Chapter 03.
기본 정렬 알고리즘
핵심 유형 문제풀이

핵심 유형 문제풀이 | 다양한 문제를 접하며 코딩 테스트에 익숙해지기

강사 나동빈

Chapter 03. 기본 정렬 알고리즘

핵심 유형 문제풀이

선 긋기: https://www.acmicpc.net/problem/2170

혼자 힘으로 풀어보기

문제 제목: 선 긋기

문제 난이도: ★★★☆☆

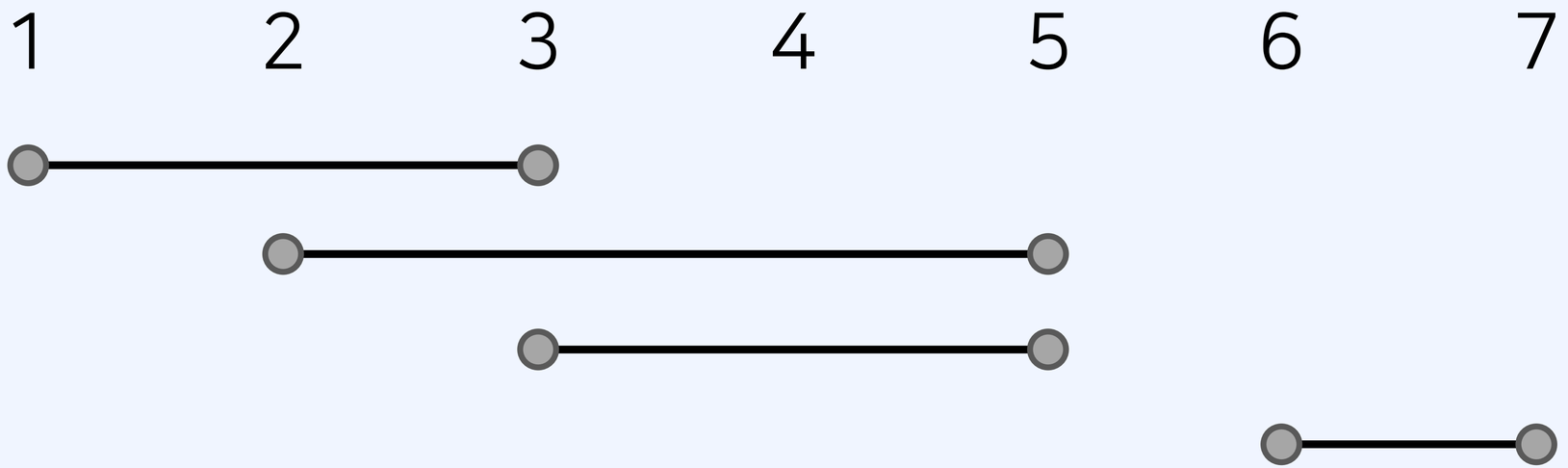
문제 유형: 정렬, 그리디

추천 풀이 시간: 50분

선 긋기: https://www.acmicpc.net/problem/2170

문제 풀이 핵심 아이디어

- 본 문제에서 주어진 예시를 그림으로 표현하면 다음과 같다.

