BOJ1316 그룹 단어 체커

상태	시작 전

count는 단순 빈도만 체크해서 그룹 단어 판정 불가. 예를 보여줘

예시 1: `aba`

- 그룹 단어가 아닌 이유 → `a`가 처음 나오고, `b`가 나오고, 다시 `a`가 등장했기 때문.
- `count`는:

```
word = "aba"
print(word.count("a"))  # 2
```

`"a"`가 2번 나왔다고만 알려줍니다.

→ 하지만 이 `"a"`가 연속된 2번(`aa`)인지, 떨어져서 나온 2번(`aba`)인지는 구분 못 함.

> aba의 경우는 a 이전에 a의 개수가 0이 아니므로 그룹 단어가 아니고, aabbbccb는 이전 b의 개수가 0이 아닌 3이므로 그룹 단어가 아니고, ccazzzzbb는 a 이전에 a의 개수가 0이므로 그룹 단어이다. 이런 아이디어로 코딩해볼 수는 없을까?

핵심은

현재 문자가 새로 시작되는 순간

(=바로 앞 문자와 다를 때) 그 문자가

과거에 한 번이라도 나왔는지

확인하면 됩니다.

이때 “과거 개수 > 0”이면 그룹 단어가 아닌 거죠. 다만

바로 앞 문자와 같을 때는

같은 블록이 계속되는 중이니 검사하면 안 됩니다.

```
import sys
from collections import defaultdict
input = sys.stdin.readline
```

```
def is_group(word: str) -> bool:
    seen_cnt = defaultdict(int)
    prev = None
    for ch in word.strip():
        # 블록 경계에서만 과거 등장 여부를 검사
        if ch != prev and seen_cnt[ch] > 0:
            return False
        seen_cnt[ch] += 1
        prev = ch
    return True

n = int(input().strip())
ans = 0
for _ in range(n):
    if is_group(input().strip()):
        ans += 1
print(ans)
```

- `ch != prev`일 때만 `seen_cnt[ch] > 0`을 검사하는 게 포인트입니다.
- 이렇게 하면 `aaa`처럼 같은 블록 안에서의 반복은 허용되고, `aba`처럼 다른 문자가 끼었다가 다시 나오면 바로 잡아냅니다.

```
def is_group(word: str) -> bool:
    seen = set()
    prev = None
    for ch in word.strip():
        if ch != prev:
            if ch in seen:   # 과거에 한 번이라도 끝난 블록이 있었음
                return False
            seen.add(ch)
            prev = ch
    return True
```

위와 완전히 같은 원리이고, 메모리/속도도 좋습니다.

```
import sys
input = sys.stdin.readline

def is_group(s):
    seen, prev = set(), None
    for c in s.strip():
        if c != prev and c in seen:   # 블록이 바뀌는데 과거에 나온 문자면 탈락
            return False
        seen.add(c)
        prev = c
    return True

n = int(input().strip())
print(sum(is_group(input()) for _ in range(n)))
```

“새 블록이 시작될 때 과거에 나온 적 있는지”만 확인한다는 발상 그대로야