Chapter 11.
탐욕 알고리즘
핵심 유형 문제풀이

핵심 유형 문제풀이 | 다양한 문제를 접하며 코딩 테스트에 익숙해지기

강사 나동빈

Chapter 11. 탐욕 알고리즘

핵심 유형 문제풀이

전화번호 목록: https://www.acmicpc.net/problem/5052

혼자 힘으로 풀어보기

문제 제목: 전화번호 목록

문제 난이도: ★★★☆☆

문제 유형: 그리디

추천 풀이 시간: 50분

전화번호 목록: https://www.acmicpc.net/problem/5052

문제 풀이 핵심 아이디어

[문제 설명]

- 한 번호가 다른 번호의 접두어인 경우가 있는지 찾는 문제다.
- 전화번호부가 있을 때, 접두어 관계가 존재하는지 찾는 프로그램을 작성한다.

전화번호 목록: https://www.acmicpc.net/problem/5052

문제 풀이 핵심 아이디어

[문제 해결 아이디어]

- 정렬된 결과가 다음과 같다고 생각해 보자.

  정렬된 결과: [456, 4567, 4576]

- 접두어를 형성하는 경우, 정렬된 이후에 인접한 문자열을 봤을 때 접두어 관계임을 알 수 있다.
- 따라서 다음의 알고리즘으로 문제를 해결할 수 있다.

1. 먼저 전화번호에 대하여 **오름차순 정렬**을 수행한다.
2. 인접한 두 개의 문자열을 확인하며, 접두어 관계인지 확인한다.

전화번호 목록: https://www.acmicpc.net/problem/5052

Ch11. 탐욕 알고리즘 소스 코드
핵심 유형 문제풀이

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

# 테스트 케이스만큼 반복
for tc in range(int(input())):
    n = int(input()) # 전화번호의 개수
    books = [] # 전화번호부
    for i in range(n):
        books.append(input().strip())
    books.sort() # 문자열 기준으로 오름차순 정렬
    result = False # 접두어가 존재하는지 여부
    for i in range(n - 1):
        # 앞 문자열이 뒤 문자열보다 짧은 경우에만 확인
        if len(books[i]) < len(books[i + 1]):
            # 앞 문자열이 뒤 문자열의 접두어라면
            if books[i] == books[i + 1][:len(books[i])]:
                result = True
                break
    if result: print("NO") # 접두어 관계 있음
    else: print("YES") # 접두어 관계 없음
```

혼자 힘으로 풀어보기

문제 제목: 합이 0인 네 정수

문제 난이도: ★★★☆☆

문제 유형: 그리디

추천 풀이 시간: 50분

문제 풀이 핵심 아이디어

- 단순한 접근을 고려해보자.
- 4중 반복문을 이용해서 $O(N^4)$의 알고리즘을 설계할 수 있다.
- 하지만 배열의 크기 N이 최대 4,000이므로, 이 경우 **시간 초과**를 받는다.

문제 풀이 핵심 아이디어

[문제 해결 아이디어]

- 시간을 줄이기 위하여 메모리를 사용해 미리 계산 결과를 기록해 놓는다.
- 배열 A, B, C, D가 있을 때, 배열 A와 B만을 사용해서 **"가능한 합"의 개수**들을 미리 구한다.
- 예를 들어 $A = [3, 4]$, $B = [4, 3]$일 때, 각 합마다 개수를 표현하면 다음과 같다.

$$counter = \{6: 1, 7: 2, 8: 1\}$$

- 이제, C와 B만을 이용해서 만들 수 있는 “가능한 합”이 $counter$에 존재한다면, 개수를 센다.
- 시간 초과를 받지 않기 위해, 가능하면 $PyPy3$으로 코드를 제출한다.

합이 0인 네 정수: https://www.acmicpc.net/problem/7453


```python
# 각 배열에 들어갈 원소의 개수(N) 입력
n = int(input())
A = []
B = []
C = []
D = []

# 4개의 배열(A, B, C, D)에 모든 원소 입력
for i in range(n):
    a, b, c, d = map(int, input().split())
    A.append(a)
    B.append(b)
    C.append(c)
    D.append(d)
```

```python
# A와 B만을 고려하기
counter = {}
for i in range(n):
    for j in range(n):
        #  "가능한 합"의 개수들을 미리 구해 놓기
        sum = A[i] + B[j]
        if sum in counter: counter[sum] += 1
        else: counter[sum] = 1