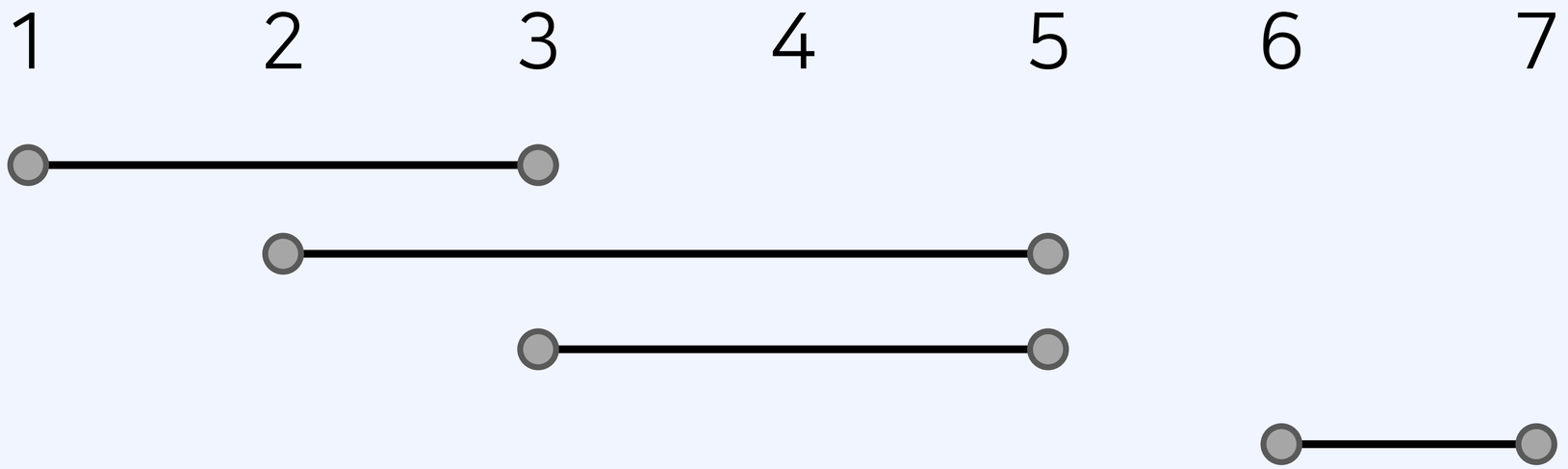


- 본 예시에서는 $[1, 5]$와 $[6, 7]$에 선을 긋게 된다.
- 따라서 총 $5 = 4 + 1$의 길이로 충분하다.

선 긋기: https://www.acmicpc.net/problem/2170

문제 풀이 핵심 아이디어

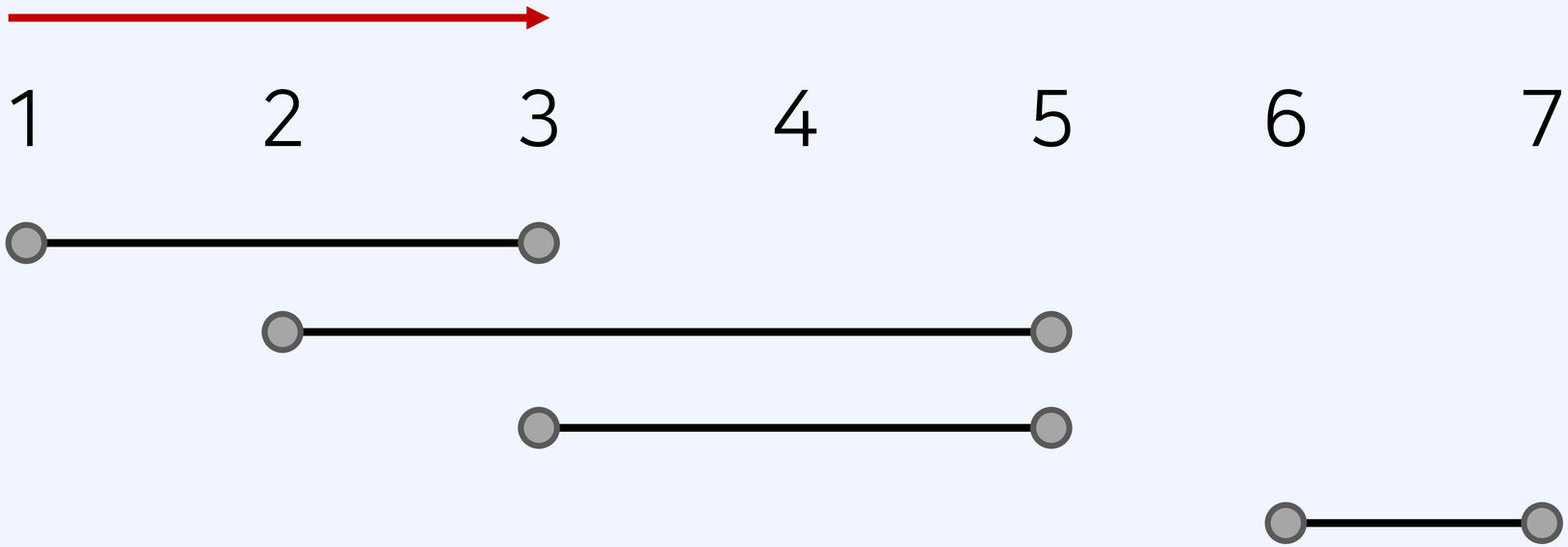
- **문제 아이디어**: 왼쪽에서부터 “펜을 떼지 않고, 최대한 쭉 선을 긋는다.”
- **데이터 예시**: (1, 3), (2, 5), (3, 5), (6, 7)
- 일단 전체 원소를 **시작점**을 기준으로 오름차순 정렬을 수행한다.



