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

좌표 압축: https://www.acmicpc.net/problem/18870

혼자 힘으로 풀어보기

문제 제목: 좌표 압축

문제 난이도: ★★☆☆☆

문제 유형: 정렬, 좌표 압축

추천 풀이 시간: 30분

좌표 압축: https://www.acmicpc.net/problem/18870

문제 풀이 핵심 아이디어

- 좌표 압축을 수행하면, 그 값은 **자기보다 작은 원소의 개수**가 된다.

  예를 들어 입력이 $[2, 4, -10, 4, -9]$라면, 결과는 $[2, 3, 0, 3, 1]$이다.

  1. “2”는 자기보다 작은 원소가 **2개**입니다.
  2. “4”는 자기보다 작은 원소가 **3개**입니다.
  3. “-10”은 자기보다 작은 원소가 **0개**입니다.
  4. “4”는 자기보다 작은 원소가 **3개**입니다.
  5. “-9”는 자기보다 작은 원소가 **1개**입니다.

- 데이터의 개수(N)가 최대 1,000,000이므로, $O(NlogN)$의 복잡도로 문제를 해결해야 한다.

좌표 압축: https://www.acmicpc.net/problem/18870

문제 풀이 핵심 아이디어

1. 문제에서는 하나의 배열(array)이 입력된다.

   예를 들어 $arr = [2, 4, -10, 4, -9]$가 입력되었다고 가정하자.

2. 가장 먼저 집합(set)을 이용해 **중복 원소를 제거**하여 $unique = [2, 4, -10, -9]$가 된다.

3. 배열을 **오름차순 정렬**한 결과로 $unique = [-10, -9, 2, 4]$를 얻는다.

4. 정렬된 원소를 **각 인덱스(압축 수행 결과)에 매핑**한다.

   결과적으로 $mapping = \{"-10": 0, "-9": 1, "2": 2, "4": 3\}$가 된다.

5. 초기 원소를 하나씩 확인하여 압축 수행 결과를 확인한다.

$$print(mapping[arr[0]]) \ \#\ 2 \ \rightarrow \ 2$$

좌표 압축: https://www.acmicpc.net/problem/18870

소스 코드


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

n = int(input()) # 좌표의 개수 N
arr = list(map(int, input().split())) # 초기 원소

unique = list(set(arr)) # 중복 원소 제거
unique.sort() # 오름차순 정렬

mapping = {} # 각 원소를 압축 수행 결과에 매핑
for i, x in enumerate(unique):
    mapping[x] = i

# 최종 압축 수행 결과 출력
for x in arr:
    print(mapping[x], end=' ')
```

시리얼 번호: https://www.acmicpc.net/problem/1431

혼자 힘으로 풀어보기

문제 제목: 시리얼 번호

문제 난이도: ★☆☆☆☆

문제 유형: 정렬

추천 풀이 시간: 20분

시리얼 번호: https://www.acmicpc.net/problem/1431

문제 풀이 핵심 아이디어

- 파이썬의 정렬 라이브러리를 잘 활용할 수 있다면 해결할 수 있다.
- 파이썬에서는 $sort()$ 메서드에 들어가는 key 함수를 조절해 자신이 원하는 기준에 따라서 원소를 정렬할 수 있다.
- key 함수의 반환 값은 **튜플(tuple)**로 구성할 수 있는데, 값이 낮을수록 우선순위가 높아서 앞쪽으로 온다.
- 예를 들어 데이터가 (이름, 성적)일 때, 다음과 같이 성적 순으로 내림차순 정렬할 수 있다.

```
arr = [["홍길동", 83], ["이순신", 95], ["김철수", 100]]

# 두 번째 원소를 기준으로 내림차순 정렬
arr.sort(key=lambda x: -x[1])
print(arr)
```

시리얼 번호: https://www.acmicpc.net/problem/1431

문제 풀이 핵심 아이디어

- 본 문제를 풀기 위한 **정렬 기준**은 다음과 같다.

1. 길이가 짧을수록
2. 각 자릿수의 합이 작을수록
3. 사전순으로

시리얼 번호: https://www.acmicpc.net/problem/1431

소스 코드


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

# 자릿수의 합을 반환하는 함수
def digit_sum(str):
    result = 0 # 각 자릿수의 합
    for x in str: # 문자를 하나씩 확인하며
        if x.isdigit(): # 숫자일 때만 더하기
            result += int(x)
    return result

n = int(input()) # 기타의 개수 N
arr = [] # 각 기타의 시리얼 번호
for i in range(n):
    arr.append(input().strip())
arr.sort(key=lambda x: (len(x), digit_sum(x), x)) # 정렬 수행

# 정렬 수행 결과 출력
for x in arr:
    print(x)
```

보물: https://www.acmicpc.net/problem/1026

혼자 힘으로 풀어보기

문제 제목: 보물

문제 난이도: ★☆☆☆☆

문제 유형: 정렬

추천 풀이 시간: 30분

보물: https://www.acmicpc.net/problem/1026

문제 풀이 핵심 아이디어

- 배열 A와 B의 동일한 위치에 대하여 각 원소를 곱해야 한다.
- **핵심 아이디어**: 기본적으로 (큰 값)과 (작은 값)을 곱할수록, 곱셈 결과는 작아진다.
- 따라서 다음과 같은 방식으로 곱하면 된다.

1. 배열 B에서 1번째로 큰 값 × 배열 A에서 1번째로 작은 값
2. 배열 B에서 2번째로 큰 값 × 배열 A에서 2번째로 작은 값

   ...

3. 배열 B에서 N번째로 큰 값 × 배열 A에서 N번째로 작은 값

보물: https://www.acmicpc.net/problem/1026

문제 풀이 핵심 아이디어

- 문제에서는 배열 B를 재배열할 수 없다고 했다.
- 하지만 배열 A를 재배열할 수 있기 때문에, 사실상 배열 A와 B를 모두 재배열할 수 있을 때와 결과는 동일하다.

1. 가장 먼저 A는 오름차순 정렬, B는 내림차순 정렬한다.
2. 각 동일한 위치의 원소끼리 곱셈을 수행하고, 합계를 저장한다.

보물: https://www.acmicpc.net/problem/1026

소스 코드


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

n = int(input()) # 정수의 개수 N
# 배열 A와 B 입력
A = list(map(int, input().split()))
B = list(map(int, input().split()))

A.sort() # 오름차순 정렬
B.sort(reverse=True) # 내림차순 정렬

result = 0
for i in range(n):
    result += A[i] * B[i] # 곱셈 결과 더하기
print(result)
```