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

Byte Coin: https://www.acmicpc.net/problem/17521

혼자 힘으로 풀어보기

문제 제목: Byte Coin

문제 난이도: ★★☆☆☆

문제 유형: 그리디

추천 풀이 시간: 30분

Byte Coin: https://www.acmicpc.net/problem/17521

문제 풀이 핵심 아이디어

- 문제에서 주어진 예시는 다음과 같다.

날짜	1일	2일	3일	4일	5일	6일	7일	8일	9일	10일
가격	5	7	5	4	2	7	8	5	3	4

Byte Coin: https://www.acmicpc.net/problem/17521

문제 풀이 핵심 아이디어

- **핵심 아이디어**: 단순히 매 날짜를 기준으로 다음 날짜에 가격이 오른다면 사기

  본 문제에서는 **거래 수수료가 없다.**
- 따라서, 두 인접한 날짜의 차이만 고려하면 된다.

날짜	1일	2일	3일	4일	5일	6일	7일	8일	9일	10일
가격	5	7	5	4	2	7	8	5	3	4
팔지		O				O	O			O
살지	O				O	O			O	

Byte Coin: https://www.acmicpc.net/problem/17521

Ch11. 탐욕 알고리즘 소스 코드
핵심 유형 문제풀이

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

# 날짜와 초기 금액 입력받기
n, w = map(int, input().split(' '))

# 날짜별 가격을 입력받아 배열로 만들기
prices = []
for i in range(n):
    prices.append(int(input()))

# 각 날짜를 확인하면서
for i in range(n - 1):
  # 다음날에 가격이 오른다면, 오늘 사서 내일 팔기
  if prices[i] < prices[i + 1]:
      # 오늘 구매 가능한 개수
      cnt = w // prices[i]
      # (구매한 개수 X 시세 차익)만큼 더해주기
      w += cnt * (prices[i + 1] - prices[i])

print(w)
```

우리집엔 도서관이 있어: https://www.acmicpc.net/problem/2872

혼자 힘으로 풀어보기

문제 제목: 우리집엔 도서관이 있어

문제 난이도: ★★☆☆☆

문제 유형: 그리디

추천 풀이 시간: 30분

우리집엔 도서관이 있어: https://www.acmicpc.net/problem/2872

문제 풀이 핵심 아이디어

- 책을 한 권씩 꺼내서 위로 올리는 방식으로 1부터 N까지 나열하는 문제다.
- 문제를 해결하기 위해 여러 가지 예시를 고민해 볼 필요가 있다.

[예시 1]

- **초기 값**: [3, 2, 1]

1. 2를 꺼내서 위로 올리기 → [2, 3, 1]
2. 1을 꺼내서 위로 올리기 → [1, 2, 3]

우리집엔 도서관이 있어: https://www.acmicpc.net/problem/2872

문제 풀이 핵심 아이디어

- 책을 한 권씩 꺼내서 위로 올리는 방식으로 1부터 N까지 나열하는 문제다.
- 문제를 해결하기 위해 여러 가지 예시를 고민해 볼 필요가 있다.

[예시 2]

- **초기 값**: [5, 6, 3, 4, 1, 2]

1. 4를 꺼내서 위로 올리기 → [4, 5, 6, 3, 1, 2]
2. 3을 꺼내서 위로 올리기 → [3, 4, 5, 6, 1, 2]
3. 2를 꺼내서 위로 올리기 → [2, 3, 4, 5, 6, 1]
4. 1을 꺼내서 위로 올리기 → [1, 2, 3, 4, 5, 6]

우리집엔 도서관이 있어: https://www.acmicpc.net/problem/2872

문제 풀이 핵심 아이디어

- [핵심 아이디어] 가장 큰 값(번호가 높은 책)은 위치를 옮길 필요가 없다.
- 그렇다면, 위치를 옮길 필요가 없는 원소들은 정확히 어떻게 결정할 수 있을까?
- **"뒤에서부터 볼 때" 가장 큰 값부터 내림차순으로 구성된 원소들**은 위치를 옮길 필요가 없다.

우리집엔 도서관이 있어: https://www.acmicpc.net/problem/2872

문제 풀이 핵심 아이디어

- **“뒤에서부터 볼 때” 가장 큰 값부터 내림차순으로 구성된 원소**의 개수 구하기

  이때, 꼭 연속된 원소일 필요는 없습니다.

[예시]

- **초기 값**: [4, 1, 2, 5, 6, 3]
- 6 → 5 → 4로 내림차순을 형성한다.
- 따라서 [4, 1, 2, 5, 6, 3]이므로, 바꿀 원소는 3개다.

[예시]

- **초기 값**: [4, 5, 1, 6, 7, 2, 3, 8]
- 8 → 7 → 6 → 5 → 4로 내림차순을 형성한다.
- 따라서 [4, 5, 1, 6, 7, 2, 3, 8]이므로, 바꿀 원소는 3개다.

우리집엔 도서관이 있어: https://www.acmicpc.net/problem/2872


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

n = int(input()) # 책의 개수 N 입력
books = [] # 각 책의 번호 배열

# 모든 책 정보 입력받기
for i in range(1, n + 1):
    books.append(int(input()))
```

```python
# 가장 큰 값과 그 인덱스를 찾기
max_value = 0
max_index = -1
for i in range(n):
    if max_value < books[i]:
        max_value = books[i]
        max_index = i

# 뒤에서부터 출발해서 내림차순 배열의 크기를 계산
length = 1
target = max_value - 1; # 다음 원소(target)
for i in range(max_index - 1, -1, -1):
    # 다음 원소를 찾았다면, 그 다음 원소(target - 1) 찾기
    if target == books[i]:
        target -= 1
        length += 1

print(n - length)
```