선 긋기: https://www.acmicpc.net/problem/2170

문제 풀이 핵심 아이디어

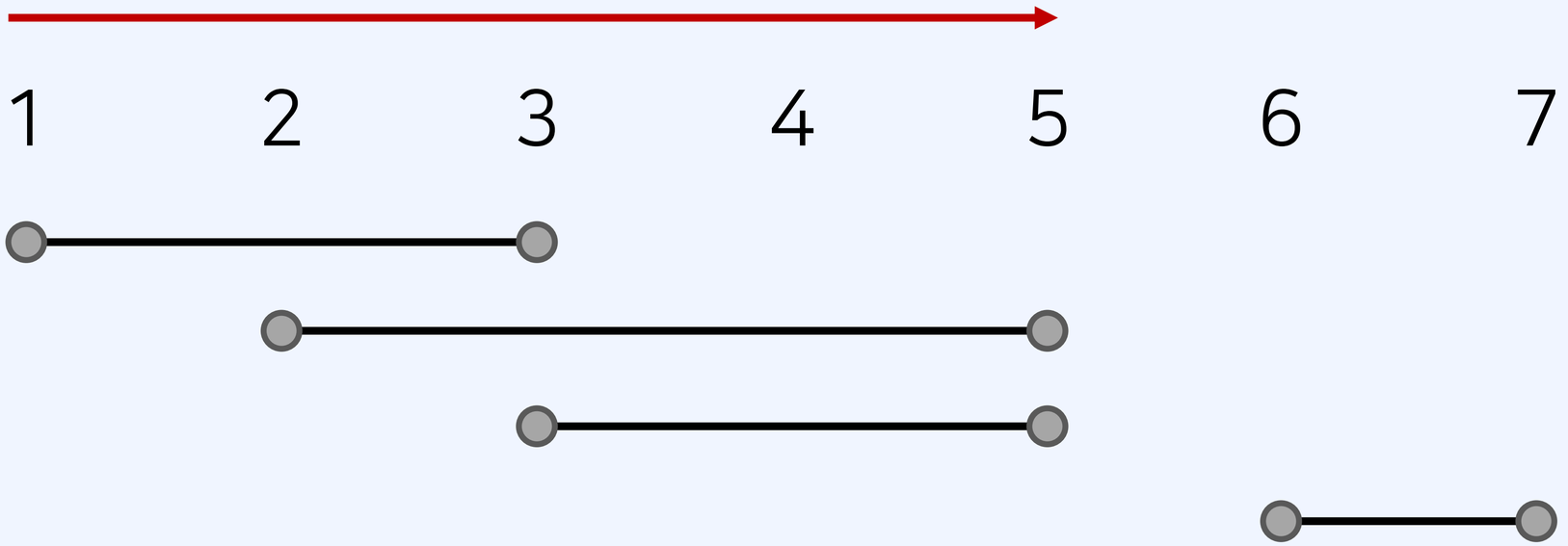
- **문제 아이디어**: 왼쪽에서부터 “펜을 떼지 않고, 최대한 쭉 선을 긋는다.”
- **데이터 예시**: (1, 3), (2, 5), (3, 5), (6, 7)
- 먼저 1부터 시작해 3까지 선을 긋는다. (결과 펜 위치 = 3)



