**139. Word Break :-**

Medium Accepted: 1.4M Submissions: 3.1M Acceptance Rate: 46.0%

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.

**Note** that the same word in the dictionary may be reused multiple times in the segmentation.

**Example 1:**

**Input:** s = "leetcode", wordDict = ["leet","code"]

**Output:** true

**Explanation:** Return true because "leetcode" can be segmented as "leet code".

**Example 2:**

**Input:** s = "applepenapple", wordDict = ["apple","pen"]

**Output:** true

**Explanation:** Return true because "applepenapple" can be segmented as "apple pen apple".

Note that you are allowed to reuse a dictionary word.

**Example 3:**

**Input:** s = "catsandog", wordDict = ["cats","dog","sand","and","cat"]

**Output:** false

**Constraints:**

* 1 <= s.length <= 300
* 1 <= wordDict.length <= 1000
* 1 <= wordDict[i].length <= 20
* s and wordDict[i] consist of only lowercase English letters.
* All the strings of wordDict are **unique**.

**Code :-**

class Solution {

public:

    bool func(string &s, unordered\_map<string,bool> &mp, int start, vector<int> &dp){

        int n=s.size();

        if(start==n)

            return 1;

        if(dp[start] != -1)

            return dp[start];

        string temp="";

        for(auto i=start; i<n; i++){

            temp += s[i];

            if(mp[temp]==true && func(s, mp, i+1, dp)==1){

                dp[i] = 1;

                return dp[i];

            }

        }

        dp[start]=0;

        return dp[start];

    }

    bool wordBreak(string s, vector<string>& dict) {

        unordered\_map<string,bool> mp;

        vector<int> dp(s.size(), -1);

        for(auto &s:dict)

            mp[s] = true;

        return func(s, mp, 0, dp);

    }

};