Problem 756: Pyramid Transition Matrix

Problem Information

Difficulty: Medium
Acceptance Rate: 53.21%
Paid Only: No
Tags: Hash Table, String, Backtracking, Bit Manipulation

Problem Description

You are stacking blocks to form a pyramid. Each block has a color, which is represented by a single letter. Each row of blocks contains **one less block** than the row beneath it and is centered on top.

To make the pyramid aesthetically pleasing, there are only specific **triangular patterns** that are allowed. A triangular pattern consists of a **single block** stacked on top of **two blocks**. The patterns are given as a list of three-letter strings `allowed`, where the first two characters of a pattern represent the left and right bottom blocks respectively, and the third character is the top block.

* For example, `"ABC"` represents a triangular pattern with a `'C'` block stacked on top of an `'A'` (left) and `'B'` (right) block. Note that this is different from `"BAC"` where `'B'` is on the left bottom and `'A'` is on the right bottom.

You start with a bottom row of blocks `bottom`, given as a single string, that you **must** use as the base of the pyramid.

Given `bottom` and `allowed`, return `true` _if you can build the pyramid all the way to the top such that**every triangular pattern** in the pyramid is in _`allowed` _, or_`false` _otherwise_.

Example 1:

![](https://assets.leetcode.com/uploads/2021/08/26/pyramid1-grid.jpg)

Input: bottom = "BCD", allowed = ["BCC","CDE","CEA","FFF"] **Output:** true **Explanation:** The allowed triangular patterns are shown on the right. Starting from the bottom (level 3), we can build "CE" on level 2 and then build "A" on level 1. There are three triangular patterns in the pyramid, which are "BCC", "CDE", and "CEA". All are allowed.

Example 2:

![](https://assets.leetcode.com/uploads/2021/08/26/pyramid2-grid.jpg)

Input: bottom = "AAAA", allowed = ["AAB","AAC","BCD","BBE","DEF"] **Output:** false **Explanation:** The allowed triangular patterns are shown on the right. Starting from the bottom (level 4), there are multiple ways to build level 3, but trying all the possibilites, you will get always stuck before building level 1.

Constraints:

* `2 <= bottom.length <= 6` * `0 <= allowed.length <= 216` * `allowed[i].length == 3` * The letters in all input strings are from the set `{'A', 'B', 'C', 'D', 'E', 'F'}`. * All the values of `allowed` are **unique**.

Code Snippets

C++:

```
class Solution {
public:
bool pyramidTransition(string bottom, vector<string>& allowed) {

}
};
```

Java:

```
class Solution {
public boolean pyramidTransition(String bottom, List<String> allowed) {

}
}
```

Python3:

```
class Solution:
def pyramidTransition(self, bottom: str, allowed: List[str]) -> bool:
```