선 긋기: https://www.acmicpc.net/problem/2170

문제 풀이 핵심 아이디어

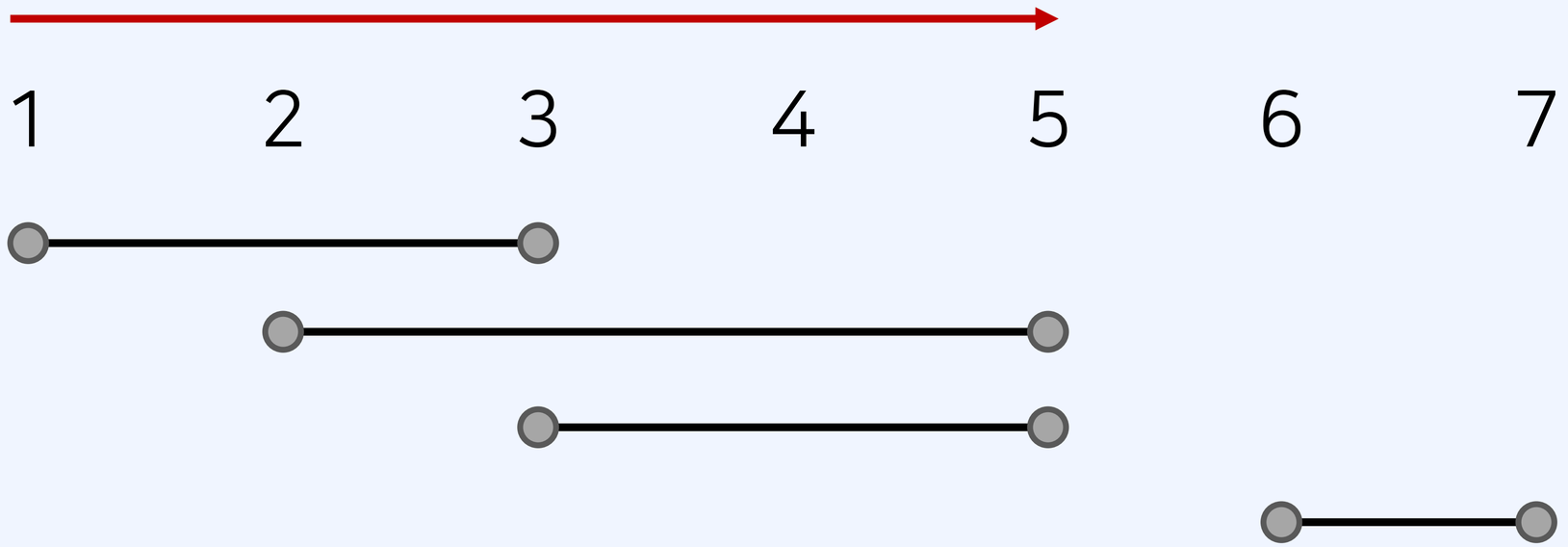
- **문제 아이디어**: 왼쪽에서부터 "펜을 떼지 않고, 최대한 쭉 선을 긋는다."
- **데이터 예시**: (1, 3), (2, 5), (3, 5), (6, 7)
- 현재 펜의 위치 3이 2보다 크거나 같으므로, 선을 쭉 긋는다. (결과 펜 위치 = 5)



선 긋기: https://www.acmicpc.net/problem/2170

문제 풀이 핵심 아이디어

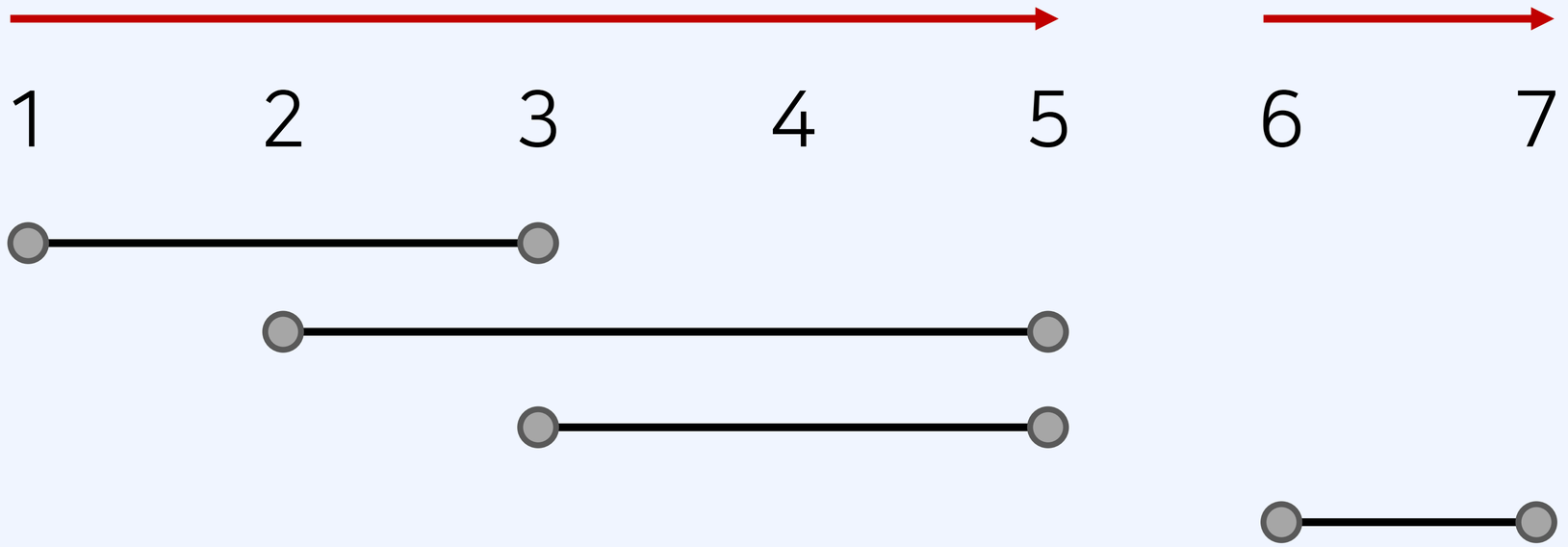
- **문제 아이디어**: 왼쪽에서부터 “펜을 떼지 않고, 최대한 쭉 선을 긋는다.”
- **데이터 예시**: (1, 3), (2, 5), (3, 5), (6, 7)
- 현재 펜의 위치 5이 3보다 크거나 같으므로, 선을 쭉 긋는다. (결과 펜 위치 = 5)