# C와 D만을 고려해 "가능한 합"에 해당하는지 확인
result = 0
for i in range(n):
    for j in range(n):
        sum = C[i] + D[j]
        if -sum in counter:
            result += counter[-sum]

print(result)
```

철로: https://www.acmicpc.net/problem/13334

문제 제목: 철로

문제 난이도: ★★★☆☆

문제 유형: 그리디

추천 풀이 시간: 50분

철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

- 일단 사람들의 정보를 **(시작, 끝)**으로 고려하자.
- 길이가 d인 철로 선분을 왼쪽에서부터 오른쪽으로 이동한다고 가정한다.

[문제 해결 아이디어]

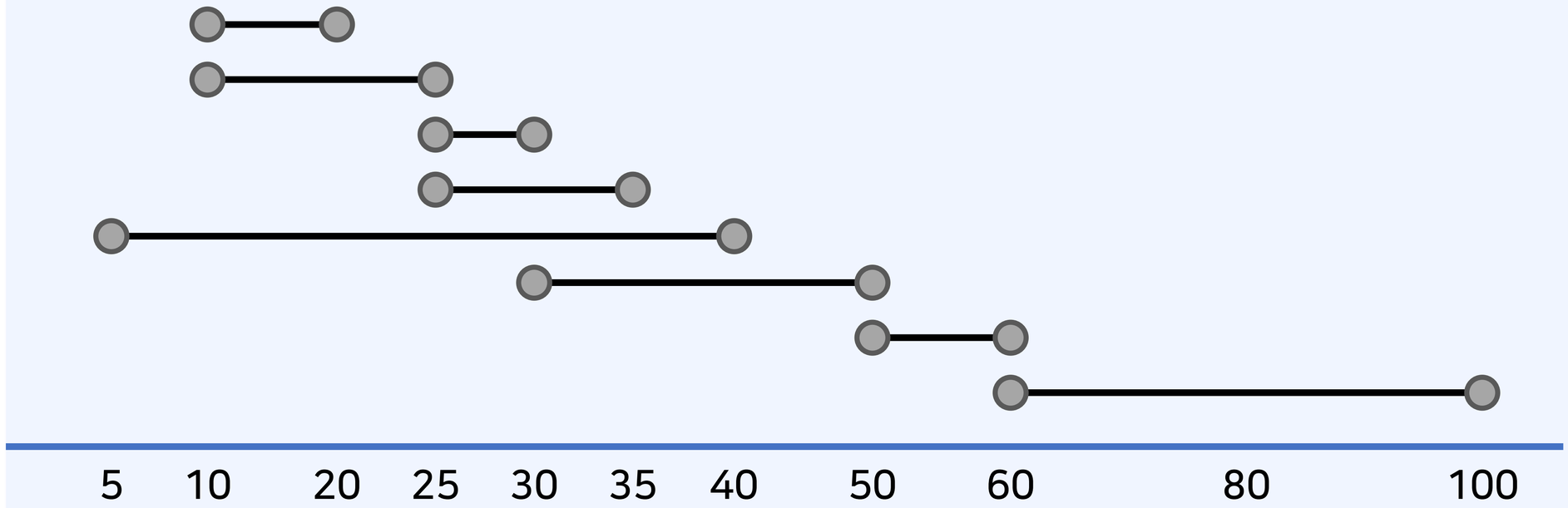
