

Chapter. 11  
탐욕 알고리즘

# | 기초 문제풀이

FAST CAMPUS  
ONLINE  
유형별 문제풀이

강사. 나동빈

Chapter. 11

# 탐욕 알고리즘(기초 문제풀이)

# I 혼자 힘으로 풀어 보기

문제 제목: 거스름돈

문제 난이도: 하(Easy)

문제 유형: 그리디

추천 풀이 시간: 10분

# I 문제 풀이 핵심 아이디어

- 거스름돈의 최소 개수를 계산해야 합니다.
- 가장 기초적인 탐욕 알고리즘 문제 유형입니다.
- 단순히 ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 주면 최적의 해를 얻을 수 있습니다.

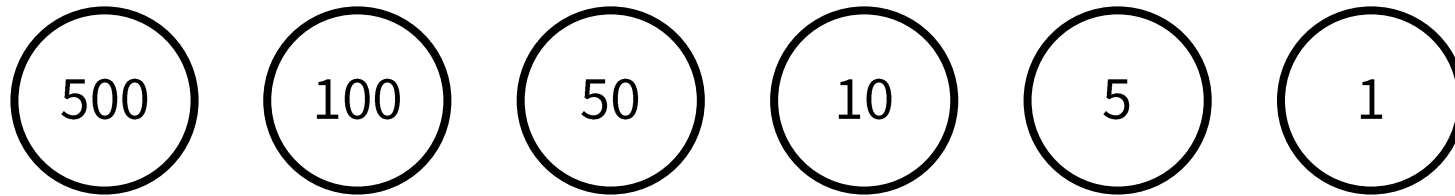
# I 문제 풀이 핵심 아이디어

- ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 줍니다.
- 거슬러 줄 돈: 620

# I 문제 풀이 핵심 아이디어

- ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 줍니다.
- 거슬러 줄 돈: 620
- 거스름돈의 개수: 0

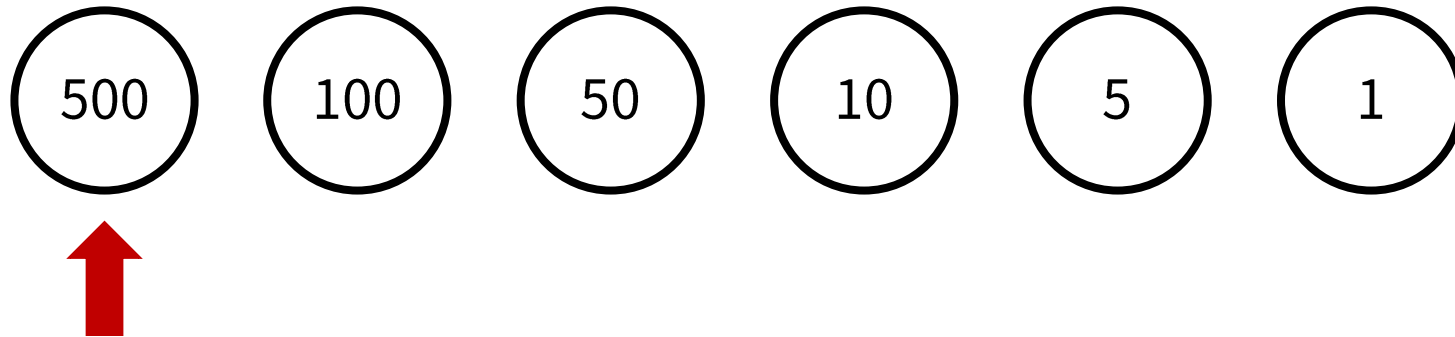
[ 화폐 단위 ]



# I 문제 풀이 핵심 아이디어

- ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 줍니다.
- 거슬러 줄 돈: 120
- 거스름돈의 개수: **1**