선 긋기: https://www.acmicpc.net/problem/2170

문제 풀이 핵심 아이디어

- **문제 아이디어**: 왼쪽에서부터 “펜을 떼지 않고, 최대한 쭉 선을 긋는다.”
- **데이터 예시**: (1, 3), (2, 5), (3, 5), (6, 7)
- 현재 펜의 위치 5가 6보다 작으므로, 6부터 다시 선을 긋는다. (결과 펜 위치 = 7)



선 긋기: https://www.acmicpc.net/problem/2170

소스 코드


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

n = int(input()) # 선의 개수 N
arr = [] # 모든 선의 정보를 담는 배열
for i in range(n):
    x, y = map(int, input().split())
    arr.append((x, y)) # (시작점, 끝점)
```

```python
arr.sort() # 시작점을 기준으로 각 선 정렬
result = 0
start, current = arr[0] # 첫 번째 선을 따라 현재 펜을 이동하기
for line in arr: # 하나씩 선들을 확인하며
    x, y = line
    if current >= x: # 현재 펜이 시작점보다 더 크다면
        current = max(current, y) # 최대한 펜을 쭉 긋기
    else: # 현재 펜이 시작점보다 작다면 (새로 그어야 한다면)
        result += current - start
        start = x # 펜을 이용해 새로 선을 긋기 시작한 점
        current = y # 현재 펜의 위치(끝점)
result += current - start

