Chapter 01.
기본 자료구조
핵심 유형 문제풀이

핵심 유형 문제풀이 | 다양한 문제를 접하며 코딩 테스트에 익숙해지기

강사 나동빈

Chapter 01. 기본 자료구조

핵심 유형 문제풀이

오큰수: https://www.acmicpc.net/problem/17298

혼자 힘으로 풀어보기

문제 제목: 오큰수

문제 난이도: 중상

문제 유형: 자료구조, 스택

추천 풀이 시간: 50분

오큰수: https://www.acmicpc.net/problem/17298

문제 풀이 핵심 아이디어

- 본 문제는 스택(stack) 자료구조를 이용해 해결할 수 있는 문제입니다.
- 다양한 예시를 생각해 보며 시뮬레이션 해보면 **아이디어를 떠올리기** 수월합니다.

오큰수: https://www.acmicpc.net/problem/17298

문제 풀이 핵심 아이디어

[알고리즘 요약]

- 수 x를 하나씩 확인하며, 내림차순 혹은 같은 값의 x가 나오는 경우 스택에 삽입한다.
- 더 큰 수(오름차순) x가 나왔을 때, x는 앞선 데이터에 대해서 오큰수가 될 수 있다.
  - 따라서 스택의 $pop()$을 이용해 **역방향으로 확인**하면서 오큰수를 기록한다.
  - 만약 역방향의 수가 x보다 크거나 같다면 멈추고, 해당 수부터 다시 내림차순/오름차순 여부를 확인한다.
- 동작 과정상 오른쪽으로 이동한 뒤에 다시 왼쪽으로(역방향으로) 이동하는 것을 반복합니다.
- 결과적으로 시간 복잡도 $O(N)$으로 해결할 수 있습니다.

오큰수: https://www.acmicpc.net/problem/17298

문제 풀이 핵심 아이디어

- **아이디어**: 오큰수를 아직 찾지 못한 **처리 중인** 원소를 담기 위해 스택을 사용
- **수열 예시**: [3, 2, 1, 10, 9, 5, 4, 8, 15, 11, 13]

① 3을 스택에 넣는다.

스택 = [3]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

② 이어서 2을 스택에 넣는다.

스택 = [3, 2]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

③ 이어서 1을 스택에 넣는다.

스택 = [3, 2, 1]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

오큰수: https://www.acmicpc.net/problem/17298

문제 풀이 핵심 아이디어

- **아이디어**: 오큰수를 아직 찾지 못한 **처리 중인** 원소를 담기 위해 스택을 사용
- **수열 예시**: [3, 2, 1, 10, 9, 5, 4, 8, 15, 11, 13]

④ top()보다 큰 수 10을 만났으므로, 역방향으로 하나씩 꺼내고 10을 스택에 넣는다.

스택 = [10]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	10	10	10	-1	-1	-1	-1	-1	-1	-1	-1

⑤ 이어서 9를 스택에 넣는다.

스택 = [10, 9]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	10	10	10	-1	-1	-1	-1	-1	-1	-1	-1

⑥ 이어서 5를 스택에 넣는다.

스택 = [10, 9, 5]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	10	10	10	-1	-1	-1	-1	-1	-1	-1	-1

오큰수: https://www.acmicpc.net/problem/17298

문제 풀이 핵심 아이디어

- **아이디어**: 오큰수를 아직 찾지 못한 **처리 중인** 원소를 담기 위해 스택을 사용
- **수열 예시**: [3, 2, 1, 10, 9, 5, 4, 8, 15, 11, 13]

⑦ 이어서 4를 스택에 넣는다.

스택 = [10, 9, 5, 4]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	10	10	10	-1	-1	-1	-1	-1	-1	-1	-1

⑧ top()보다 큰 수 8을 만났으므로, 역방향으로 하나씩 꺼내고 8을 스택에 넣는다.

스택 = [10, 9, 8]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	10	10	10	-1	-1	8	8	-1	-1	-1	-1

⑨ top()보다 큰 수 15를 만났으므로, 역방향으로 하나씩 꺼내고 15를 스택에 넣는다.

스택 = [15]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	10	10	10	15	15	8	8	15	-1	-1	-1

오큰수: https://www.acmicpc.net/problem/17298

문제 풀이 핵심 아이디어

- **아이디어**: 오큰수를 아직 찾지 못한 **처리 중인** 원소를 담기 위해 스택을 사용
- **수열 예시**: [3, 2, 1, 10, 9, 5, 4, 8, 15, 11, 13]

⑩ 이어서 11을 스택에 넣는다.