1. 모든 선분 **(시작, 끝)**이 있을 때, "끝"을 기준으로 오름차순 정렬을 수행한다.
2. 각 선분을 하나씩 확인하며, 아래의 알고리즘을 수행한다.
   1) 해당 선분이 d보다 길다면 무시한다.
   2) 해당 선분이 끝점에 닿도록 철로 선분을 이동시키고, **힙**에 시작점을 삽입한다.
   3) 철로 선분이 이동함에 따라서, 철로를 벗어난 시작점을 **힙**에서 추출한다.
   4) 현재 **"힙의 크기"**를 계산한다. (철로가 포함할 수 있는 선분의 개수)

- 결과적으로, "힙의 크기"의 **최댓값**을 구하면 정답이다.

철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

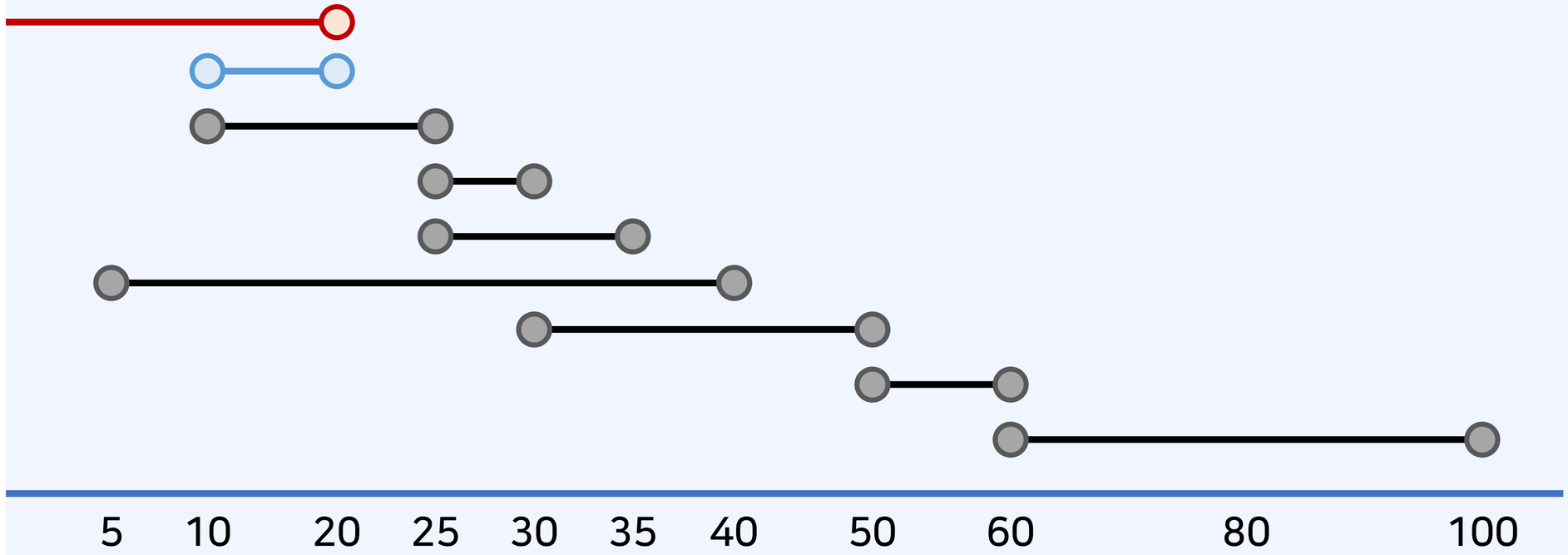
- 문제에서 주어진 예시를 확인해 보자. $(d = 30)$
- 가장 먼저 각 선분에 대하여 **"끝"**을 기준으로 오름차순 정렬을 수행한다.