print(result)
```

두 수의 합: https://www.acmicpc.net/problem/3273

혼자 힘으로 풀어보기

문제 제목: 두 수의 합

문제 난이도: ★★☆☆☆

문제 유형: 정렬, 투 포인터

추천 풀이 시간: 30분

두 수의 합: https://www.acmicpc.net/problem/3273

문제 풀이 핵심 아이디어

- 본 문제에서는 N개의 서로 다른 양의 정수가 입력으로 주어진다.
- 이때 서로 다른 두 양의 정수를 골라서, 합이 x가 되는 경우의 수를 세어야 한다.
- 데이터의 개수(N)가 최대 100,000이므로, $O(NlogN)$의 복잡도로 문제를 해결해야 한다.

두 수의 합: https://www.acmicpc.net/problem/3273

문제 풀이 핵심 아이디어

- 본 문제는 투 포인터 알고리즘을 이용해 해결할 수 있다.
- 다음과 같이 원소가 입력된 상황을 확인해 보자.

$$arr = [5, 12, 7, 10, 9, 1, 2, 3, 11]$$

- 원소를 오름차순 정렬한 결과는 다음과 같다.

$$sorted = [1, 2, 3, 5, 7, 9, 10, 11, 12]$$

- 모든 정수가 양의 정수이며, **투 포인터**를 이용해 문제를 해결할 수 있다.

두 수의 합: https://www.acmicpc.net/problem/3273

문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.
- **초기화**: 시작점($start$) = 0, 끝점(end) = $N - 1$
- 정렬이 되어 있기 때문에, 다음이 성립한다.
- 시작 위치를 1만큼 증가시키면, $sorted[s] + sorted[e]$가 증가한다.
- 끝 위치를 1만큼 감소시키면, $sorted[s] + sorted[e]$가 감소한다.

[핵심] 이러한 방식으로 현재의 합($sorted[s] + sorted[e]$)과 x를 매번 비교한다.

문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.

  1) 현재의 합이 x보다 작거나 같으면 시작점($start$)을 1 증가

  2) 현재의 합이 x보다 크면 끝점(end)을 1 감소

- $x = 13$, 현재의 합($sorted[s] + sorted[e]$) = 13, **카운트($count$) = 1**

두 수의 합: https://www.acmicpc.net/problem/3273

문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.

  1) 현재의 합이 x보다 작거나 같으면 시작점($start$)을 1 증가

  2) 현재의 합이 x보다 크면 끝점(end)을 1 감소

- $x = 13$, 현재의 합($sorted[s] + sorted[e]$) = 14, **카운트($count$) = 1**

두 수의 합: https://www.acmicpc.net/problem/3273

문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.

  1) 현재의 합이 x보다 작거나 같으면 시작점($start$)을 1 증가

  2) 현재의 합이 x보다 크면 끝점(end)을 1 감소

- $x = 13$, 현재의 합($sorted[s] + sorted[e]$) = 13, **카운트($count$) = 2**

두 수의 합: https://www.acmicpc.net/problem/3273

문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.

  1) 현재의 합이 x보다 작거나 같으면 시작점($start$)을 1 증가

  2) 현재의 합이 x보다 크면 끝점(end)을 1 감소

