BOJ

#1759. 암호 만들기

https://www.acmicpc.net/problem/1759

25.04.14

📝 Problem 시간: 1시간 15분

#1759. 암호 만들기

암호 만들기 성공 다국어

☆ 한국어

골드 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	88037	42320	29011	44.966%

문제

바로 어제 최백준 조교가 방 열쇠를 주머니에 넣은 채 깜빡하고 서울로 가 버리는 황당한 상황에 직면한 조교들은, 702호에 새로운 보안 시스템을 설치하기로 하였다. 이 보안 시스템은 열쇠가 아닌 암호로 동작하게 되어 있는 시스템이다.

암호는 서로 다른 L개의 알파벳 소문자들로 구성되며 최소 한 개의 모음(a, e, i, o, u)과 최소 두 개의 자음으로 구성되어 있다고 알려져 있다. 또한 정렬된 문자열을 선호하는 조교들의 성향으로 미루어 보아 암호를 이루는 알파벳이 암호에서 증가하는 순서로 배열되었을 것이라고 추측된다. 즉, abc는 가능성이 있는 암호이지만 bac는 그렇지 않다.

새 보안 시스템에서 조교들이 암호로 사용했을 법한 문자의 종류는 C가지가 있다고 한다. 이 알파벳을 입수한 민식, 영식 형제는 조교들의 방에 침투하기 위해 암호를 추측해 보려고 한다. C개의 문자들이 모두 주어졌을 때, 가능성 있는 암호들을 모두 구하는 프로그램을 작성하시오.

입력

첫째 줄에 두 정수 L, C가 주어진다. (3 ≤ L ≤ C ≤ 15) 다음 줄에는 C개의 문자들이 공백으로 구분되어 주어진다. 주어지는 문자들은 알파벳 소문자이며, 중복되는 것은 없다.

출력

각 줄에 하나씩, 사전식으로 가능성 있는 암호를 모두 출력한다.

예제 입력 1 복사

```
4 6
a t c i s w
```

예제 출력 1 복사

```
acis
acit
aciw
acst
acsw
```

💡 Hint

Step 1. 문제 유형

💡 문제 요약

조건에 맞는 **암호**를 찾는 문제이다.

- 총 L자리이며, C개의 알파벳 중에서 L개를 **조합**하여 만든다.
- 반드시 **오름차순** 정렬되어야 한다. (ex: abc는 가능, bac는 불가능)
- **모음이 최소 1개**, **자음이 최소 2개** 포함되어야 한다.
- 가능한 모든 암호를 **사전 순으로 출력**해야 한다.

💡 문제 유형

- 단순히 C개의 알파벳 중 L개를 뽑는 **조합 문제**처럼 보이지만,
- 모음/자음 조건과 정렬 조건이 있어 **백트래킹 방식**으로 구현해야 한다.

💡 Hint

Step 2. 문제 유형

💡 풀이 아이디어

1. **입력받은 문자들을 오름차순으로 정렬**
   → 백트래킹을 하면서 사전 순 출력을 위함

2. **백트래킹**
   - 현재까지 만든 암호의 길이가 L이면 종료 조건
   - 모음/자음 개수를 체크해서 조건을 만족할 경우 출력
   - 그렇지 않으면 다음 알파벳을 선택해 재귀 호출

3. **모음/자음 판별 함수 필요**
   - 모음: a, e, i, o, u
   - 모음 개수 ≥ 1, 자음 개수 ≥ 2 인지 확인하는 유효성 검사 함수 사용

💡 Hint

Step 3. 백트래킹

💡 백트래킹 기본 구조

```
def backtrack(path, start):
    # 1. 종료 조건 (필요시 조건 추가)
    if 종료조건:
        결과저장(path)
        return
    # 2. 가능한 모든 선택에 대해 탐색
    for i in range(start, len(데이터)):
        if 조건에_맞지_않으면_건너뛰기:
            continue
        # 3. 선택
        path.append(데이터[i])

        # 4. 재귀 호출 (다음 단계로 이동)
        backtrack(path, i+1)  # 조합의 경우
        # backtrack(path, i)    # 중복 조합
        # backtrack(path, 0)    # 중복 순열
        # backtrack(path, used) # 순열 등 복잡한 조건이면 상태값 함께 넘기기
        # 5. 선택 취소 (백트래킹)
        path.pop()
```

Hint

Step 4. 코드 플로우

1. **입력 처리**
    - 암호 길이 L, 전체 알파벳 개수 C를 입력받고
    - 알파벳 리스트를 사전 순으로 정렬한다.
2. **유효성 검사 함수 is_valid 정의**
    - 모음이 최소 1개 이상, 자음이 최소 2개 이상인지 확인한다.
3. **백트래킹 함수 backtrack 정의**
    - 현재 암호 길이가 L이면 종료
    - 유효성 검사 후 조건에 맞으면 출력
    - 반복문을 통해 알파벳을 추가하고, 재귀 호출로 조합 생성
4. **초기 호출**
    - backtrack([], 0)으로 시작한다.