스택 = [15, 11]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	10	10	10	15	15	8	8	15	-1	-1	-1

⑪ top()보다 큰 수 13를 만났으므로, 역방향으로 하나씩 꺼내고 13을 스택에 넣는다.

스택 = [15, 13]

수열	3	2	1	10	9	5	4	8	15	11	13
오큰수	10	10	10	15	15	8	8	15	-1	13	-1

오큰수: https://www.acmicpc.net/problem/17298

소스 코드


```
n = int(input()) # 수의 개수
arr = list(map(int, input().split())) # 수열 데이터
stack = [] # 스택(stack) 초기화
NGE = [-1] * n # 오큰수 배열

for i in range(n):
    x = arr[i] # 하나씩 수 확인
    if len(stack) == 0 or stack[-1][0] >= x: # 내림차순 형태라면(작거나 같은 원소를 만났다면)
        stack.append((x, i)) # (수, 인덱스) 형태로 삽입
    else: # 오름차순 형태라면(큰 수를 만났다면)
        while len(stack) > 0: # 역방향으로 하나씩 꺼내기
            previous, index = stack.pop()
            if previous >= x: # 크거나 같은 이전 원소를 만났다면 다시 삽입
                stack.append((previous, index))
                break
            else:
                NGE[index] = x # 오큰수 기록
        stack.append((x, i)) # (수, 인덱스) 형태로 삽입

for x in NGE: # 오큰수를 하나씩 출력
    print(x, end=' ')
```

회전하는 큐: https://www.acmicpc.net/problem/1021

혼자 힘으로 풀어보기

문제 제목: 회전하는 큐

문제 난이도: 하(Easy)

문제 유형: 자료구조, 덱

추천 풀이 시간: 30분

회전하는 큐: https://www.acmicpc.net/problem/1021

문제 풀이 핵심 아이디어

- 양방향 순환 큐의 기능은 덱(deque)을 이용해 대체할 수 있습니다.
  1. 회전 연산 (왼쪽으로 돌리기): $popleft()$ - $append()$
  2. 회전 연산 (오른쪽으로 돌리기): $pop()$ - $appendleft()$
  3. 추출 연산 (앞에서 꺼내기): $popleft()$

회전하는 큐: https://www.acmicpc.net/problem/1021

문제 풀이 핵심 아이디어

[알고리즘 요약]

- 먼저 번호(index)가 1부터 시작하도록 N개의 데이터를 덱(deque)에 삽입한다.
- 이후에 M개의 뽑아야 하는 인덱스들이 주어질 때는 다음과 같이 한다.
  - 매번 왼쪽과 오른쪽 중에서 어디가 더 가까운지 먼저 확인하여, 더 가까운 쪽으로 연속적으로 "왼쪽으로 돌리기"나 "오른쪽으로 돌리기"를 사용한 뒤에 원소를 뽑는다.
  - 만약 [1, 2, 3, 4, 5, 6]처럼 개수가 **짝수**라면, [o, o, o, o, x, x] 위치는 왼쪽으로 돌리기
  - 만약 [1, 2, 3, 4, 5]처럼 개수가 **홀수**라면, [o, o, o, x, x] 위치는 왼쪽으로 돌리기
- 이러한 간단한 알고리즘으로 항상 최적의 해를 보장할 수 있다.
  꺼내야 하는 원소는 순서대로 이미 정해져 있기 때문이다.

회전하는 큐: https://www.acmicpc.net/problem/1021

문제 풀이 핵심 아이디어

- 아이디어: 덱(deque)을 이용해 원소의 회전 기능을 구현할 수 있습니다.
- **덱의 초기 상태**: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] / **꺼낼 원소**: [2, 9, 5]

① **회전 연산의 횟수** = 1

덱의 상태: [3, 4, 5, 6, 7, 8, 9, 10, 1] / **꺼낼 원소**: [9, 5]

② **회전 연산의 횟수** = 4

덱의 상태: [10, 1, 3, 4, 5, 6, 7, 8] / **꺼낼 원소**: [5]

③ **회전 연산의 횟수** = 8

덱의 상태: [6, 7, 8, 10, 1, 3, 4] / **꺼낼 원소**: []

