4.8 WORD BREAK PROBLEM USING DYNAMIC PROGRAMMING

Question:

Given a string s and a dictionary of strings wordDict, return true if s can be segmented into a space-separated sequence of one or more dictionary words.

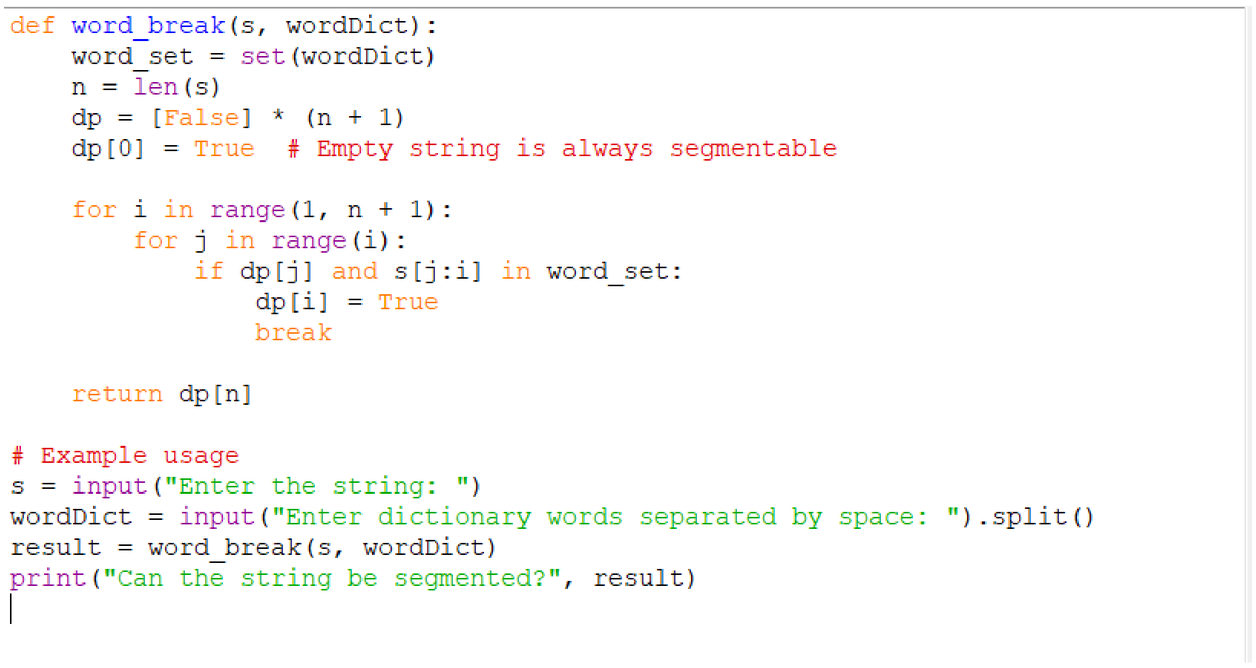
AIM

To implement a Python program that determines whether a string can be segmented into valid dictionary words using dynamic programming.

ALGORITHM

1. Convert wordDict into a set for faster lookup.
2. Initialize a boolean array dp of size len(s)+1, where dp[i] is True if the substring s[0:i] can be segmented.
3. Set dp[0] = True (empty string is segmentable).
4. For each index i from 1 to len(s), check all j < i such that:
    - dp[j] == True and
    - s[j:i] is in the dictionary
    - If both conditions are met, set dp[i] = True
5. Return dp[len(s)] as the result.

PROGRAM



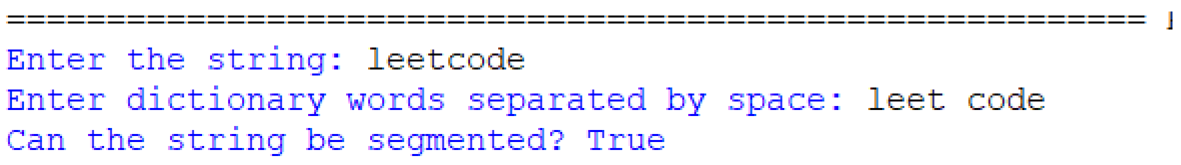
```
def word_break(s, wordDict):
    word_set = set(wordDict)
    n = len(s)
    dp = [False] * (n + 1)
    dp[0] = True  # Empty string is always segmentable

    for i in range(1, n + 1):
        for j in range(i):
            if dp[j] and s[j:i] in word_set:
                dp[i] = True
                break

    return dp[n]

# Example usage
s = input("Enter the string: ")
wordDict = input("Enter dictionary words separated by space: ").split()
result = word_break(s, wordDict)
print("Can the string be segmented?", result)
```

Input:

Enter the string: leetcode

Enter dictionary words separated by space: leet code

Output:



```
============================================================
Enter the string: leetcode
Enter dictionary words separated by space: leet code
Can the string be segmented? True
```

RESULT:

Thus the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity: $O(n^2)$
- Space Complexity: $O(n)$