Algorithm/알고리즘

[BOJ](Python) 백준 1620번: 나는야 포켓몬 마스터 이다솜

그리버 | 2023. 11. 7. 17:28 | 수정 | 삭제

문제

https://www.acmicpc.net/problem/1620

주제	이진 탐색
시간 / 메모리 제한	2초 / 256MB
정답 비율	34.067%

입력

- 첫째 줄에는 도감에 수록된 포켓몬의 개수 N과 내가 맞춰야 하는 문제의 개수 M이 주어진다.
  - 1 <= N, M <= 100,000
- 둘째 줄부터 N개의 줄은 포켓몬의 이름이 입력된다.
  - 포켓몬의 번호가 1번인 포켓몬부터 N번에 해당하는 포켓몬까지 한 줄에 하나씩 입력으로 들어온다.
  - 포켓몬의 이름은 모두 영어이며 첫 글자는 대문자, 이후는 소문자로 이루어진다.
    - 일부 포켓몬은 마지막 문자만 대문자일 수도 있다.
    - 이름의 최대 길이는 20, 최소 길이는 2이다.
- 그 다음 줄부터 총 M개의 줄은 맞춰야 하는 문제가 입력된다.
  - 문제가 알파벳으로 들어오면 포켓몬 번호를 맞춰야 한다.
  - 문제가 숫자로 들어오면 포켓몬 번호에 해당하는 이름을 맞춰야 한다.

출력

- 첫째 줄부터 차례대로 M개의 줄에 각각의 문제에 대한 답을 출력한다.

- 예제 입력 1

```bash
26 5
Bulbasaur
Ivysaur
Venusaur
Charmander
Charmeleon
Charizard
Squirtle
Wartortle
Blastoise
Caterpie
Metapod
Butterfree
Weedle
Kakuna
Beedrill
Pidgey
Pidgeotto
Pidgeot
Rattata
Raticate
Spearow
Fearow
Ekans
Arbok
Pikachu
Raichu
25
Raichu
3
Pidgey
Kakuna
```

- 예제 출력 1

```bash
Pikachu
26
Venusaur
16
14
```

풀이

단순하게 바로 생각나는 대로 코드를 작성해보았다.

우선 N, M을 입력받고 N번 반복하며 포켓몬 이름이 담겨있는 리스트를 완성한다.

그리고 숫자가 들어오면 해당 숫자를 인덱스로 하여 이름을 출력하고

알파벳이 들어오면 리스트의 index 메서드를 이용하여 해당 포켓몬의 번호를 찾아 출력하였다.

아래와 같이 제출하였는데 시간 초과가 발생하였다.

```python
import sys

```

```python
N, M = list(map(int, sys.stdin.readline().strip().split()))

pokemon = [sys.stdin.readline().strip() for _ in range(N)]

for i in range(M):
    target = sys.stdin.readline().strip()

    if target.isdigit():
        print(pokemon[int(target) - 1])
    else:
        print(pokemon.index(target) + 1)

print("time: ", time.time() - start)
```

포켓몬의 이름을 받아오는 부분에 사용된 반복문은 어쩔 수 없는 부분이고

M개의 문제를 푸는 부분에 사용된 반복문도 어쩔 수 없는 부분이다.

숫자를 받을 경우에 인덱스로 포켓몬 이름을 받아오는 부분도 O(1)의 최소 시간 복잡도이기에

결국에 시간을 줄일 수 있는 부분은 이름을 가져오는 부분에서 index 메서드를 사용하는 부분이라고 생각이 들었다.

위 코드에 사용된 시간 복잡도는 O(N^2)이기 때문에

O(N) 시간 복잡도를 갖는 index 대신 O(logN), O(1) 복잡도를 갖는 연산을 사용하여야 한다.

2중 반복문을 사용한 것도 아니고

그리 복잡한 코드도 아닌데 시간 초과가 발생한 것이 의문이라 함수의 시간 복잡도를 찾아보았다.

찾아보니 시간 복잡도는 리스트 자료형 보다 딕셔너리를 사용하는 것이 더 유리했다.

딕셔너리를 사용해도 포켓몬의 이름에 대한 인덱스 값을 가져오기 위해서는 O(N) 이상의 시간 복잡도를 사용해야 했다.

따라서 O(log N)의 시간 복잡도를 갖는 이진탐색을 사용할 수 있는 방법을 고민해보았다.

이진탐색을 사용하려면 대상 리스트가 먼저 정렬이 되어있어야 한다.

사전을 이용하는 편이 시간 복잡도에서 유리하기에 사전을 사용하기로 결정했다.

따라서 문자열로 이루어진 사전을 문자열의 순서대로 정렬한 후에 이진탐색을 진행하는 알고리즘을 만들었다.

주어진 알파벳에 대응하는 번호를 출력하는 알고리즘의 순서는 아래와 같다.

1) 딕셔너리에 포켓몬의 번호를 key로, 이름을 value로 하여 저장한다.

Copy

```python
pokemon = {}

for i in range(N):
    pokemon[str(i + 1)] = sys.stdin.readline().strip()
```

key 값은 인덱스 값과 혼동되는 것을 방지하기 위하여 str 자료형으로 넣어주었다.

2) 딕셔너리의 value 값을 기준으로 딕셔너리를 정렬하여 리스트로 저장한다.

```python
pokemon_list = sorted(pokemon.items(), key=lambda x: x[1])
```

문자열 정렬도 숫자 정렬하는 것과 다르지 않게 sorted() 함수에 넣어주면 알파벳의 사전 순대로 정렬된다.

items()와 lambda를 같이 써서 딕셔너리의 키 값으로 정렬을 수행할 수 있다.