- $x = 13$, 현재의 합($sorted[s] + sorted[e]$) = 14, **카운트($count$) = 2**

두 수의 합: https://www.acmicpc.net/problem/3273

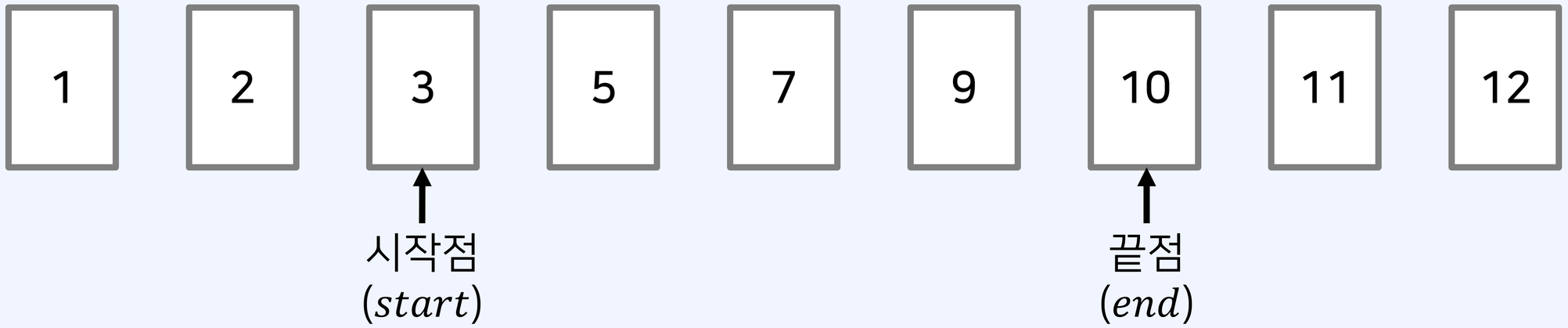
문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.

  1) 현재의 합이 x보다 작거나 같으면 시작점($start$)을 1 증가

  2) 현재의 합이 x보다 크면 끝점(end)을 1 감소

- $x = 13$, 현재의 합($sorted[s] + sorted[e]$) = 13, **카운트($count$) = 3**



두 수의 합: https://www.acmicpc.net/problem/3273

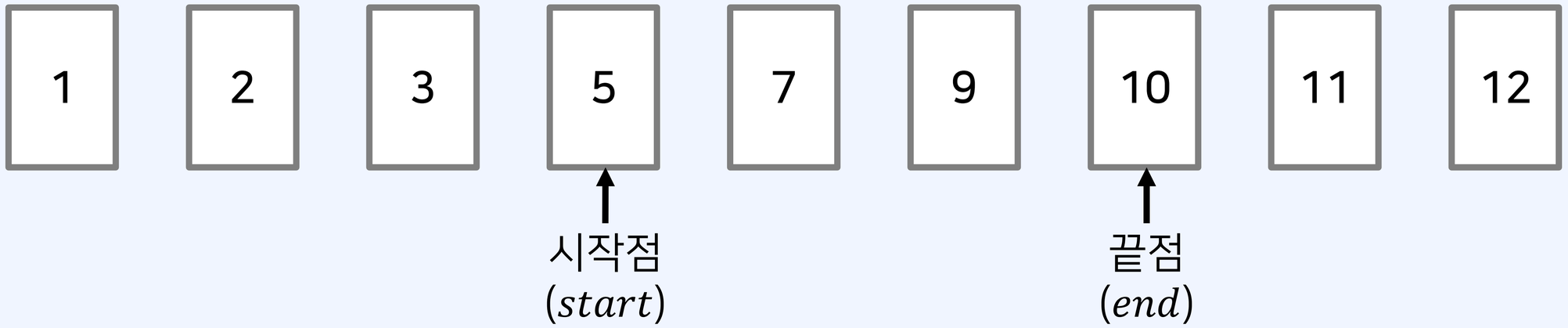
문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.

  1) 현재의 합이 x보다 작거나 같으면 시작점($start$)을 1 증가

  2) 현재의 합이 x보다 크면 끝점(end)을 1 감소

- $x = 13$, 현재의 합($sorted[s] + sorted[e]$) = 15, **카운트($count$) = 3**



두 수의 합: https://www.acmicpc.net/problem/3273

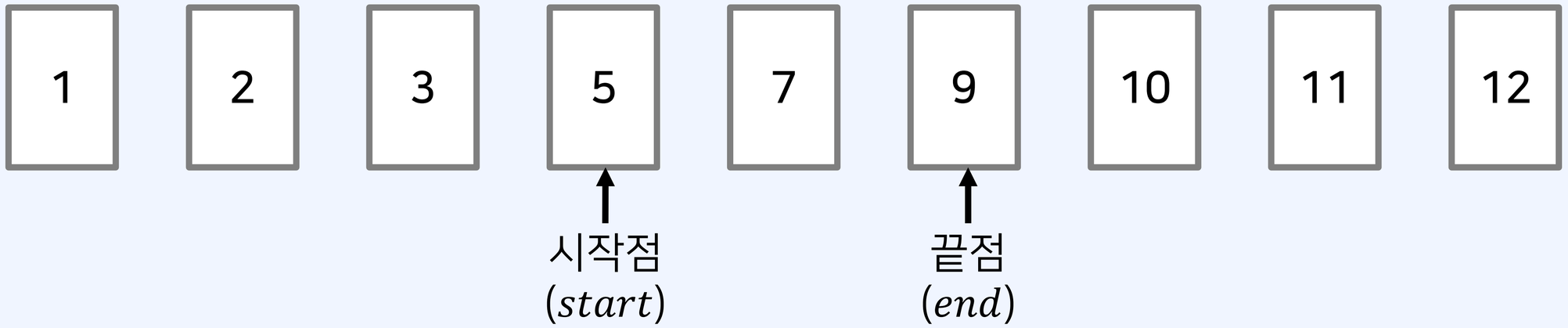
문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.

  1) 현재의 합이 x보다 작거나 같으면 시작점($start$)을 1 증가

  2) 현재의 합이 x보다 크면 끝점(end)을 1 감소

- $x = 13$, 현재의 합($sorted[s] + sorted[e]$) = 14, **카운트($count$) = 3**



두 수의 합: https://www.acmicpc.net/problem/3273

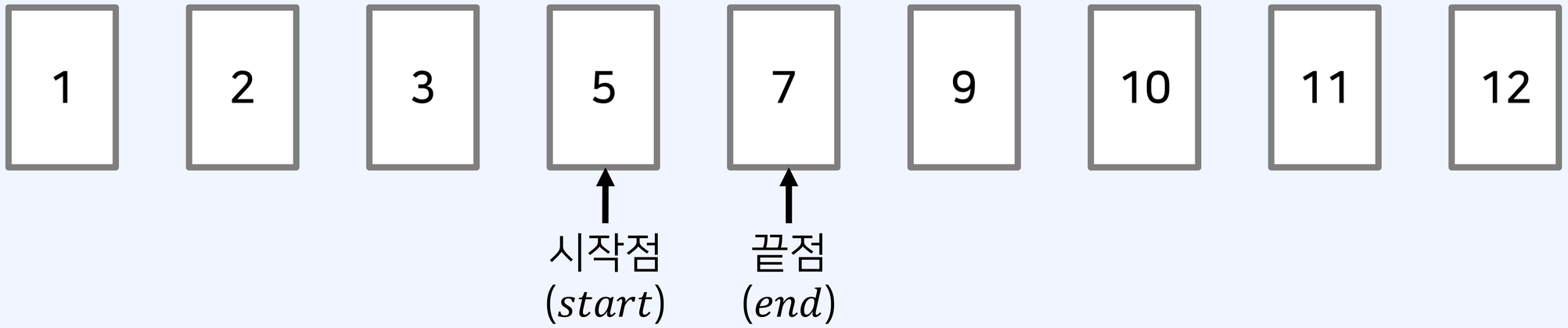
문제 풀이 핵심 아이디어

- 투 포인터는 시작점($start$)와 끝점(end)을 이용해 문제를 해결하는 알고리즘이다.

  1) 현재의 합이 x보다 작거나 같으면 시작점($start$)을 1 증가

  2) 현재의 합이 x보다 크면 끝점(end)을 1 감소

- $x = 13$, 현재의 합($sorted[s] + sorted[e]$) = 12, **카운트($count$) = 3**



두 수의 합: https://www.acmicpc.net/problem/3273

소스 코드


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

n = int(input()) # 원소의 개수 N
# 전체 원소 입력
arr = list(map(int, input().split()))
x = int(input()) # X 입력

arr.sort() # 오름차순 정렬
```

```python
result = 0
start = 0 # 시작점(start)
end = n - 1 # 끝점(end)

while start < end:
    current = arr[start] + arr[end]
    if current == x: # X를 찾은 경우
        result += 1 # 카운트
        start += 1
    elif current < x: # 현재 합이 X보다 작은 경우
        start += 1 # 합을 증가시키기
    elif current > x: # 현재 합이 X보다 큰 경우
        end -= 1 # 합을 감소시키기

print(result)
```