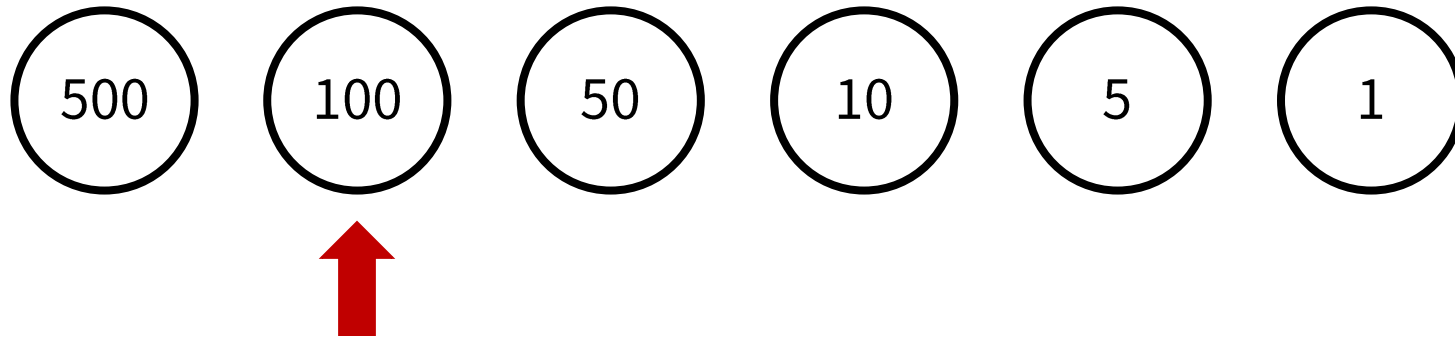
[ 화폐 단위 ]



# I 문제 풀이 핵심 아이디어

- ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 줍니다.
- 거슬러 줄 돈: 20
- 거스름돈의 개수: 2

[ 화폐 단위 ]

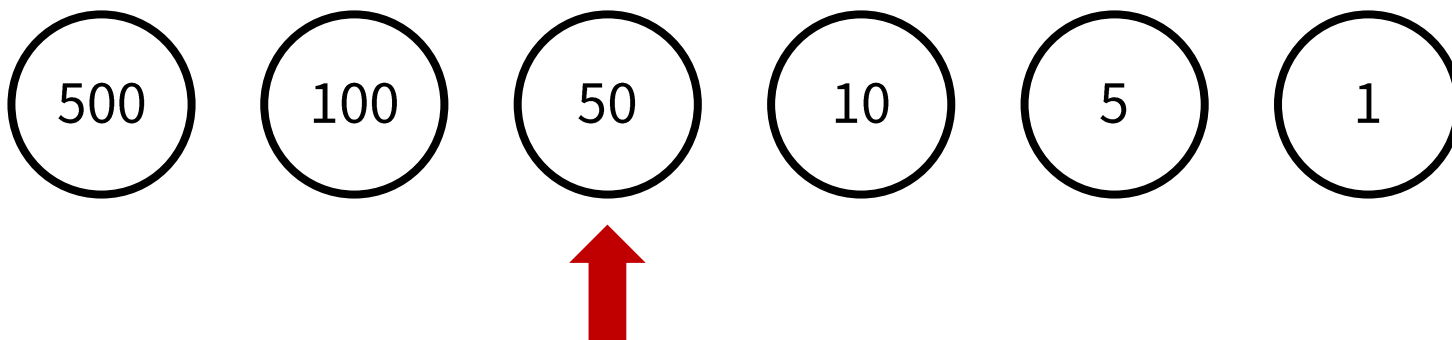




# I 문제 풀이 핵심 아이디어

- ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 줍니다.
- 거슬러 줄 돈: 20
- 거스름돈의 개수: 2