회전하는 큐: https://www.acmicpc.net/problem/1021

소스 코드


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

n, m = map(int, input().split()) # 원소의 개수, 뽑아내는 횟수
d = deque([i for i in range(1, n + 1)]) # 1부터 N까지의 원소를 삽입
targets = list(map(int, input().split())) # 뽑아낼 원소 목록
cnt = 0 # 회전 연산 수행 횟수

for target in targets: # 뽑아낼 원소를 하나씩 확인하며
    index = d.index(target) # 덱에서 해당 원소의 위치를 찾기
    if index <= len(d) // 2: # 왼쪽으로 돌리는 게 더 빠른 경우
        for i in range(index): # 회전 연산 반복 수행
            x = d.popleft()
            d.append(x)
            cnt += 1
    else: # 오른쪽으로 돌리는 게 더 빠른 경우
        for i in range(len(d) - index): # 회전 연산 반복 수행
            x = d.pop()
            d.appendleft(x)
            cnt += 1
    d.popleft() # 원소 꺼내기

print(cnt) # 결과 출력
```

혼자 힘으로 풀어보기

문제 제목: 요세푸스 문제 0

문제 난이도: 하(Easy)

문제 유형: 자료구조, 덱

추천 풀이 시간: 30분

요세푸스 문제 0: https://www.acmicpc.net/problem/11866

문제 풀이 핵심 아이디어

- 회전 연산을 포함하는 큐의 기능은 덱(deque)을 이용해 대체할 수 있습니다.
  1. 회전 연산 (왼쪽으로 돌리기): $popleft()$ - $append()$
  2. 추출 연산 (앞에서 꺼내기): $popleft()$

문제 풀이 핵심 아이디어

[알고리즘 요약]

- 먼저 번호(index)가 1부터 시작하도록 N개의 데이터를 덱(deque)에 삽입한다.
- 덱(deque)이 빌 때까지 계속해서 원소를 추출한다.
  - 매번 K-1개만큼 "왼쪽으로 돌리기"를 수행하고, 1개를 그냥 추출한다.
- 데이터의 개수가 최대 1,000개이므로, 이러한 시뮬레이션으로 간단히 해결할 수 있다.

요세푸스 문제 0: https://www.acmicpc.net/problem/11866

문제 풀이 핵심 아이디어

- 아이디어: 덱(deque)을 이용해 원소의 회전 기능을 구현할 수 있습니다.
- N=7, K=3, 초기 원소 구성: [1, 2, 3, 4, 5, 6, 7]

① 덱 내부: [4, 5, 6, 7, 1, 2]

결과: [3]

② 덱 내부: [7, 1, 2, 4, 5]

결과: [3, 6]

③ 덱 내부: [4, 5, 7, 1]

결과: [3, 6, 2]

문제 풀이 핵심 아이디어

- 아이디어: 덱(deque)을 이용해 원소의 회전 기능을 구현할 수 있습니다.
- N=7, K=3, 초기 원소 구성: [1, 2, 3, 4, 5, 6, 7]

④ 덱 내부: [1, 4, 5]

결과: [3, 6, 2, 7]

⑤ 덱 내부: [1, 4]

결과: [3, 6, 2, 7, 5]

⑥ 덱 내부: [4]

결과: [3, 6, 2, 7, 5, 1]

문제 풀이 핵심 아이디어

- 아이디어: 덱(deque)을 이용해 원소의 회전 기능을 구현할 수 있습니다.
- N=7, K=3, 초기 원소 구성: [1, 2, 3, 4, 5, 6, 7]

⑦ 덱 내부: []

결과: [3, 6, 2, 7, 5, 1, 4]

요세푸스 문제 0: https://www.acmicpc.net/problem/11866

소스 코드


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

# 원소의 개수 N과 제거할 간격 K 입력
n, k = map(int, input().split())
# 1부터 N까지의 원소를 삽입
d = deque([i for i in range(1, n + 1)])
result = [] # 결과 배열

# N개의 원소를 모두 꺼내기
for i in range(n):
    # K - 1번 "왼쪽으로 돌리기" 수행
    for i in range(k - 1):
        x = d.popleft()
        d.append(x)
    x = d.popleft() # 원소 꺼내기
    result.append(x)

# 정답 출력
print('<', end='')
for i in range(len(result)):
    if i < len(result) - 1:
        print(result[i], end=', ')
    else: # 마지막 원소
        print(result[i], end='')
print('>')
```