철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

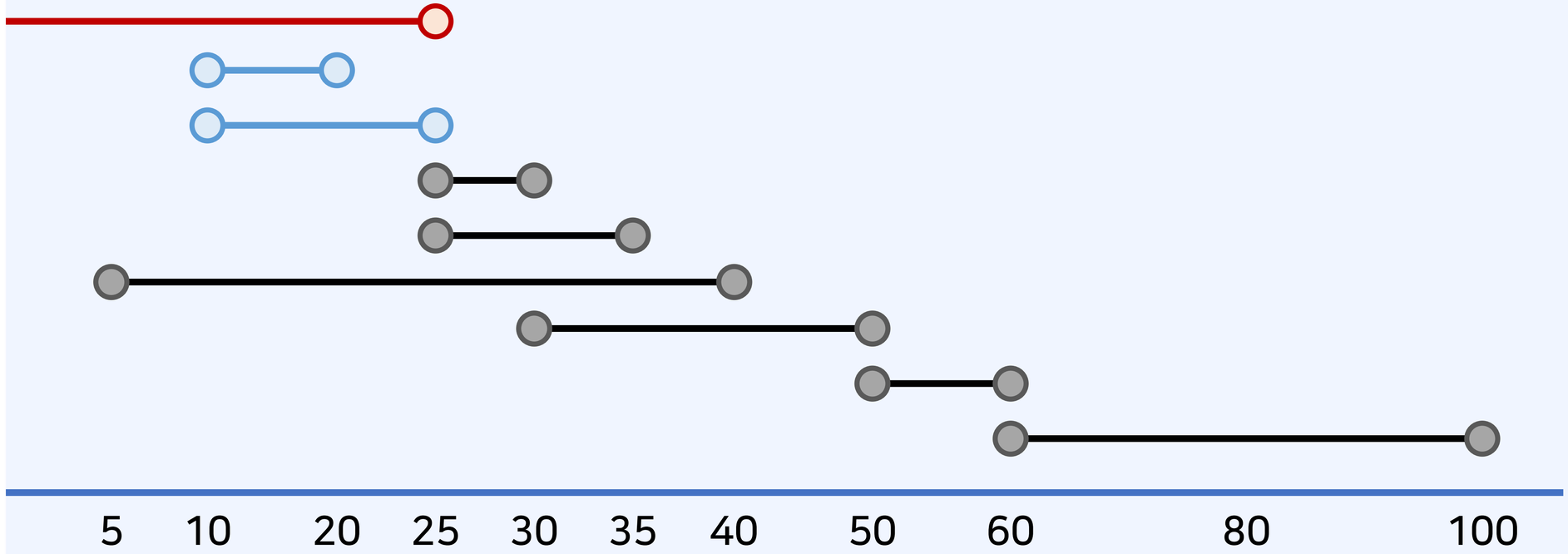
- 문제에서 주어진 예시를 확인해 보자. ($d = 30$)
- **첫 번째 선분**을 확인하고, 우선순위 큐(힙)에 삽입한다.



철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

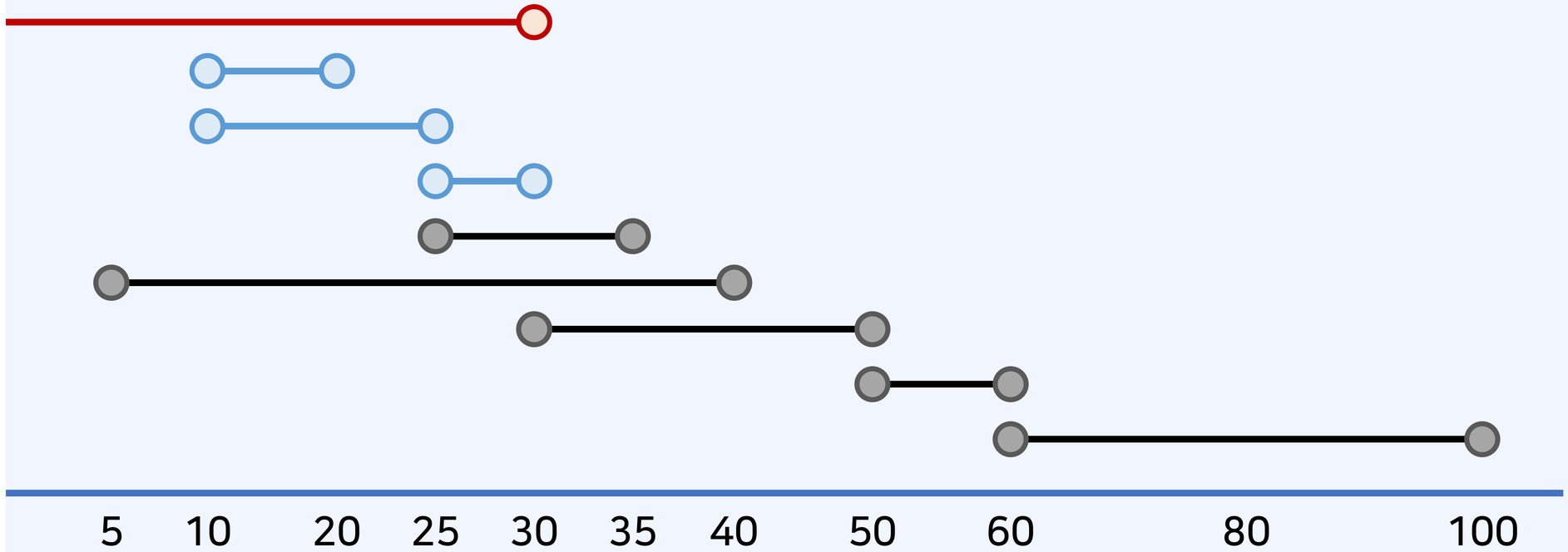
- 문제에서 주어진 예시를 확인해 보자. ($d = 30$)
- **두 번째 선분**을 확인하고, 우선순위 큐(힙)에 삽입한다.



철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

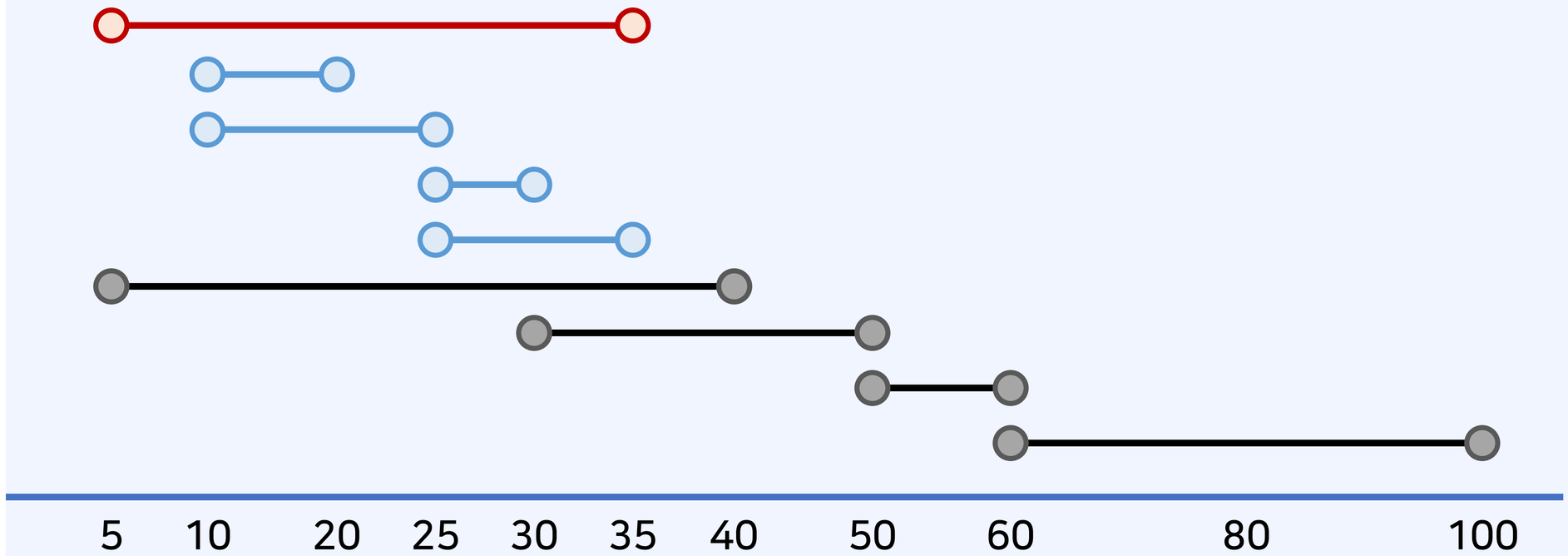
- 문제에서 주어진 예시를 확인해 보자. ($d = 30$)