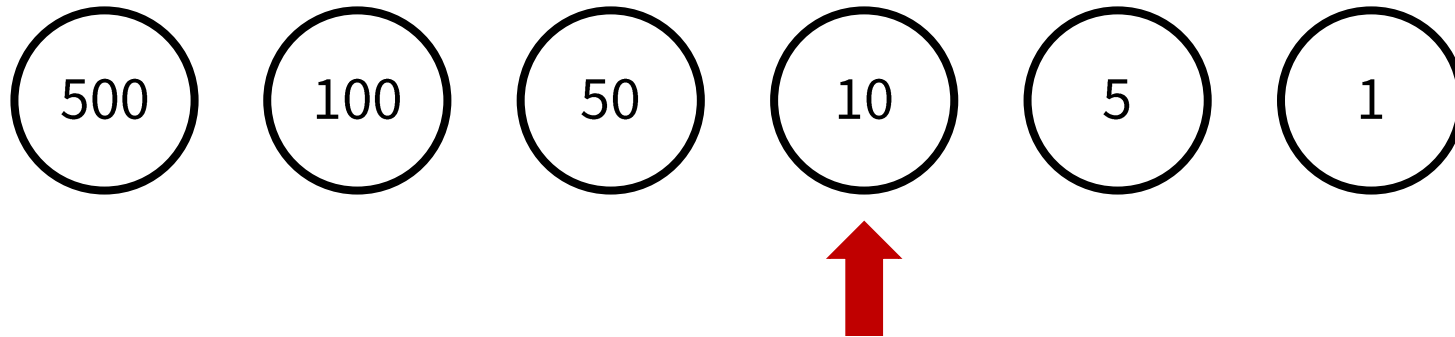
[ 화폐 단위 ]



# I 문제 풀이 핵심 아이디어

- ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 줍니다.
- 거슬러 줄 돈: 0
- 거스름돈의 개수: 4

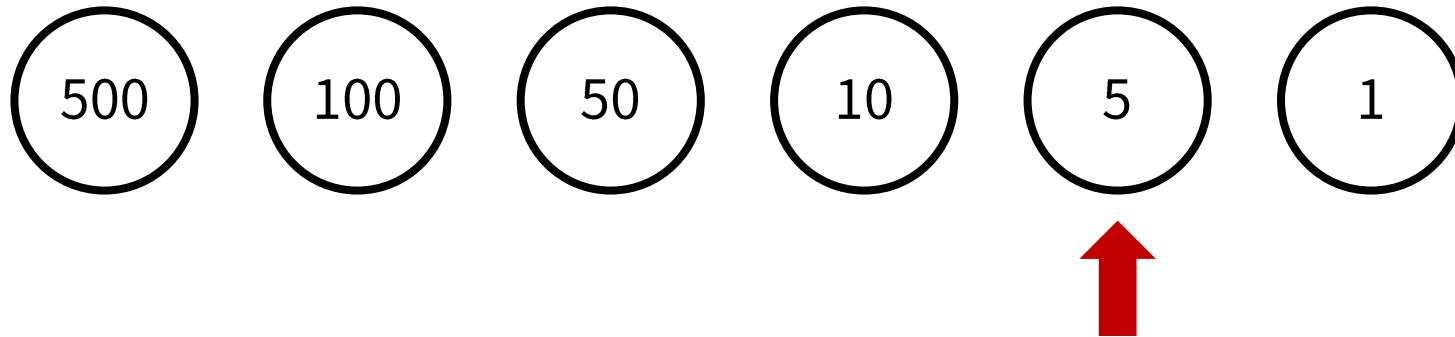
[ 화폐 단위 ]



# I 문제 풀이 핵심 아이디어

- ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 줍니다.
- 거슬러 줄 돈: 0
- 거스름돈의 개수: 4

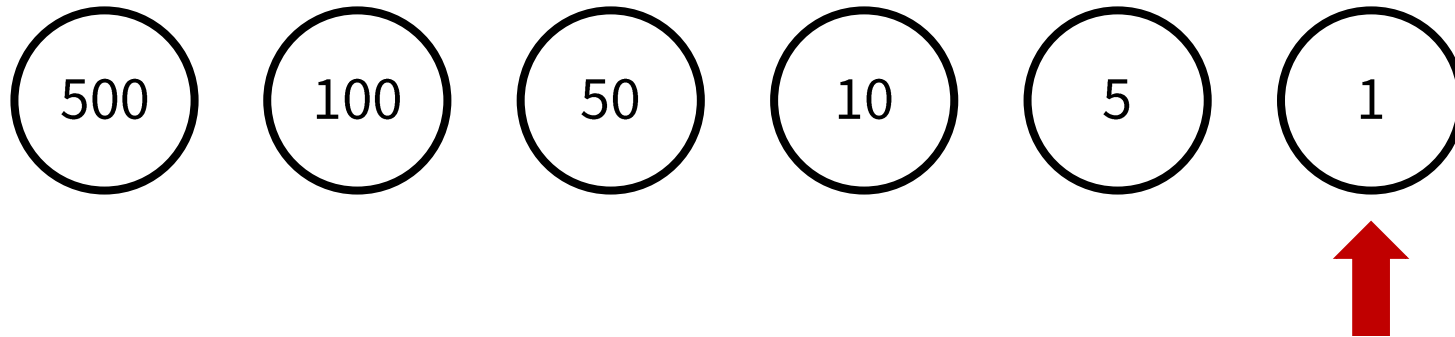
[ 화폐 단위 ]