문제 풀이 핵심 아이디어

- 본 문제에서 **난이도**란, "인접한 두 통나무 간의 높이의 차의 최댓값"이다.

[가장 간단한 접근 방법]

- 우선 난이도를 최소로 만들기 위해서는, 오름차순 정렬을 시키면 될 것 같다.
- 하지만 $arr[0]$과 $arr[N-1]$도 인접한 것으로 판단되므로, 단순히 정렬로는 해결할 수 없다.

통나무 건너뛰기: https://www.acmicpc.net/problem/11497

문제 풀이 핵심 아이디어

[문제 해결 아이디어]

- 정규 분포 형태를 떠올리면 된다.
- 중간이 가장 높고, **끝으로 갈수록 작아지는 형태**로 만들자.

  따라서 가장 높은 통나무를 중간에 두고, 차례대로 왼쪽, 오른쪽에 배치하는 것을 반복한다.

통나무 건너뛰기: https://www.acmicpc.net/problem/11497

문제 풀이 핵심 아이디어

- 예를 들어 $[2, 4, 5, 7, 9]$가 입력으로 들어온 경우를 가정해 보자.
- 먼저 **내림차순 정렬**을 수행하여 $[9, 7, 5, 4, 2]$가 된다.
- 이후에 가장 중간부터 차례대로 좌우로 원소를 배치하면 된다.

1)

0	1	2	3	4
		9		

2)

0	1	2	3	4
	7	9		

3)

0	1	2	3	4
	7	9	5	

4)

0	1	2	3	4
4	7	9	5	

5)

0	1	2	3	4
4	7	9	5	2

통나무 건너뛰기: https://www.acmicpc.net/problem/11497

소스 코드


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

# 테스트 케이스를 하나씩 확인하며
for test_case in range(int(input())):
    n = int(input()) # 통나무의 개수 N
    arr = list(map(int, input().split())) # 전체 통나무 배열 입력받기
    arr.sort(reverse=True) # 내림차순 정렬

    result = [0] * n # 재배치 결과
    result[n // 2] = arr[0] # 가장 중간에 가장 큰 수 배치
    for i in range(1, n):
        if i % 2 == 1: # 홀수인 경우 왼쪽에 배치
            result[n // 2 - i // 2 - 1] = arr[i]
        else: # 짝수인 경우 오른쪽에 배치
            result[n // 2 + i // 2] = arr[i]

    max_dif = 0 # 인접한 두 통나무 간의 높이의 차의 최댓값
    for i in range(n):
        dif = abs(result[i] - result[(i + 1) % n]) # 차이 계산
        max_dif = max(max_dif, dif)
    print(max_dif)
```

문제 풀이 핵심 아이디어

- 조금 더 간단하게 문제를 풀 수 있다.
- 예를 들어 원소가 7개이고, 정렬된 결과 배열을 r이라고 하자.
- 이때는 다음과 같이 배치해야 한다.

0	1	2	3	4	5	6
$r[5]$	$r[3]$	$r[1]$	$r[0]$	$r[2]$	$r[4]$	$r[6]$

- 이때 정답은 **다음 중에서 가장 큰 것**이 된다.

  $abs(r[0]- r[2])$, $abs(r[1]- r[3])$, $abs(r[2]- r[4])$, $abs(r[3]- r[5])$, $abs(r[4]- r[6])$
- 따라서 단순히 오름차순 정렬을 수행한 뒤에, 2칸 옆에 있는 원소와 비교하면 된다.

통나무 건너뛰기: https://www.acmicpc.net/problem/11497

소스 코드


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

# 테스트 케이스를 하나씩 확인하며
for test_case in range(int(input())):
    n = int(input()) # 통나무의 개수 N
    arr = list(map(int, input().split())) # 전체 통나무 배열 입력받기
    arr.sort() # 정렬 수행

    max_dif = 0 # 인접한 두 통나무 간의 높이의 차의 최댓값
    for i in range(n - 2):
        dif = abs(arr[i] - arr[i + 2]) # 차이 계산
        max_dif = max(max_dif, dif)
    print(max_dif)
```