이코테 3장 <구현>

22기 분석 원종빈

목차

1. 구현 문제란?

- 생각하는/주어진 알고리즘을 코드로 잘 '구현'할 수 있는가?

- 교재에서 다루는 구현 문제
  - 완전 탐색 (brute force)
  - 시뮬레이션 (simulation)

2. 구현 문제를 어떻게 준비해야할까?

1. 기본적으로는 알고리즘을 코드로 깔끔하게 구현하는 연습

    - 소위 pythonic이라고 불리는 코드들

```
## swapping
a, b = b, a
```

출처:https://stackoverflow.com/questions/14836228/is-there-a-standardized-method-to-swap-two-variables-in-python

```
## flatten list of lists
sum([[1,2], [3,4,]], [])

## returns [1, 2, 3, 4]
```

2. 구현 문제를 어떻게 준비해야할까?

2. built-in 라이브러리를 활용하는 연습

- collections (OrderedDict, defaultdict, Counter, deque)
- itertools (permutations, product)
- math (floor, ceil, log)

Greedy 실전 4) 1이 될 때 까지

```
from math import log

def exhaustive_one(n, k):
    count = 0
    count += (n % k)
    count += int(log(n, k))

    return count

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

exhaustive_one(n, k)
```

3. 구현 문제에서 고려할 점

- 직접 동작을 특정하는 경우가 많다

```
# L, R, U, D에 따른 이동 방향
dx = [0, 0, -1, 1]
dy = [-1, 1, 0, 0]
move_types = ['L', 'R', 'U', 'D']
```

```
# 북, 동, 남, 서 방향 정의
dx = [-1, 0, 1, 0]
```

```
# 나이트가 이동할 수 있는 8가지 방향 정의
steps = [(-2, -1), (-1, -2), (1, -2), (2, -1), (2, 1), (1, 2), (-1, 2), (-2, 1)]
```

3. 구현 문제에서 고려할 점

- 시간복잡도를 고려할 필요가 있다!

`len(a)`	O(1)	전체 요소의 개수를 리턴한다.
`a[i]`	O(1)	인덱스 i의 요소를 가져온다.
`a[i:j]`	O(k)	인덱스 i부터 j-1까지 슬라이스의 길이만큼인 k개의 요소를 가져온다. 이 경우 객체 k개에 대한 조회가 필요하므로 O(k)이다.
`elem in a`	O(n)	elem 요소가 존재하는지 확인한다. 처음부터 순차 탐색하므로 n만큼 시간이 소요된다.
`a.count(elem)`	O(n)	elem 요소의 개수를 리턴한다.
`a.index(elem)`	O(n)	elem 요소의 인덱스를 리턴한다.
`a.append(elem)`	O(1)	리스트 마지막에 elem 요소를 추가한다.

3. 구현 문제에서 고려할 점

- 시간복잡도를 고려할 필요가 있다!

`a.pop()`	O(1)	리스트 마지막 요소를 추출한다. 스택의 연산이다.
`a.pop(0)`	O(n)	리스트 첫번째 요소를 추출한다. 큐의 연산이다. 이 경우 전체 복사가 필요하므로 O(n)이다. 큐의 연산을 주로 사용한다면 리스트보다는 O(1)에 가능한 데크(deque)를 권장한다.
`del a[i]`	O(n)	i에 따라 다르다. 최악의 경우 O(n)이다.
`a.sort()`	O(NlogN)	정렬한다. 팀소트(Timsort)를 사용하며, 최선의 경우 O(n)에도 실행될 수 있다.
`min(a), max(a)`	O(n)	최솟값/최댓값을 계산하기 위해서는 전체를 선형 탐색해야 한다.
`a.reverse()`	O(n)	뒤집는다. 리스트는 입력 순서가 유지되므로 뒤집게 되면 입력 순서가 반대로 된다.

4. 구현문제 예시 - 완전 탐색(brute force)

- 예제 4-2 <시각> 문제

```
N = int(input())

count = 0
for i in range(N + 1):
    for j in range(60):
        for k in range(60):
            if '3' in str(i) + str(j) + str(k):
                count += 1

print(count)
```

4. 구현문제 예시 - 시뮬레이션

```
world_n, world_m = map(int, input().split())
n, m, d = map(int, input().split())

world = [list(map(int, input().split())) for _ in range(world_n)]

moves = [(-1, 0), (0, 1), (1, 0), (0, -1)]

world[n][m] = 2 ## 가본 땅은 2로 값 설정
count = 1   ## 맨 처음 도착한 땅이 있으니 1
turn_time = 0
while True:
    d = (d + 1) % 4
    dn, dm = n + moves[d][0], m + moves[d][1]
    if world[dn][dm] == 0:
        n, m = dn, dm
        count += 1
        turn_time = 0
        world[n][m] = 2

    elif world[dn][dm] != 0:
        turn_time += 1

    if turn_time == 4:
        dn, dm = n - moves[d][0], m - moves[d][1]
        if world[dn][dm] == 2:
            n, m = dn, dm

        else:
            break

        turn_time = 0

print(count)
```

다음주 과제

- 완전탐색: 5568-카드놓기 (https://www.acmicpc.net/problem/5568)
- 시뮬: 1018-체스판 다시칠하기 (https://www.acmicpc.net/problem/1018)

+ leetcode 1. Two Sum(http://leetcode.com/problems/two-sum/)