# I 문제 풀이 핵심 아이디어

- ‘큰 화폐 단위’ 순서대로 잔돈을 거슬러 줍니다.
- 거슬러 줄 돈: 0
- 거스름돈의 개수: 4

[ 화폐 단위 ]



# | 소스코드

```
changes = 1000 - int(input())
count = 0

for i in [500, 100, 50, 10, 5, 1]:
    count += changes // i
    changes %= i

print(count)
```

# I 혼자 힘으로 풀어 보기

문제 제목: 뒤집기

문제 난이도: 하(Easy)

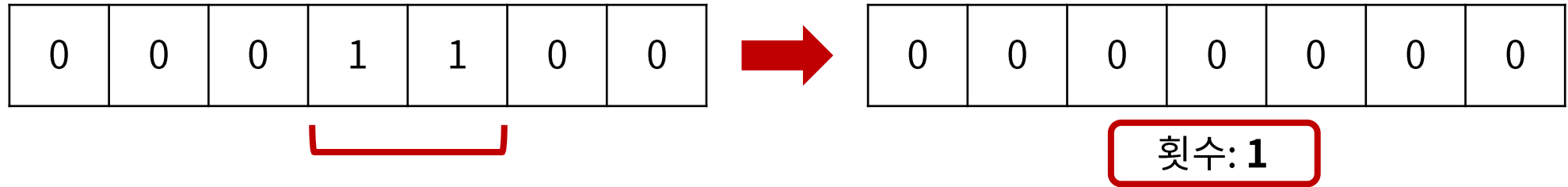
문제 유형: 그리디

추천 풀이 시간: 20분

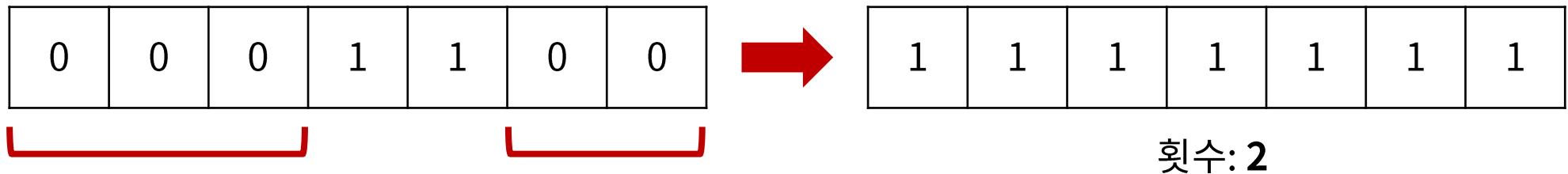
# I 문제 풀이 핵심 아이디어

- 문자열에 있는 숫자를 모두 0 혹은 모두 1로 만들어야 합니다.
- 다음의 두 가지 경우를 모두 계산하면 됩니다.
- 문자열 S의 길이는 100만 이하이므로, 시간 복잡도는  $O(N)$ 에 해결해야 합니다.

## ① 모두 0으로 만드는 경우



## ② 모두 1로 만드는 경우



# I 문제 풀이 핵심 아이디어

- 구체적인 알고리즘 예시) 모두 1로 만드는 경우
  - 첫 번째 원소가 0인 경우
  - 2개씩 원소를 비교할 때, 1에서 0으로 바뀌는 경우

0	0	0	1	1	0	0
---	---	---	---	---	---	---

 횟수: 0



# I 문제 풀이 핵심 아이디어

- 구체적인 알고리즘 예시) 모두 1로 만드는 경우
  - 첫 번째 원소가 0인 경우
  - 2개씩 원소를 비교할 때, 1에서 0으로 바뀌는 경우

