에서 가장 거리가 먼 책만큼 이동해야 합니다.



왕복 거리: $(39 + 29 + 6 + 11) * 2 = 170$

가장 먼 책의 편도 거리 제외: $170 - 39 = \mathbf{131}$ (정답)

도서관: https://www.acmicpc.net/problem/1461

소스코드

```python
import heapq

n, m = map(int, input().split(' '))
array = list(map(int, input().split(' ')))
positive = []
negative = []

# 가장 거리가 먼 책까지의 거리
largest = max(max(array), - min(array))

# 최대 힙(Max Heap)을 위해 원소를 음수로 구성
for i in array:
    # 책의 위치가 양수인 경우
    if i > 0:
        heapq.heappush(positive, -i)
    # 책의 위치가 음수인 경우
    else:
        heapq.heappush(negative, i)

result = 0

while positive:
    # 한 번에 m개씩 옮길 수 있으므로 m개씩 빼내기
    result += heapq.heappop(positive)
    for _ in range(m - 1):
        if positive:
            heapq.heappop(positive)

while negative:
    # 한 번에 m개씩 옮길 수 있으므로 m개씩 빼내기
    result += heapq.heappop(negative)
    for _ in range(m - 1):
        if negative:
            heapq.heappop(negative)

# 일반적으로 왕복 거리를 계산하지만, 가장 먼 곳은 편도 거리 계산
print(-result * 2 - largest)
```

컵라면: https://www.acmicpc.net/problem/1781

I 혼자 힘으로 풀어 보기

문제 제목: 컵라면

문제 난이도: 중(Medium)

문제 유형: 그리디

추천 풀이 시간: 30분

컴라면: https://www.acmicpc.net/problem/1781

I 문제 풀이 핵심 아이디어

- 데드라인을 초과하는 문제는 풀 수 없다는 점을 기억해야 합니다.
- 데이터의 개수(N)는 최대 200,000입니다.
- 정렬 및 우선순위 큐를 이용하여 $\boldsymbol{O(NlogN)}$의 시간에 해결할 수 있습니다.

컴라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 가장 먼저, 문제 데이터 중에서 데드라인을 기준으로 오름차순 정렬을 수행합니다.

데드라인	1	1	3	3	2	2	6
컴라면 수	6	7	2	1	4	5	1

(오름차순 정렬 수행)

데드라인	1	1	2	2	3	3	6
컴라면 수	6	7	4	5	1	2	1

컵라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 각 문제의 '컵라면 수'를 우선순위 큐에 넣으면서, 데드라인을 초과하는 경우에는 최소 원소를 제거합니다.

데드라인	1	1	2	2	3	3	6
컵라면 수	6	7	4	5	1	2	1

- 우선순위 큐(Min Heap)의 크기: **0**

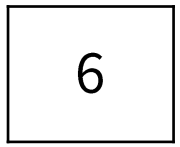
컵라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 각 문제의 '컵라면 수'를 우선순위 큐에 넣으면서, 데드라인을 초과하는 경우에는 최소 원소를 제거합니다.

데드라인	1	1	2	2	3	3	6
컵라면 수	6	7	4	5	1	2	1

- 우선순위 큐(Min Heap)의 크기: **1**



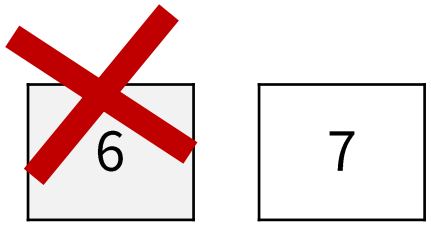
컴라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 각 문제의 '컴라면 수'를 우선순위 큐에 넣으면서, 데드라인을 초과하는 경우에는 최소 원소를 제거합니다.

데드라인	1	1	2	2	3	3	6
컴라면 수	6	7	4	5	1	2	1

- 우선순위 큐(Min Heap)의 크기: **1**



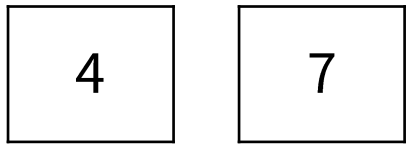
컴라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 각 문제의 '컵라면 수'를 우선순위 큐에 넣으면서, 데드라인을 초과하는 경우에는 최소 원소를 제거합니다.