원래 자료형인 딕셔너리가 아니라 리스트로 저장하는 이유는

후에 이진 탐색 함수에 넣을 때 딕셔너리로 넣어주면 인덱스를 사용하지 못하기 때문이다.

리스트로 넣어주어야 순서대로 정렬된 문자열의 인덱스를 사용하여 탐색을 수행할 수 있다.

3) 리스트에서 target을 찾는 이진 탐색을 수행한다.

```python
def binary_search(dic, target, start, end):
    if start > end:
        return None
    mid = (start + end) // 2
    if dic[mid][1] == target:
        return dic[mid][0]
    elif dic[mid][1] > target:
        return binary_search(dic, target, start, mid - 1)
    else:
        return binary_search(dic, target, mid + 1, end)

print(binary_search(pokemon_list, target, 0, N - 1))
```

일반적인 이진 탐색과 다르게 이번 이진 탐색은 딕셔너리 자료형의 문자열을 기준으로 한 이진 탐색이다.

하지만 크게 다르지 않다.

pokemon_list에 데이터가 저장된 꼴은 아래와 같다.

```python
[('24', 'Arbok'), ('15', 'Beedrill'), ('9', 'Blastoise'), ('1', 'Bulbasaur'), ('12', 'Butterfree')
, ('10', 'Caterpie'), ('6', 'Charizard'), ('4', 'Charmander'), ('5', 'Charmeleon'), ('23', 'Ekans')
, ('22', 'Fearow'), ('2', 'Ivysaur'), ('14', 'Kakuna'), ('11', 'Metapod'), ('18', 'Pidgeot')
, ('17', 'Pidgeotto'), ('16', 'Pidgey'), ('25', 'Pikachu'), ('26', 'Raichu'), ('20', 'Raticate')
, ('19', 'Rattata'), ('21', 'Spearow'), ('7', 'Squirtle'), ('3', 'Venusaur'), ('8', 'Wartortle'), ('13', 'Weedle')]
```

target이 문자열이기 때문에 각 요소의 두 번째 값과 비교되도록 인덱싱을 잘 해주면 된다.

그리고 주어진 번호에 대응하는 이름을 출력하는 알고리즘은 쉽다.

처음 받아온 딕셔너리에서 번호를 키로 하는 값을 출력해주면 된다.

```python
if target.isdigit():
```

```
        print(pokemon[target])
```
python

정답 코드

전체적인 코드는 아래와 같다.

Copy

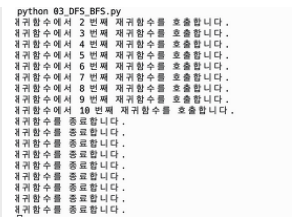
```python
import sys

def binary_search(dic, target, start, end):
    if start > end:
        return None
    mid = (start + end) // 2
    if dic[mid][1] == target:
        return dic[mid][0]
    elif dic[mid][1] > target:
        return binary_search(dic, target, start, mid - 1)
    else:
        return binary_search(dic, target, mid + 1, end)

N, M = list(map(int, sys.stdin.readline().strip().split()))

pokemon = {}

for i in range(N):
    pokemon[str(i + 1)] = sys.stdin.readline().strip()

pokemon_list = sorted(pokemon.items(), key=lambda x: x[1])

for _ in range(M):
    target = sys.stdin.readline().strip()

    if target.isdigit():
        print(pokemon[target])
    else:
        print(binary_search(pokemon_list, target, 0, N - 1))
```
python

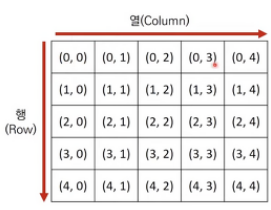
제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
68971437	aprilwldnjs	1620	맞았습니다!!	57692 KB	520 ms	Python 3 / 수정	738 B	24분 전
68965160	aprilwldnjs	1620	시간 초과			Python 3 / 수정	312 B	2시간 전

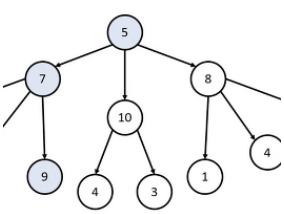
'Algorithm > 알고리즘' 카테고리의 다른 글

[휴학 중 코테 부수기] # 5. 정렬 알고리즘 (0) 2023.11.06
[휴학 중 코테 부수기] # 4. 그래프 탐색 알고리즘: DFS/BFS (1) 2023.11.06
[휴학 중 코테 부수기] # 3. 구현(Implementation) (1) 2023.11.04
[휴학 중 코테 부수기] # 2. 그리디 알고리즘 (3) 2023.10.23
[알고리즘](Python) 5가지 정렬 알고리즘: 선택, 버블, 병합, 퀵 정렬 (0) 2023.10.03

'Algorithm/알고리즘' Related Articles







[휴학 중 코테 부수기] # 5. 정렬 알고리즘
[휴학 중 코테 부수기] # 4. 그래프 탐색 알고리즘:…
[휴학 중 코테 부수기] # 3. 구현(Implementation)
[휴학 중 코테 부수기] # 2. 그리디 알고리즘

Secret

안녕하세요! 어떤 댓글이든 환영합니다!

댓글달기