0	0	0	1	1	0	0
---	---	---	---	---	---	---




 횟수: 1

# I 문제 풀이 핵심 아이디어

- 구체적인 알고리즘 예시) 모두 1로 만드는 경우
  - 첫 번째 원소가 0인 경우
  - 2개씩 원소를 비교할 때, 1에서 0으로 바뀌는 경우

0	0	0	1	1	0	0
---	---	---	---	---	---	---



 횟수: 1

# I 문제 풀이 핵심 아이디어

- 구체적인 알고리즘 예시) 모두 1로 만드는 경우
  - 첫 번째 원소가 0인 경우
  - 2개씩 원소를 비교할 때, 1에서 0으로 바뀌는 경우

0	0	0	1	1	0	0
---	---	---	---	---	---	---

└────────┘

 횟수: 1

# I 문제 풀이 핵심 아이디어

- 구체적인 알고리즘 예시) 모두 1로 만드는 경우
  - 첫 번째 원소가 0인 경우
  - 2개씩 원소를 비교할 때, 1에서 0으로 바뀌는 경우

0	0	0	1	1	0	0
---	---	---	---	---	---	---

└────────┘

 횟수: 1

# I 문제 풀이 핵심 아이디어

- 구체적인 알고리즘 예시) 모두 1로 만드는 경우
  - 첫 번째 원소가 0인 경우
  - 2개씩 원소를 비교할 때, 1에서 0으로 바뀌는 경우

0	0	0	1	1	0	0
---	---	---	---	---	---	---

└────────┘

 횟수: 1

# I 문제 풀이 핵심 아이디어

- 구체적인 알고리즘 예시) 모두 1로 만드는 경우
  - 첫 번째 원소가 0인 경우
  - 2개씩 원소를 비교할 때, 1에서 0으로 바뀌는 경우

0	0	0	1	1	0	0
---	---	---	---	---	---	---

└────────┘

 횟수: 2

# I 문제 풀이 핵심 아이디어

- 구체적인 알고리즘 예시) 모두 1로 만드는 경우
  - 첫 번째 원소가 0인 경우
  - 2개씩 원소를 비교할 때, 1에서 0으로 바뀌는 경우

0	0	0	1	1	0	0
---	---	---	---	---	---	---

└──────────┘

➡ **횟수: 2**

# | 소스코드

```
data = input()
count0 = 0 # 전부 0으로 바꾸는 경우
count1 = 0 # 전부 1로 바꾸는 경우

if data[0] == '1':
    count0 += 1
else:
    count1 += 1

for i in range(len(data) - 1):
    if data[i] != data[i + 1]:
        # 다음 수에서 1로 바뀌는 경우
        if data[i + 1] == '1':
            count0 += 1
        # 다음 수에서 0으로 바뀌는 경우
        else:
            count1 += 1

print(min(count0, count1))
```



# I 혼자 힘으로 풀어 보기

문제 제목: 등수 매기기

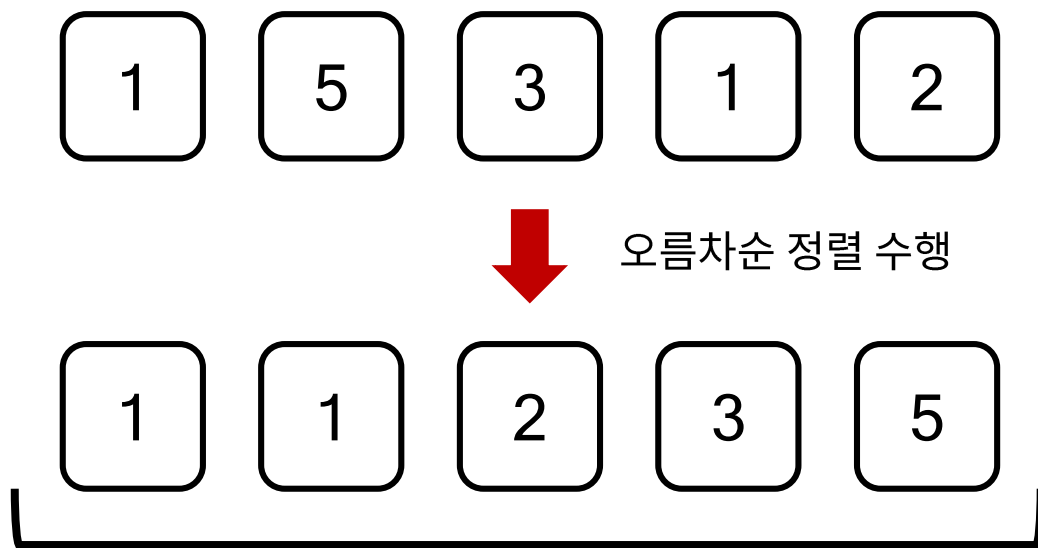
문제 난이도: 하(Easy)