보석 도둑: https://www.acmicpc.net/problem/1202

혼자 힘으로 풀어보기

문제 제목: 보석 도둑

문제 난이도: ★★★☆☆

문제 유형: 그리디

추천 풀이 시간: 50분

보석 도둑: https://www.acmicpc.net/problem/1202

문제 풀이 핵심 아이디어

- 보석은 **(무게, 가격)**으로 표현되며, 가방은 **(허용량)**으로 표현된다.
- [핵심] 가방의 무게가 높을수록 더 좋은 가방이다.

1. 따라서 반복문을 이용해 허용량이 낮은 가방부터 확인한다.
2. 남은 보석들 중에서 해당 가방에 **들어갈 수 있는 보석 중 가장 가격이 높은 보석**을 넣는다.

- 이러한 방식으로 항상 최적의 해를 보장할 수 있다.

보석 도둑: https://www.acmicpc.net/problem/1202

문제 풀이 핵심 아이디어

- 이러한 알고리즘을 단순히 구현하려면 어떻게 하면 될까?
- 만약 2중 반복문을 이용해 각 가방마다 모든 보석을 확인한다면, 시간 초과가 발생한다.

  $O(NM)$의 **수행 시간**이 소요되기 때문이다.
- **보석의 개수** N은 최대 300,000, **가방의 개수** K는 최대 300,000이다.

보석 도둑: https://www.acmicpc.net/problem/1202

문제 풀이 핵심 아이디어

- 따라서, “**가격이 가장 높은 보석**”에 빨리 접근하기 위해 **우선순위 큐**를 사용한다.

1. 먼저 각 보석 **(무게, 가격)**들과 가방 **(허용량)**들을 오름차순 정렬한다.
2. 가방을 하나씩 차례대로 확인한다.

   해당 가방에 들어갈 수 있는 모든 보석들을 “가격”만 우선순위 큐에 넣는다.

   우선순위 큐에서 top(가장 가격이 높은 보석)을 꺼내어 해당 보석을 가방에 넣는다.

- 결과적으로 모든 보석이 우선순위 큐에 담겼다가 꺼내지는 것으로 이해할 수 있다.

  따라서, 정렬에 의해 시간 복잡도는 $O(NlogN + KlogK)$다.

보석 도둑: https://www.acmicpc.net/problem/1202

문제 풀이 핵심 아이디어

- 예를 들어 보석이 (1, 65), (5, 23), (2, 99), (13, 10) 가방이 (10), (2), (15)이다.
- **가방**을 오름차순 정렬하면 (2), (10), (15)가 된다.
- **보석**을 오름차순 정렬하면 (1, 65), (2, 99), (5, 23), (13, 10)이 된다.

1. **첫 번째 가방**: 현재 가방의 허용량이 2이므로, (1, 65)와 (2, 99)를 넣는다.

   우선순위 큐: [99, 65] → 이후에 99를 꺼낸다.

2. **두 번째 가방**: 현재 가방의 허용량이 10이므로, (5, 23)을 넣는다.

   우선순위 큐: [65, 23] → 이후에 65를 꺼낸다.

3. **세 번째 가방**: 현재 가방의 허용량이 15이므로, (13, 10)을 넣는다.

   우선순위 큐: [23, 10] → 이후에 23을 꺼낸다.

- 따라서 **정답은 187**이다.

보석 도둑: https://www.acmicpc.net/problem/1202

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, k = map(int, input().split())

# 각 보석 입력받기
stones = []
for i in range(n):
    weight, price = map(int, input().split())
    stones.append((weight, price))

# 각 가방 정보 입력받기
bags = []
for i in range(k):
    bag = int(input())
    bags.append(bag)

# 보석 목록과 가방 목록을 정렬하기
stones.sort()
bags.sort()

heap = [] # 힙 생성
cur = 0
result = 0
for bag in bags: # 각 가방을 하나씩 확인하며
    while cur < n:
        # 가방에 들어갈 수 있는 보석들을 힙에 삽입
        weight, price = stones[cur]
        if bag >= weight:
            # 최대 힙 기능을 위해 삽입/삭제할 때 부호 변경
            heapq.heappush(heap, -price)
            cur += 1
        else: # 최대한 힙에 넣었다면 탈출
            break
    # 해당 가방에 넣을 보석이 있다면
    if len(heap) > 0:
        price = -heapq.heappop(heap)
        result += price

print(result)
```