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

크게 만들기: https://www.acmicpc.net/problem/2812

혼자 힘으로 풀어보기

문제 제목: 크게 만들기

문제 난이도: ★★★☆☆

문제 유형: 그리디, 스택

추천 풀이 시간: 50분

크게 만들기: https://www.acmicpc.net/problem/2812

문제 풀이 핵심 아이디어

- 하나의 수가 주어졌을 때, K개의 숫자를 지워 얻을 수 있는 가장 큰 수를 계산한다.
- **최대한 큰 숫자가 앞쪽(왼쪽)에** 남아 있어야 한다.
- 본 문제는 스택 자료구조를 이용해 효율적으로 해결할 수 있다.

크게 만들기: https://www.acmicpc.net/problem/2812

문제 풀이 핵심 아이디어

[문제 해결 아이디어]

1. 원소를 차례대로 하나씩 확인한다.

   단, 현재 원소가 스택의 $top()$보다 크다면, 작을 때까지(가능한 만큼) 스택에서 $pop()$한다.

2. 현재 원소를 스택에 삽입한다.

크게 만들기: https://www.acmicpc.net/problem/2812

문제 풀이 핵심 아이디어

- 데이터의 개수(N): 9, 삭제 횟수(K): 5
- **초기 값**: $[1, 9, 6, 5, 4, 8, 8, 9, 8]$

1. 원소 1을 스택에 삽입한다.

   스택: [1], 현재까지 삭제 횟수: 0

2. 원소 9가 스택의 $top()$보다 크므로, $pop()$ 이후에 삽입한다.

   스택: [9], 현재까지 삭제 횟수: 1

3. 원소 6을 스택에 삽입한다.

   스택: [9, 6], 현재까지 삭제 횟수: 1

4. 원소 5를 스택에 삽입한다.

   스택: [9, 6, 5], 현재까지 삭제 횟수: 1

문제 풀이 핵심 아이디어

- 데이터의 개수(N): 9, 삭제 횟수(K): 5
- **초기 값**: $[1, 9, 6, 5, 4, 8, 8, 9, 8]$

5. 원소 4를 스택에 삽입한다.

   스택: [9, 6, 5, 4], 현재까지 삭제 횟수: 1

6. 원소 8이 스택의 $top()$보다 크므로, 가능한 만큼 $pop()$ 이후에 삽입한다.

   스택: [9, 8], 현재까지 삭제 횟수: 4

7. 원소 8을 스택에 삽입한다.

   스택: [9, 8, 8], 현재까지 삭제 횟수: 4

8. 원소 9가 스택의 $top()$보다 크므로, 가능한 만큼 $pop()$ 이후에 삽입한다.

   스택: [9, 8, 9], 현재까지 삭제 횟수: 5

문제 풀이 핵심 아이디어

- 데이터의 개수(N): 9, 삭제 횟수(K): 5
- **초기 값**: $[1, 9, 6, 5, 4, 8, 8, 9, 8]$

9. 원소 8을 스택에 삽입한다.

   스택: [9, 8, 9, 8], 현재까지 삭제 횟수: 5

크게 만들기: https://www.acmicpc.net/problem/2812


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

# 자릿수 N과 삭제 횟수 K를 입력
n, k = map(int, input().split())
data = input().strip() # 초기 입력
```

```python
deleted = 0 # 현재까지 삭제 횟수
stack = [] # 스택
for x in data:
    # 스택이 비어있지 않고 현재 원소가 스택의 top()보다 크면
    while len(stack) > 0 and stack[-1] < x:
        # 더 이상 삭제할 수 없다면 탈출
        if deleted == k:
            break
        # 스택에서 pop() 수행
        else:
            stack.pop()
            deleted += 1
    stack.append(x)