문제 유형: 그리디

추천 풀이 시간: 20분

# I 문제 풀이 핵심 아이디어

- 예상된 등수와 실제 등수의 차이를 최소화해야 합니다.
- 이를 위해서, 예상된 등수를 오름차순으로 정렬하면 됩니다.



불만도의 합: 3

# | 소스코드

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

for _ in range(n):
    array.append(int(input()))

# 오름차순 정렬 수행
array.sort()

# 불만도의 합 계산
result = 0
for i in range(1, len(array) + 1):
    result += abs(i - array[i - 1])

print(result)
```

# I 혼자 힘으로 풀어 보기

문제 제목: 배

문제 난이도: 중(Medium)

문제 유형: 그리디

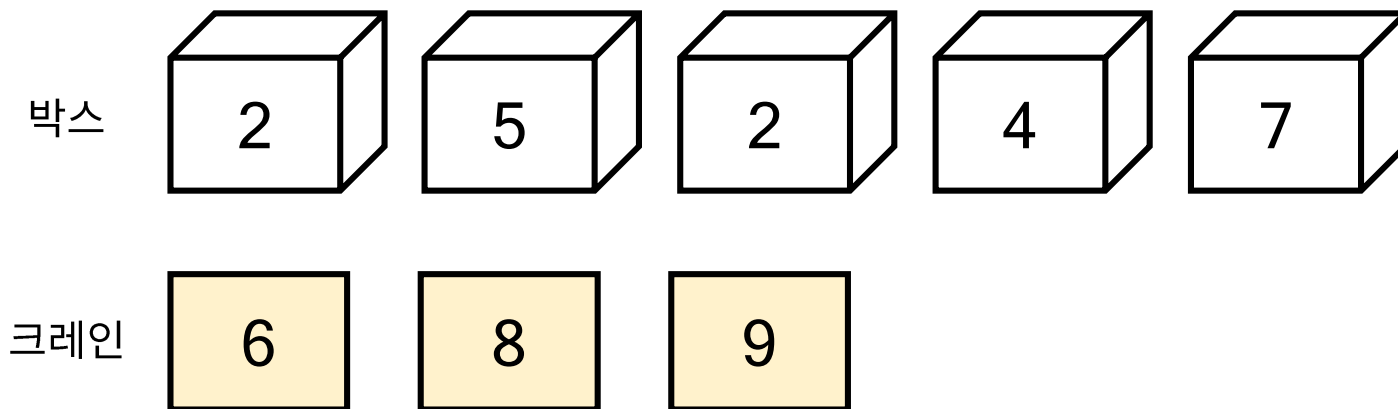
추천 풀이 시간: 35분

# I 문제 풀이 핵심 아이디어

- 모든 박스를 배로 옮기는데 드는 시간의 최솟값을 계산해야 합니다.
- 매 분마다, 모든 크레인에 대하여 옮길 수 있는 박스를 선택하여 옮기도록 합니다.
- 박스를 무게 기준으로 내림차순 정렬한 뒤에, 무거운 것부터 옮기도록 하면 됩니다.
- 시간 복잡도  $O(NM)$ 에 문제를 해결할 수 있습니다.

