homework → word break

Week 4 - Algorithms - 12-08-2017
- Input string + dictionary of words.
- Find out if the string can be segmented into a space-separated sequence of dictionary words

Example { pear, salmon, foot, points, you, enjoy, sun, girl, leave }
string → "you enjoy"
output → true

string → "youleavefootpoint"
output → true

string → "salmonenjoyapples"
output → false

T-TALK

- What is the input? → string + Vector of strings
- What is the output? → True or False
- Use any library? → yes
- Assume the string can be empty [False]
- Assume the vector contains can be empty [False]
- Assume the vector can have duplicate words and special chars [False]
- Assume there is no difference with lowercase and uppercase
- Assume vector only contains lowercase letters

E - EXAMPLES

Sample Input	Class Equivalent	Out
Hello, [hello, world]	One word sequence	true
youenjoy [pear, you, enjoy]	Two words sequence	true
youleavefootpoints [pear,]	More than two words sequence	true
applesalmon [you, apple]	Unsuccesful sequence	False
hello [apple]	No word dictionary	False
hello!!! [hello, world]	Special chars in string	False
'' [hello, world]	Empty string	False
hello [ ]	Empty vector	False

B - BRUTE FORCE

hello world
[you, hello, apple, world]
h
he
hell
hello
w
wo
wor
worl
world ✓

- Validate input string and vector
- Form all the possible substrings of the given string
  your joy → y, yo, you, your, .., o, ou, our, ...
- Compare each substring to the words in the vector
- if found, delete from string and keep searching for the rest
- If all the chars from the string were deleted return true
  - If not return false

Time complexity $O(2^n)$:(

O - OPTIMIZATION

- Use a data structure to optimize and store value
- Use of dynamic programming to save the comparisons results
  → Use a vector to store results

W-WALK THROUGH

✓ Input Validation → check if string is empty ↗ False
check if vector is empty → False

✓ Preparation → Store words in a unordered_set
→ Initialize a vector of ints in Ø

✓ Loop through word → check if substring is in dictionary
→ if yes → store 1 in vector

✓ Check vector → If last value of vector is 1, return true, else False

I-IMPLEMENTATION

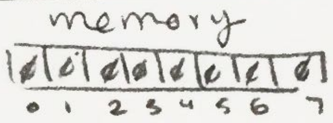
```
bool wordBreak ( string s, vector <string> wordDict ){
    if (S.empty() || wordDict.empty()){
        return False;
    }
    unordercler_set<string> dic;
    for (int i=Ø; i< wordDict.size(); i++){
        dict.insert(wordDict[i]);
    }
    int size= s.size();
    Vector<int> dp(size, Ø);
    for (int i=Ø ; i<size ; i++){
        if (dic.find(s.substr(Ø, i+1)) != dic.end()){
            dp[i]=1;
        }
        else if (i>Ø) {
            for (int j=i; j>Ø; j--){
                if (dp[j-1] && dic.find(S.substr(j, i-j+1)) != dic.end()) {
                    dp[i]=1;
                    break;
                }
            }
        }
    }
    if (dp[size-1]==1){
        return true;
    }
    else{
        return false;
    }
}
```

T-TEST

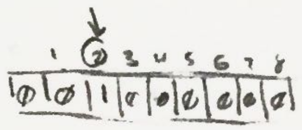
time complex.y (n^2)

Input "", [you, enjoy] → False ✓
Input "youenjoy", [ ] → False ✓
Input ① "youenjoy", [pear, salmon, foot, prints, footprints, leave, you, sun, girl, enjoy] → True ✓
Input "youleavefootprints" + same dict → True ✓
② Input "salmon enjoy apples" + same dict → False ✓

① 1 2 3 4 5 6 7 8
youenjoy → memory



ø → not in dict



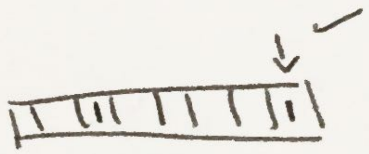
~~e~~you → yes! →

oue
ue
~~you~~e → you already in dict

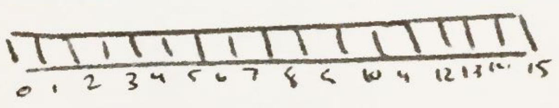
n
en
uen
ouen
youen → you already in dict

.
.
.

y
oy
joy
njoy
enjoy → yes a substring!



② "salmon enjoy apples"
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16



↓
apples is not in dict
↓
False.