62. Number of Ways of Cutting a Pizza

Given a rectangular pizza represented as a rows x cols matrix containing the following characters: 'A' (an apple) and '.' (empty cell) and given the integer k. You have to cut the pizza into k pieces using k-1 cuts.

For each cut you choose the direction: vertical or horizontal, then you choose a cut position at the cell boundary and cut the pizza into two pieces. If you cut the pizza vertically, give the left part of the pizza to a person. If you cut the pizza horizontally, give the upper part of the pizza to a person. Give the last piece of pizza to the last person.

Return the number of ways of cutting the pizza such that each piece contains at least one apple. Since the answer can be a huge number, return this modulo 10^9 + 7.

Program:

```
def ways(pizza, k):
    MOD = 10**9 + 7
    rows, cols = len(pizza), len(pizza[0])

    # Precompute the prefix sums for apples in the pizza
    apples = [[0] * (cols + 1) for _ in range(rows + 1)]
    for r in range(rows - 1, -1, -1):
        for c in range(cols - 1, -1, -1):
            apples[r][c] = (pizza[r][c] == 'A') + apples[r + 1][c] + apples[r][c + 1] - apples[r + 1][c + 1]

    # Memoization table
    memo = {}

    def dp(r, c, cuts):
        if apples[r][c] == 0:
            return 0
        if cuts == 0:
            return 1
        if (r, c, cuts) in memo:
            return memo[(r, c, cuts)]

        result = 0
```

```
        # Make horizontal cuts
        for nr in range(r + 1, rows):
            if apples[r][c] - apples[nr][c] > 0:
                result = (result + dp(nr, c, cuts - 1)) % MOD

        # Make vertical cuts
        for nc in range(c + 1, cols):
            if apples[r][c] - apples[r][nc] > 0:
                result = (result + dp(r, nc, cuts - 1)) % MOD

        memo[(r, c, cuts)] = result
        return result

    return dp(0, 0, k - 1)

# Example usage
pizza = [
    "A..",
    "AAA",
    "...",
]
k = 3
print(ways(pizza, k))  # Output: 3

pizza = [
    "A..",
    "AA.",
    "...",
]
k = 3
print(ways(pizza, k))  # Output: 1
```

pizza = [

"A..",

"A..",

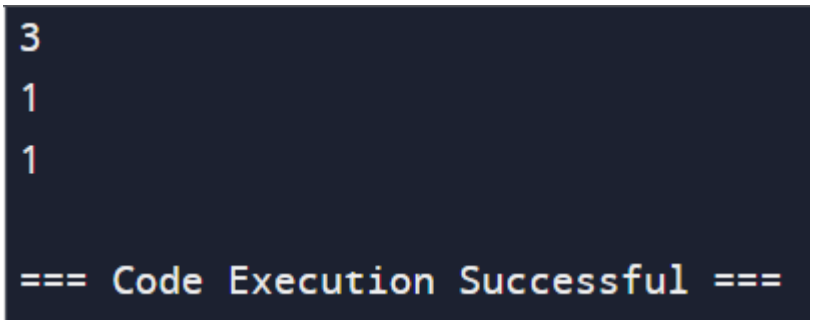
"..."

]

k = 1

print(ways(pizza, k)) # Output: 1

Output:



Timecomplexity:O(xyrows x colum x(rows+colum))