# I 문제 풀이 핵심 아이디어

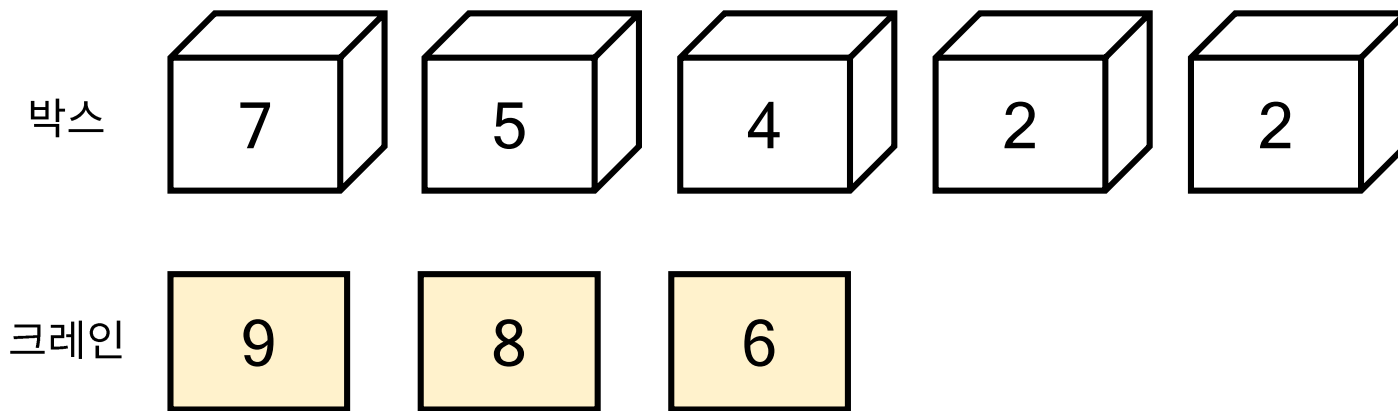
- 크레인과 박스를 내림차순 정렬합니다.
- 매 분마다, 모든 크레인에 대하여 옮길 수 있는 박스를 선택하여 옮기도록 합니다.



# I 문제 풀이 핵심 아이디어

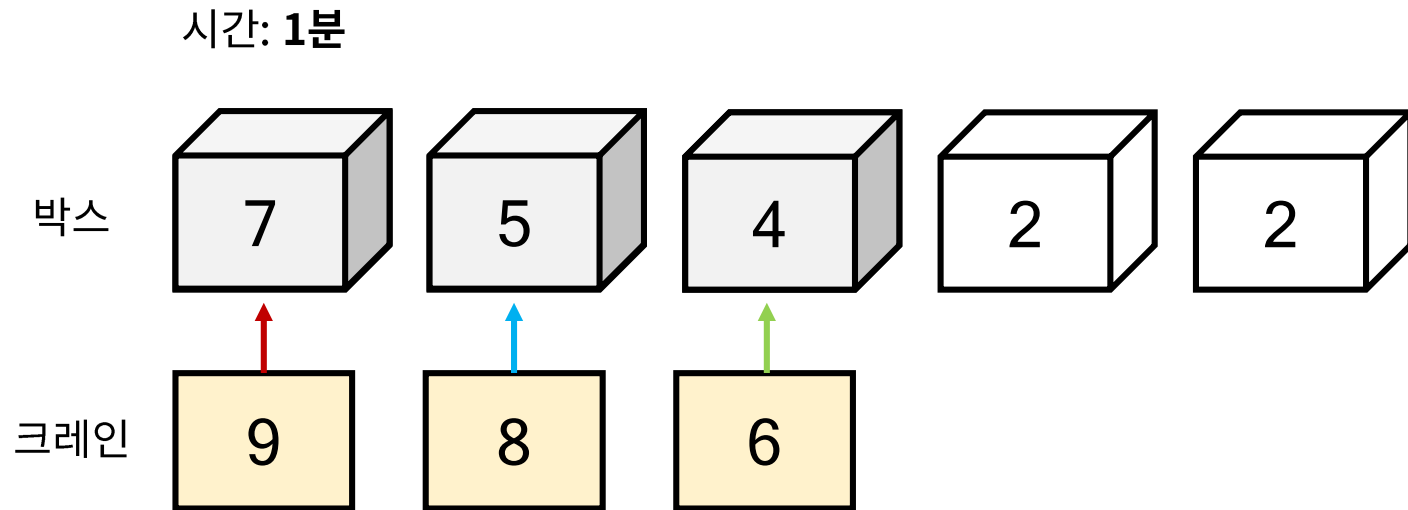
- 크레인과 박스를 내림차순 정렬합니다.
- 매 분마다, 모든 크레인에 대하여 옮길 수 있는 박스를 선택하여 옮기도록 합니다.

(내림차순 정렬 수행)



# I 문제 풀이 핵심 아이디어

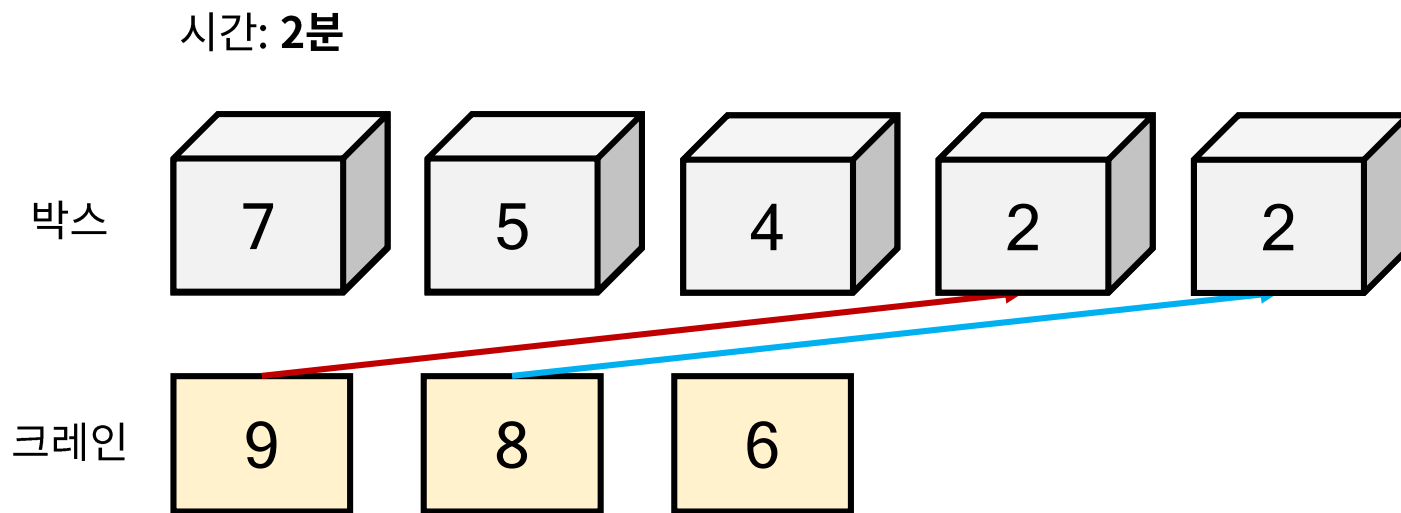
- 크레인과 박스를 내림차순 정렬합니다.
- 매 분마다, 모든 크레인에 대하여 옮길 수 있는 박스를 선택하여 옮기도록 합니다.





# I 문제 풀이 핵심 아이디어

- 크레인과 박스를 내림차순 정렬합니다.
- 매 분마다, 모든 크레인에 대하여 옮길 수 있는 박스를 선택하여 옮기도록 합니다.



➡ 정답: 2분

## | 소스코드

```
import sys

n = int(input())
cranes = list(map(int, input().split()))

m = int(input())
boxes = list(map(int, input().split()))

# 모든 박스를 옮길 수 없는 경우
if max(cranes) < max(boxes):
    print(-1)
    sys.exit()

# 각 크레인이 현재 옮겨야 하는 박스의 번호 (0부터 시작)
positions = [0] * n
# 각 박스를 옮겼는지의 여부
checked = [False] * m
# 최적의 해를 구해야 하므로, 내림차순 정렬
cranes.sort(reverse=True)
boxes.sort(reverse=True)

result = 0
count = 0
```

```
while True:
    if count == len(boxes): # 박스를 다 옮겼다면 종료
        break
    for i in range(n): # 모든 크레인에 대하여 각각 처리
        while positions[i] < len(boxes):
            # 아직 안 옮긴 박스 중에서, 옮길 수 있는 박스를 만날 때까지 반복
            if not checked[positions[i]] and cranes[i] >= boxes[positions[i]]:
                checked[positions[i]] = True
                positions[i] += 1
                count += 1
                break
            positions[i] += 1
        result += 1

print(result)
```