Description | Editorial | Solutions (4.6K) | Submissions

139. Word Break

Medium | 14.5K | 607

Companies

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:

Java | Auto

```java
class Solution {
    public boolean wordBreak(String s, List<String> wordDict) {
        HashMap<String,Integer>h1=new HashMap<>();
        for(String str:wordDict){
            h1.put(str,1);
        }
        int n=s.length();
        int dp[]=new int[n+1];
        Arrays.fill(dp,-1);
        return helper(s,0,h1,dp)==1;
    }
    int  helper(String s,int i,HashMap<String,Integer>h1,int dp[]){
        if(i==s.length()){
            return 1;
        }
        if(dp[i]!=-1){
            return dp[i];
        }
        for(int j=i;j<s.length();j++){
            if(isfound(s.substring(i,j+1),h1)){
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

s =
"leetcode"

wordDict =
["leet","code"]

Output

Console | Run | Submit

Description | Editorial | Solutions (4.4K) | Submissions

209. Minimum Size Subarray Sum

Medium | 10.7K | 297

Companies

Given an array of positive integers `nums` and a positive integer `target`, return *the* ***minimal length*** *of a subarray whose sum is greater than or equal to* `target`. If there is no such subarray, return `0` instead.

Example 1:

```
Input: target = 7, nums = [2,3,1,2,4,3]
Output: 2
Explanation: The subarray [4,3] has the minimal length under the problem constraint.
```

Example 2:

```
Input: target = 4, nums = [1,4,4]
Output: 1
```

Example 3:

```
Input: target = 11, nums = [1,1,1,1,1,1,1,1]
Output: 0
```

Constraints:

- `1 <= target <= 10^9`
- `1 <= nums.length <= 10^5`

Java | Auto

```java
class Solution {
    public int minSubArrayLen(int target, int[] nums) {
        int ans = nums.length+1;
        int sum = 0;
        int st = 0;
        for(int i=0;i<nums.length;i++){
            sum += nums[i];
            while(sum >= target && st <= i){
                ans = Math.min(ans , i - st + 1);
                sum -= nums[st++];
            }
        }
        return ans==nums.length+1?0:ans;
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

target =
7

nums =
[2,3,1,2,4,3]

Output

Console | Run | Submit