# 삭제 횟수가 남아있다면 단순히 반복적으로 pop()
for i in range(k - deleted):
    stack.pop()

# 결과 출력
print(''.join(stack))
```

이장님 초대: https://www.acmicpc.net/problem/9237

혼자 힘으로 풀어보기

문제 제목: 이장님 초대

문제 난이도: ★☆☆☆☆

문제 유형: 그리디, 정렬

추천 풀이 시간: 20분

이장님 초대: https://www.acmicpc.net/problem/9237

문제 풀이 핵심 아이디어

[문제 풀이 아이디어]

- 묘목 하나를 심기 위해 1일의 시간이 소요된다.
- 따라서 **"가장 오래 걸리는" 묘목부터 가장 먼저** 심어야, 최종적으로 모든 나무가 완전히 다 자라는 시간이 감소한다.

이장님 초대: https://www.acmicpc.net/problem/9237

문제 풀이 핵심 아이디어

- 예를 들어 주어진 **각 묘목의 정보**가 [2, 3, 4, 3]이라고 해보자.

  내림차순 정렬한 결과는 [4, 3, 3, 2]이다.
- 이때 각 나무가 모두 자라는 날짜는 다음과 같다.

  [1 + 4, 2 + 3, 3 + 3, 4 + 2]
- 따라서 **최댓값은 6**이므로, 그 다음날인 $7 = 6 + 1$이 정답이다.

이장님 초대: https://www.acmicpc.net/problem/9237

Ch11. 탐욕 알고리즘 소스 코드
핵심 유형 문제풀이

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

n = int(input()) # 묘목의 수 N
arr = list(map(int, input().split())) # 각 나무가 자라는데 걸리는 시간

arr.sort(reverse=True) # 내림차순 정렬

max_value = 0 # 모든 나무가 완전히 자라는 날짜
for i in range(n):
    max_value = max(max_value, i + 1 + arr[i])
print(max_value + 1) # 그 다음 날짜를 출력
```

주식: https://www.acmicpc.net/problem/11501

혼자 힘으로 풀어보기

문제 제목: 주식

문제 난이도: ★★☆☆☆

문제 유형: 그리디

추천 풀이 시간: 30분

주식: https://www.acmicpc.net/problem/11501

문제 풀이 핵심 아이디어

- 각 날짜별로, 뒤에 오는 날짜의 주가 중에서 최댓값을 알아내면 된다.
- 예를 들어 다음과 같이 주가가 구성된다고 가정해 보자.

  입력: [7, 8, 10, 9, 12, 7, 4, 5]
- 이때 각 날짜를 기준으로, "이후 날짜"의 최댓값을 구하면 다음과 같다.

  목표: [12, 12, 12, 12, 12, 7, 5, 5]
- 결과적으로, 이득을 볼 수 있는 경우에만 주식을 사고 팔면 된다.

  이득: [5, 4, 2, 3, 0, 0, 1, 0]
- 이 경우 정답은 15다.

주식: https://www.acmicpc.net/problem/11501

문제 풀이 핵심 아이디어

- **"이후 날짜"**의 최댓값을 구하는 방법은 간단하다.
- 뒤에서부터 하나씩 보면서 더 큰 값을 만날 때까지 최댓값을 기록한다.
- **입력**: [7, 8, 10, 9, 12, 7, 4, 5]

1. 5를 확인하여, 더 큰 값을 만날 때까지 기록한다.

   목표: [0, 0, 0, 0, 0, 0, 5, 5]

2. 7을 확인하여, 더 큰 값을 만날 때까지 기록한다.

   목표: [0, 0, 0, 0, 0, 7, 5, 5]

3. 12를 확인하여, 더 큰 값을 만날 때까지 기록한다.

   목표: [12, 12, 12, 12, 12, 7, 5, 5]

주식: https://www.acmicpc.net/problem/11501

Ch11. 탐욕 알고리즘 소스 코드
핵심 유형 문제풀이

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

for test_case in range(int(input())):
    n = int(input()) # 전체 날짜
    arr = list(map(int, input().split())) # 날짜별 주가

    max_value = arr[n - 1] # 가장 뒤쪽 원소
    target = [0] * n # 팔 가격
    # 뒤쪽에서부터 하나씩 확인하며
    for i in range(n - 1, -1, -1):
        # 각 원소의 이후 원소 중에서 가장 큰 원소 기록
        max_value = max(max_value, arr[i])
        target[i] = max_value

    result = 0
    for i in range(n):
        # 이익이 생기는 경우에만 합계에 더해주기
        result += max(0, target[i] - arr[i])
    print(result)
```