Word Break Problem using Backtracking

Difficulty Level : Hard ● Last Updated : 06 Jul, 2022

Given a valid sentence without any spaces between the words and a dictionary of valid English words, find all possible ways to break the sentence into individual dictionary words.

Example:

```
Consider the following dictionary
{ i, like, sam, sung, samsung, mobile, ice,
  and, cream, icecream, man, go, mango}

Input: "ilikesamsungmobile"
Output: i like sam sung mobile
        i like samsung mobile

Input: "ilikeicecreamandmango"
Output: i like ice cream and man go
        i like ice cream and mango
        i like icecream and man go
        i like icecream and mango
```

Recommended Practice

Word Break – Part 2

Try It!

We have discussed a Dynamic Programming solution in the below post.
Dynamic Programming | Set 32 (Word Break Problem)

We start scanning the sentence from the left. As we find a valid word, we need to check whether the rest of the sentence can make valid words or not. Because in some situations the first found word from the left side can leave a remaining portion that is not further separable. So, in that case, we should come back and leave the currently found word and keep on searching for the next word. And this process is recursive because to find out whether the right portion is separable or not, we need the same logic. So we will use recursion and backtracking to solve this problem. To keep track of the found words we will use a stack. Whenever the right portion of the string does not make valid words, we pop the top string from the stack and continue finding.

Below is the implementation of the above idea:

C++

```cpp
// A recursive program to print all possible
// partitions of a given string into dictionary
// words
#include <iostream>
using namespace std;

/* A utility function to check whether a word
  is present in dictionary or not.  An array of
  strings is used for dictionary.  Using array
  of strings for dictionary is definitely not
  a good idea. We have used for simplicity of
  the program*/
int dictionaryContains(string &word)
{
    string dictionary[] = {"mobile","samsung","sam","sung",
                            "man","mango", "icecream","and",
                            "go","i","love","ice","cream"};
    int n = sizeof(dictionary)/sizeof(dictionary[0]);
    for (int i = 0; i < n; i++)
        if (dictionary[i].compare(word) == 0)
            return true;
    return false;
}

// Prototype of wordBreakUtil
void wordBreakUtil(string str, int size, string result);
```

```
    // Last argument is prefix
    wordBreakUtil(str, str.size(), "");
}

// Result store the current prefix with spaces
// between words
void wordBreakUtil(string str, int n, string result)
{
    //Process all prefixes one by one
    for (int i=1; i<=n; i++)
    {
        // Extract substring from 0 to i in prefix
        string prefix = str.substr(0, i);

        // If dictionary contains this prefix, then
        // we check for remaining string. Otherwise
        // we ignore this prefix (there is no else for
        // this if) and try next
        if (dictionaryContains(prefix))
        {
            // If no more elements are there, print it
            if (i == n)
            {
                // Add this element to previous prefix
                result += prefix;
                cout << result << endl;
                return;
            }
            wordBreakUtil(str.substr(i, n-i), n-i,
                                    result + prefix + " ");
        }
    }
}

//Driver Code
int main()
{

    // Function call
    cout << "First Test:\n";
    wordBreak("iloveicecreamandmango");

    cout << "\nSecond Test:\n";
    wordBreak("ilovesamsungmobile");
    return 0;
}
```

```java
// A recursive program to print all possible
// partitions of a given string into dictionary
// words
import java.io.*;
import java.util.*;

class GFG {

  // Prints all possible word breaks of given string
  static void wordBreak(int n, List<String> dict, String s)
  {
    String ans="";
    wordBreakUtil(n, s, dict, ans);
  }

  static void wordBreakUtil(int n, String s, List<String> dict, String ans)
  {
    for(int i = 1; i <= n; i++)
    {

      // Extract substring from 0 to i in prefix
      String prefix=s.substring(0, i);

      // If dictionary contains this prefix, then
      // we check for remaining string. Otherwise
      // we ignore this prefix (there is no else for
      // this if) and try next
      if(dict.contains(prefix))
      {
        // If no more elements are there, print it
        if(i == n)
        {

          // Add this element to previous prefix
          ans += prefix;
          System.out.println(ans);
          return;
        }
        wordBreakUtil(n - i, s.substring(i,n), dict, ans+prefix+" ");
      }
    }
  }

  // main function
  public static void main(String args[])
  {
    int n2 = str2.length();                  // length of second string

    // List of strings in dictionary
    List <String> dict= Arrays.asList("mobile","samsung","sam","sung",
                                      "man","mango", "icecream","and",
                                      "go","i","love","ice","cream");
    System.out.println("First Test:");

    // call to the method
    wordBreak(n1,dict,str1);
    System.out.println("\nSecond Test:");

    // call to the method
    wordBreak(n2,dict,str2);
  }
}

// This code is contributed by mohitjha727.
```

Python3

```
# A recursive program to print all possible
# partitions of a given string into dictionary
# words

# A utility function to check whether a word
# is present in dictionary or not.  An array of
# strings is used for dictionary.  Using array
# of strings for dictionary is definitely not
# a good idea. We have used for simplicity of
# the program
def dictionaryContains(word):
    dictionary = {"mobile", "samsung", "sam", "sung", "man",
                  "mango", "icecream", "and", "go", "i", "love", "ice", "cream"}
    return word in dictionary

# Prints all possible word breaks of given string
def wordBreak(string):

    # Last argument is prefix
    wordBreakUtil(string, len(string), "")

# Result store the current prefix with spaces
# between words
def wordBreakUtil(string, n, result):
        # Extract substring from 0 to i in prefix
        prefix = string[:i]

        # If dictionary contains this prefix, then
        # we check for remaining string. Otherwise
        # we ignore this prefix (there is no else for
        # this if) and try next
        if dictionaryContains(prefix):

            # If no more elements are there, print it
            if i == n:

                # Add this element to previous prefix
                result += prefix
                print(result)
                return
            wordBreakUtil(string[i:], n - i, result+prefix+" ")

# Driver Code
if __name__ == "__main__":
    print("First Test:")
    wordBreak("iloveicecreamandmango")

    print("\nSecond Test:")
    wordBreak("ilovesamsungmobile")

# This code is contributed by harshitkap00r
```