- **세 번째 선분**을 확인하고, 우선순위 큐(힙)에 삽입한다.



철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

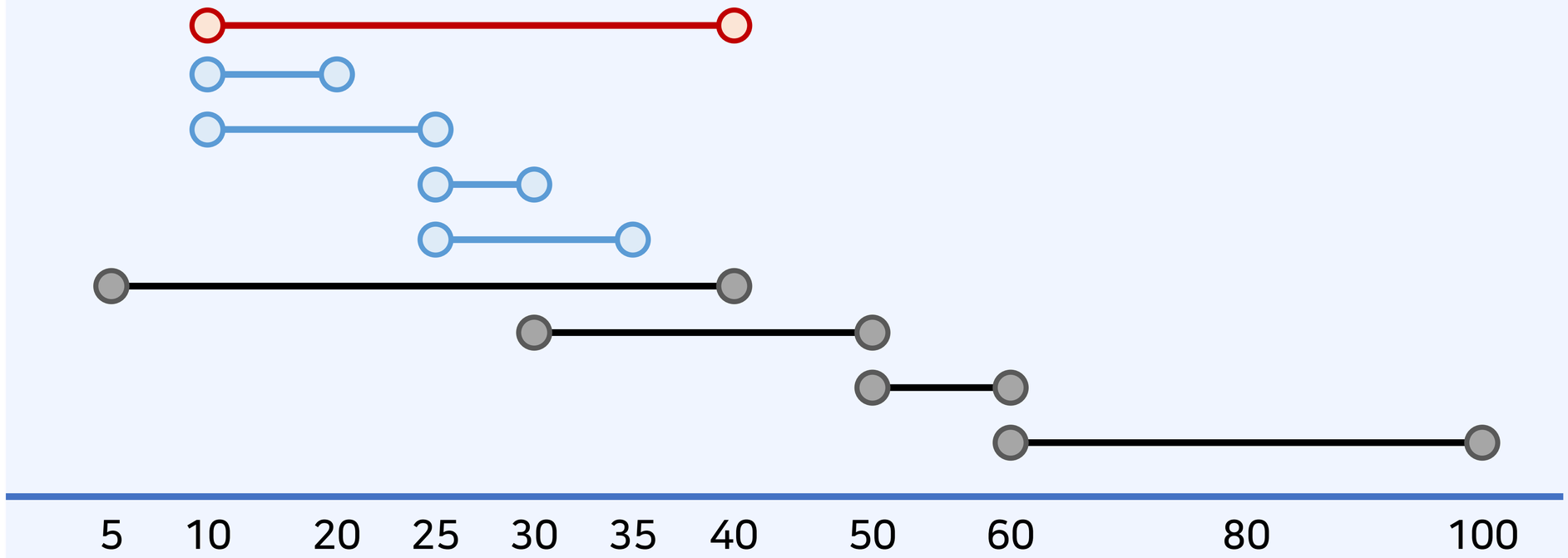
- 문제에서 주어진 예시를 확인해 보자. ($d = 30$)
- **네 번째 선분**을 확인하고, 우선순위 큐(힙)에 삽입한다.



철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

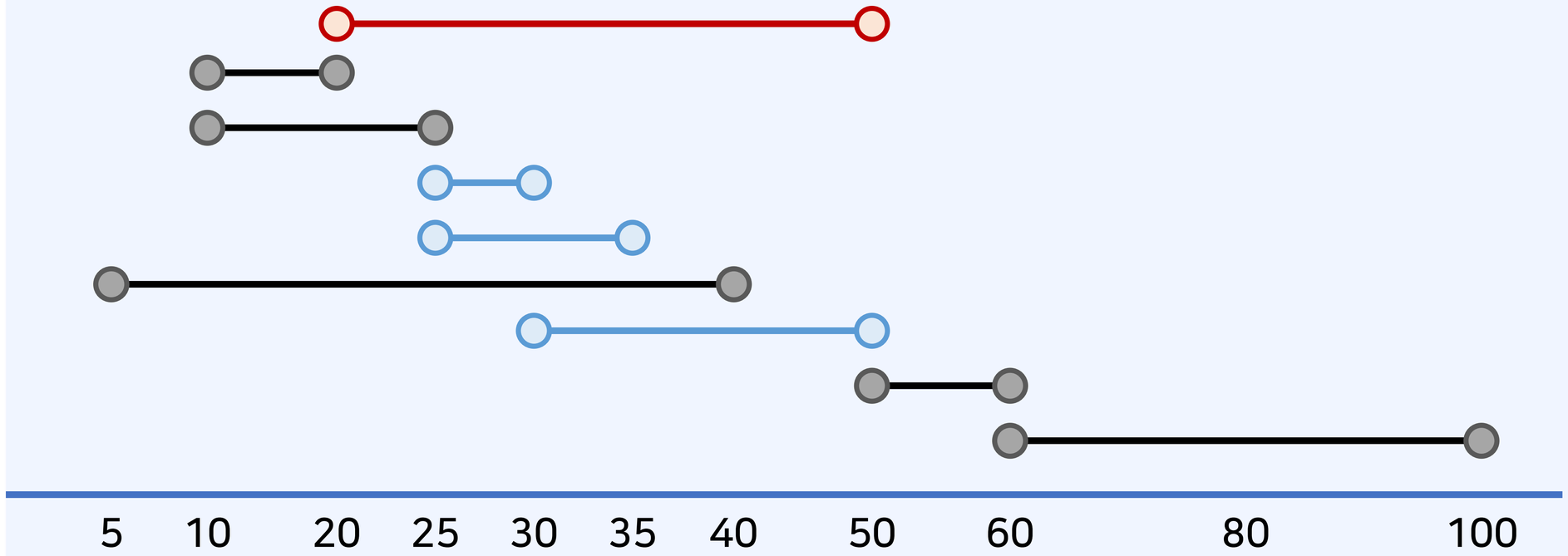
- 문제에서 주어진 예시를 확인해 보자. $(d = 30)$
- **다섯 번째 선분**은 철로의 길이(d)보다 길기 때문에 무시한다.



철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

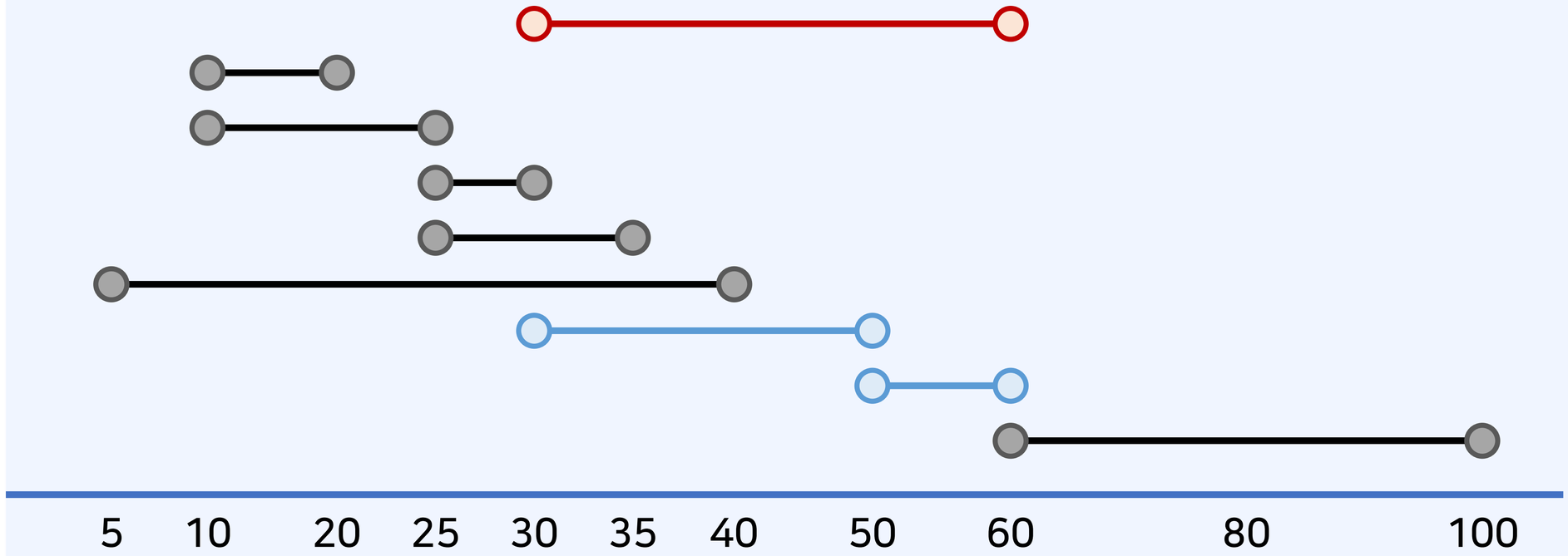
