<https://leetcode.com/problems/verifying-an-alien-dictionary/description/>

In an alien language, surprisingly, they also use English lowercase letters, but possibly in a different order. The order of the alphabet is some permutation of lowercase letters.

Given a sequence of words written in the alien language, and the order of the alphabet, return true if and only if the given words are sorted lexicographically in this alien language.

**Example 1:**

Input: words = ["hello","leetcode"], order = "hlabcdefgijkmnopqrstuvwxyz"

Output: true

Explanation: As 'h' comes before 'l' in this language, then the sequence is sorted.

**Example 2:**

Input: words = ["word","world","row"], order = "worldabcefghijkmnpqstuvxyz"

Output: false

Explanation: As 'd' comes after 'l' in this language, then words[0] > words[1], hence the sequence is unsorted.

**Example 3:**

Input: words = ["apple","app"], order = "abcdefghijklmnopqrstuvwxyz"

Output: false

Explanation: The first three characters "app" match, and the second string is shorter (in size.) According to lexicographical rules "apple" > "app", because 'l' > '∅', where '∅' is defined as the blank character which is less than any other character (More info).

**Constraints:**

* 1 <= words.length <= 100
* 1 <= words[i].length <= 20
* order.length == 26
* All characters in words[i] and order are English lowercase letters.

**Attempt 1: 2023-02-01**

**Solution 1:  Hash Table (30 min)**

class Solution {

public boolean isAlienSorted(String[] words, String order) {

Map<Character, Integer> map = new HashMap<Character, Integer>();

for(int i = 0; i < order.length(); i++) {

map.put(order.charAt(i), i);

}

int n = words.length;

if(n <= 1) {

return true;

}

for(int i = 0; i < n - 1; i++) {

if(!compare(words[i], words[i + 1], map)) {

return false;

}

}

return true;

}

// Return 'true' means s1 and s2 sorted, 'false' means not not sorted

private boolean compare(String s1, String s2, Map<Character, Integer> map) {

int len1 = s1.length();

int len2 = s2.length();

int len = Math.min(len1, len2);

for(int i = 0; i < len; i++) {

char c1 = s1.charAt(i);

char c2 = s2.charAt(i);

// character has higher priority from left leaning, so first hit the logic

// if s1 has a character strictly smaller than the character in s2 at the

// same position will directly return true, ONLY when find a character in

// s1 strictly larger than character in s2 at the same poistion will directly

// return false, that's the second logic only after first one, also when

// character in s1 and s2 at the same poistion equal just ignore

if(map.get(c1) < map.get(c2)) {

return true;

} else if(map.get(c1) > map.get(c2)) {

return false;

}

}

if(len1 - len > 0) {

return false;

}

return true;

}

}

**Refer to**

<https://leetcode.com/problems/verifying-an-alien-dictionary/solutions/203185/java-c-python-mapping-to-normal-order/>

Mapping to Normal Order

## **Explanation**

Build a transform mapping from order,Find all alien words with letters in normal order.

For example, if we have order = "xyz..."We can map the word "xyz" to "abc" or "123"

Then we check if all words are in sorted order.

## **Complexity**

Time O(NS)Space O(1)

**Java**

int[] mapping = new int[26];

public boolean isAlienSorted(String[] words, String order) {

for (int i = 0; i < order.length(); i++)

mapping[order.charAt(i) - 'a'] = i;

for (int i = 1; i < words.length; i++)

if (bigger(words[i - 1], words[i]))

return false;

return true;

}

boolean bigger(String s1, String s2) {

int n = s1.length(), m = s2.length();

for (int i = 0; i < n && i < m; ++i)

if (s1.charAt(i) != s2.charAt(i))

return mapping[s1.charAt(i) - 'a'] > mapping[s2.charAt(i) - 'a'];

return n > m;

}