✨Solution

💡 백트래킹 풀이

https://jih3508.tistory.com/149

```python
# 모음 개수 구하는 함수
def check_password(password):
    count = 0
    for char in password:
        if char in {"a", "e", "i", "o", "u"}:
            count += 1
    return count


def make_password(depth, password):
    if len(password) == L:
        count = check_password(password)
        if count >= 1 and L - count >= 2:
            print(password)
        return
    for i in range(depth, C):
        make_password(i + 1, password + alpabet[i])


L, C = map(int, input().split())
alpabet = sorted(list(input().split()))
make_password(0, "")
```

민정 풀이

```python
L, C = map(int, input().split())  # L: 암호 길이, C: 문자 종류
chars = sorted(input().split())   # 사전순 정렬
vowels = {'a', 'e', 'i', 'o', 'u'}

def isvalid(word):
    # 최소 한 개의 모음과 최소 두 개의 자음으로 구성되어있는지 확인
    vowel_cnt, consonant_cnt = 0, 0  # 모음 개수, 자음 개수
    for w in word:
        if w in vowels:
            vowel_cnt += 1
        else:
            consonant_cnt += 1

    return vowel_cnt >= 1 and consonant_cnt >= 2

def backtrack(word, start):
    if len(word) == L:  # 종료 조건
        if isvalid(word):
            print(''.join(word))
        return

    for i in range(start, C):
        word.append(chars[i])
        backtrack(word, i+1)
        word.pop()

backtrack([], 0)
```

✨Solution

💡 Combination 풀이

https://velog.io/@dlgosla/%EB%B0%B1%EC%A4%80-BOJ-%EC%95%94%ED%98%B8-%EB%A7%8C%EB%93%A4%EA%B8%B01759-python

1. 가능한 모든 암호 조합을 구함 (주어진 알파벳 중에서 L개를 순서없이 중복없이 뽑음)(이 때 sort 후에 comb를 함으로써 결과가 암호 내부에서도, 각 암호끼리도 정렬되도록 함)
2. 각 조합들을 for문으로 돌면서 조건에 맞는 지 판별 후 맞으면 출력

```python
from itertools import combinations

L, C = map(int, input().split())

alphabets = input().split()

# 길이가 L인 모든 조합, 증가하는 순서로 배열해야되기 때문에 sort 후 comb
alpha_combs = combinations(sorted(alphabets), L)

answer = []

for alpha_comb in alpha_combs:  # 가능한 조합 중에서
    consonant_count = 0
    vowel_count = 0

    # 자음 모음 개수 세기
    for alpha in alpha_comb:
        if alpha in "aeiou":
            consonant_count += 1
        else:
            vowel_count += 1

    # 모음이 1개 이상, 자음이 2 개 이상이면 출력
    if consonant_count >= 1 and vowel_count >= 2:
        print("".join(alpha_comb))
```

✨Solution

💡 일반 DFS 재귀 풀이

https://velog.io/@dlgosla/%EB%B0%B1%EC%A4%80-BOJ-%EC%95%94%ED%98%B8-%EB%A7%8C%EB%93%A4%EA%B8%B01759-python

1. 주이진 암호 길이가 되면 자음 모음 개수를 세서 조건에 만족하면 출력한다.
2. 주어진 암호 길이보다 작으면 현재 제일 끝에 있는 알파벳보다사전 순으로 더 큰 알파벳을 골라서 추가해서 dfs를 돌린다.
3. 1,2번을 계속 반복한다.

```python
L, C = map(int, input().split())

alphabets = sorted(input().split())


def dfs(idx, codes):
    if L == idx:
        vowel_count = 0
        consonant_count = 0

        # 자음 모음 개수 세기
        for code in codes:
            if code in "aeiou":
                consonant_count += 1
            else:
                vowel_count += 1

        # 자음 2개 이상, 모음 한개 이상이면 암호가 될 수 있으므로 출력
        if consonant_count >= 1 and vowel_count >= 2:
            print("".join(codes))

    else:
        for i in range(idx, C):
            if codes and alphabets[i] <= codes[-1]:  # 오름차순 아니면 버림
                continue
            dfs(idx + 1, codes + [alphabets[i]])


dfs(0, [])
```

💻 Assignment

백준 #6603. 로또 (실버2)

백트래킹 + 조합(Combination) 유형 문제

발제 문제와 비슷한 유형으로 난이도 쉬운 버전

비교	#1759. 암호 만들기	#6603. 로또
조합	✅ L개 선택	✅ 6개 선택
조건 필터링	모음/자음 개수	없음
사전순 정렬	필요 (입력 후 정렬)	입력이 이미 정렬
백트래킹 방식	인덱스로 조합 생성	동일
추가 조건	알파벳 조합 + 필터	숫자 조합