데드라인	1	1	2	2	3	3	6
컵라면 수	6	7	4	5	1	2	1

- 우선순위 큐(Min Heap)의 크기: **2**



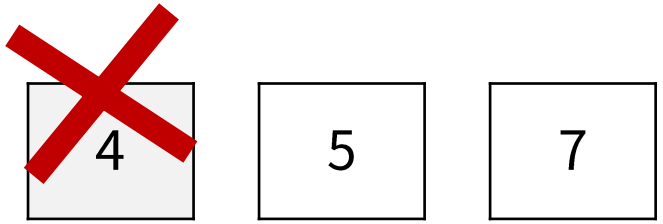
컵라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 각 문제의 '컵라면 수'를 우선순위 큐에 넣으면서, 데드라인을 초과하는 경우에는 최소 원소를 제거합니다.

데드라인	1	1	2	2	3	3	6
컵라면 수	6	7	4	5	1	2	1

- 우선순위 큐(Min Heap)의 크기: **2**



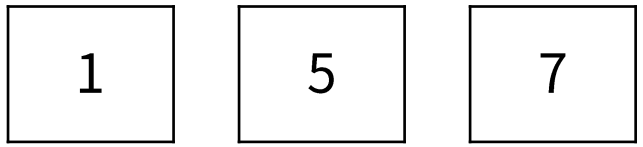
컴라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 각 문제의 '컴라면 수'를 우선순위 큐에 넣으면서, 데드라인을 초과하는 경우에는 최소 원소를 제거합니다.

데드라인	1	1	2	2	3	3	6
컴라면 수	6	7	4	5	1	2	1

- 우선순위 큐(Min Heap)의 크기: **3**



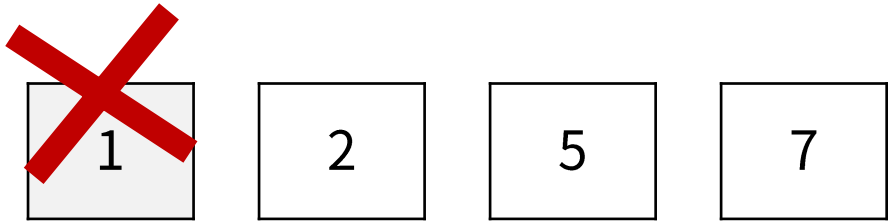
컴라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 각 문제의 '컴라면 수'를 우선순위 큐에 넣으면서, 데드라인을 초과하는 경우에는 최소 원소를 제거합니다.

데드라인	1	1	2	2	3	3	6
컴라면 수	6	7	4	5	1	2	1

- 우선순위 큐(Min Heap)의 크기: **3**



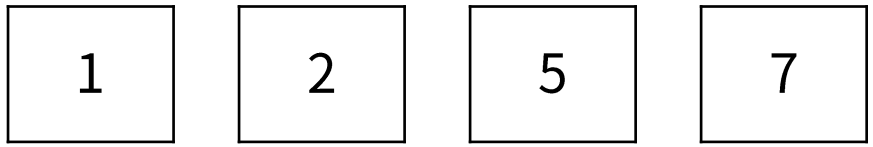
컴라면: https://www.acmicpc.net/problem/1781

문제 풀이 핵심 아이디어

- 각 문제의 '컴라면 수'를 우선순위 큐에 넣으면서, 데드라인을 초과하는 경우에는 최소 원소를 제거합니다.

데드라인	1	1	2	2	3	3	6
컴라면 수	6	7	4	5	1	2	1

- 우선순위 큐(Min Heap)의 크기: **4**



정답: 15

컴라면: https://www.acmicpc.net/problem/1781

소스코드

```python
import heapq

n = int(input())
array = []
q = []

# 문제 정보를 입력 받은 이후에, 데드라인을 기준으로 정렬
for i in range(n):
    a, b = map(int, input().split(' '))
    array.append((a, b))
array.sort()

for i in array:
    a = i[0]
    heapq.heappush(q, i[1])
    # 데드라인을 초과하는 경우에는 최소 원소를 제거
    if a < len(q):
        heapq.heappop(q)

print(sum(q))
```