P11

21700477 윤다은

문제정의

주어진 단어들이 사전순으로 끝말잇기가 가능한지 보는 문제

아이디어

단어를 사전순으로 정렬하고, 정렬한 단어들이 이어질 수 있는지 검사한다.

세부구현

words[n] : 주어진 단어들을 저장하는 배열
graph[n][n] : 단어간의 연결 상태를 나타내는 2차원 배열
answer : 이어진 단어를 저장하는 벡터
sorted_index[n] : 단어를 사전순으로 정렬했을 때, 인덱스의 번호를 저장하는 배열

입력

단어를 입력받아 words에 저장한다(이때 words의 id는 들어온 순서대로 0부터 n-1까지로 한다 .

단어 간 연결성 파악하기

모든 단어 쌍에 대해서 첫말과 끝말이 이어질 수 있는지 검사한다.
비교하고자 하는 단어쌍이 (s1,s2)로 주어진다면
s1[s1.size()-1] == s2[0]이면 true 아닐 시 false를 반환한다.

두 단어가 이어진다면 graph의 s1의 아이디를 행 번호 s2의 아이디를 열 번호로 하여 해당 배열의 원소를 1로 업데이트한다.

단어들을 사전순으로 정렬하기

i(0≤i≤n-1)에 대해서 sorted_index[i]를 i로 초기화한다.
words의 모든 쌍(s1, s2)을 비교하면서 s1이 s2보다 사전순으로 뒤에 있는 경우 sorted_index에서 s1과 s2에 해당하는 id를 swap한다.
두 단어가 사전순인지 비교하는 방법은 다음과 같다.
두 단어의 한글자씩 비교하여 만약 다른 글자가 나올 경우 두 문자의 아스키코드를 비교하여, 순서를 파악한다.
만약 s1이 s2의 일부분이라면 null과 알파벳을 비교하게 되므로 s2가 더 뒤에 나온다.

사전순으로 정렬한 단어간 연결성 검사

i(0≤i≤n-1)에 대해서 graph[sorted_index[i]][sorted_index[i+1]] == 1인지 확인하여 만약 만족하지 못한다면 반복문을 탈출하고 만족한다면 answer에 words[sorted_index[i]]를 집어넣는다. 반복문을 나왔을 때, 벡터의 사이즈가 n-1개라면 words[sorted_index[n-1]]을 추가하고 answer를 출력한다. 그렇지 않은 경우 0을 출력한다.

시간복잡도

입력 : O(n)
단어 간 연결성 : 모든 쌍에 대해 탐색하므로 $O(n^2)$이다.
사전 순 정렬 : 모든 쌍에 대해 검색하므로 $O(n^2)$이다.
연결성 검사 : O(n)
따라서 $O(n^2)$이다.

공간복잡도

1차원 배열과 2차원배열을 사용하므로 최종적인 공간복잡도는 $O(n^2)$이다.

토의

문제에 least가 가장 작은 이라는 뜻이라고 이해하여, 가장작은 사전순이 어떤 말인지 잘 이해를 못하였다.