C#

```
// A recursive program to print all possible
// partitions of a given string into dictionary
// words
using System;
using System.Collections.Generic;
class GFG {

      // Prints all possible word breaks of given string
      static void wordBreak(int n, List<string> dict, string s)
      {
        string ans="";
        wordBreakUtil(n, s, dict, ans);
      }
```

```
        for(int i = 1; i <= n; i++)
        {

          // Extract substring from 0 to i in prefix
          string prefix=s.Substring(0, i);

          // If dictionary contains this prefix, then
          // we check for remaining string. Otherwise
          // we ignore this prefix (there is no else for
          // this if) and try next
          if(dict.Contains(prefix))
          {
            // If no more elements are there, print it
            if(i == n)
            {

              // Add this element to previous prefix
              ans += prefix;
              Console.WriteLine(ans);
              return;
            }
            wordBreakUtil(n - i, s.Substring(i,n-i), dict, ans+prefix+" ");
          }
        }
      }

  static void Main() {
    string str1 = "iloveicecreamandmango"; // for first test case
    string str2 ="ilovesamsungmobile";     // for second test case
    int n1 = str1.Length;                 // length of first string
    int n2 = str2.Length;                 // length of second string

    // List of strings in dictionary
    List<string> dict= new List<string>(new string[]{"mobile","samsung","sam","su
                                  "man","mango", "icecream","and",
                                  "go","i","love","ice","cream"});
    Console.WriteLine("First Test:");

    // call to the method
    wordBreak(n1,dict,str1);
    Console.WriteLine();
    Console.WriteLine("Second Test:");

    // call to the method
    wordBreak(n2,dict,str2);
  }
}
```

Javascript

```
<script>
// A recursive program to print all possible
// partitions of a given string into dictionary
// words

// Prints all possible word breaks of given string
function wordBreak(n,dict,s)
{
    let ans="";
    wordBreakUtil(n, s, dict, ans);
}

function wordBreakUtil(n,s,dict,ans)
{
    for(let i = 1; i <= n; i++)
    {

      // Extract substring from 0 to i in prefix
      let prefix=s.substring(0, i);

      // If dictionary contains this prefix, then
      // we check for remaining string. Otherwise
      // we ignore this prefix (there is no else for
      // this if) and try next
      if(dict.includes(prefix))
      {
        // If no more elements are there, print it
        if(i == n)
        {

          // Add this element to previous prefix
          ans += prefix;
          document.write(ans+"<br>");
          return;
        }
        wordBreakUtil(n - i, s.substring(i,n), dict, ans+prefix+" ");
      }
    }


 // main function
let  str1 = "iloveicecreamandmango"; // for first test case
```

```
// List of strings in dictionary
let dict= ["mobile","samsung","sam","sung",
                                  "man","mango", "icecream","and",
                                  "go","i","love","ice","cream"];
document.write("First Test:<br>");

// call to the method
wordBreak(n1,dict,str1);
document.write("<br>Second Test:<br>");

// call to the method
wordBreak(n2,dict,str2);

// This code is contributed by avanitrachhadiya2155
</script>
```

Output

```
First Test:
i love ice cream and man go
i love ice cream and mango
i love icecream and man go
i love icecream and mango

Second Test:
i love sam sung mobile
i love samsung mobile
```

Complexities:

- **Time Complexity**: O((2^n)). Because there are 2^n combinations in The Worst Case.
- **Auxiliary Space**: O((n^2)). Because of the Recursive Stack of wordBreakUtil(…) function in The Worst Case.

Where n is the length of the input string.

his article is contributed by **Raghav Jajodia**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail

Start Your Coding Journey Now! Login Register

Test Series for Service-Based Companies
200+ Practice Problems | Mocks + Assessment Tests And Much More!
Practice Now
GeeksforGeeks

Like 57

Previous

Next

AD

Disney+ | Hulu | ESPN+
Get the best deal on the best movies, series, and sports with The Disney Bundle.
Disney+

RECOMMENDED ARTICLES

Page : 1 2 3

01 Word Break Problem | DP-32 | Set - 2
07, May 20

05 Minimum Word Break
17, Nov 17

14, Jul 11

03 **Word Break Problem | (Trie solution)**
10, Sep 17

04 **Travelling Salesman Problem implementation using BackTracking**
10, Apr 19

07 **m Coloring Problem | Backtracking-5**
01, May 12

08 **N Queen Problem | Backtracking-3**
21, Jul 11

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Hard

Easy Normal Medium Hard Expert

Improved By : sreejithsankar55, pratikraut0000, harshitkap00r, mohitjha727, pushvind_1, avanitrachhadiya2155, anikakapoor, divyeshrabadiya07, hardikkoriintern

Article Tags : D-E-Shaw, Google, IBM, Backtracking, Recursion, Strings

Practice Tags : D-E-Shaw, Google, IBM, Strings, Recursion, Backtracking

Improve Article Report Issue

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Start Your Coding Journey Now!

Login

Register