- 문제에서 주어진 예시를 확인해 보자. $(d = 30)$
- **여섯 번째 선분**을 우선순위 큐(힙)에 삽입하고, 두 개의 선분을 우선순위 큐에서 추출한다.



철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

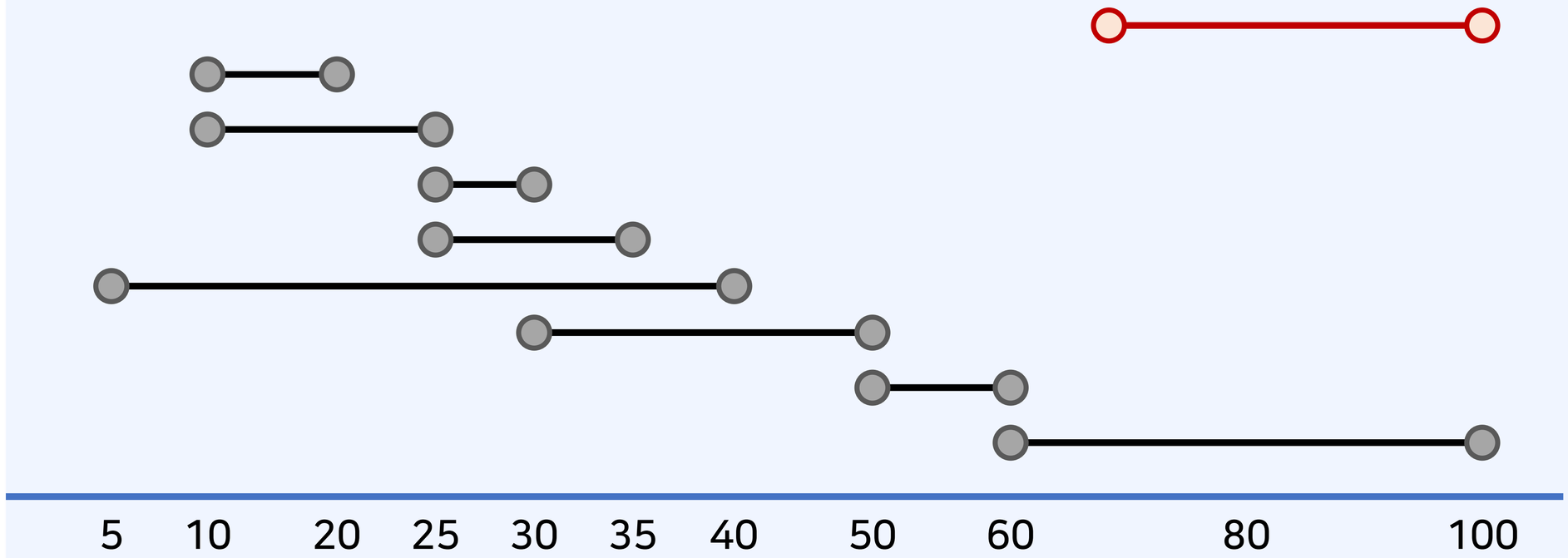
- 문제에서 주어진 예시를 확인해 보자. ($d = 30$)
- **일곱 번째 선분**을 우선순위 큐(힙)에 삽입하고, 두 개의 선분을 우선순위 큐에서 추출한다.



철로: https://www.acmicpc.net/problem/13334

문제 풀이 핵심 아이디어

- 문제에서 주어진 예시를 확인해 보자. $(d = 30)$
- **여덟 번째 선분**은 철로의 길이(d)보다 길기 때문에 무시하고, 두 개의 선분을 추출한다.



철로: https://www.acmicpc.net/problem/13334

Ch11. 탐욕 알고리즘 소스 코드
핵심 유형 문제풀이

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

n = int(input()) # 사람의 수
lines = []
# 각 사람의 (시작점, 끝점) 위치 입력
for i in range(n):
    start, end = map(int, input().split())
    # 항상 시작점이 더 작도록 저장
    if end < start:
        start, end = end, start
    lines.append((start, end))
# "끝" 위치를 기준으로 오름차순 정렬
lines.sort(key=lambda x: x[1])
d = int(input()) # 철로의 길이
heap = []
cur = 0 # 철로 선분의 끝점(천천히 오른쪽으로 이동)
result = 0
for line in lines: # 각 선분을 하나씩 확인
    start, end = line
    # 해당 선분이 d보다 길다면 무시
    if end - start > d:
        continue
    cur = end # 철로 선분을 오른쪽으로 이동시키기
    heapq.heappush(heap, start) # 시작점 넣기
    while len(heap) > 0:
        # 가장 왼쪽에 있는 시작점을 확인하며
        start = heapq.heappop(heap)
        # 철로를 벗어난 시작점 모두 꺼내기
        if cur - start > d:
            continue
        else: # 아니라면, 시작점 꺼내지 말기
            heapq.heappush(heap, start)
            break
    # 최댓값 계산
    result = max(result, len(heap))
print(result)
```