CCC Junior Lesson 8

Derrick Guo

Note: This course note is prepared based on questions from www.leetcode.com, under the answer of each question, you can find a reference number of that question on leetcode. When you teach students using slides, let them go to leetcode.com and copy there solutions there to run tests against their programs. That way we can provide better test cases in both quantity and quality.

1 Letter Combinations of a Phone Number

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent.
A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example:

```
Input: "23"
Output: ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"].
```

Idea:

Use DFS. The idea of DFS is to go to the end of one solution, then backtrack to another.

Answer:

```
def letterCombinations(digits):
    if not digits:
        return []
```

```
    dic = {"2":"abc", "3":"def", "4":"ghi", "5":"jkl", "6":"mno", "7":"pqrs", "8
    res = []
    dfs(digits, dic, 0, "", res)
    return res

def dfs(digits, dic, index, path, res):
    if len(path) == len(digits):
        res.append(path)
        return
    for i in xrange(index, len(digits)):
        for j in dic[digits[i]]:
            dfs(digits, dic, i+1, path+j, res)
```

Reference: leetcode #17

2 House Robber

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security system connected and it will automatically contact the police if two adjacent houses were broken into on the same night.
Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight without alerting the police.

Example1:

```
Input: [1,2,3,1]
Output: 4
Explanation: Rob house 1 (money = 1) and then
             rob house 3 (money = 3).
             Total amount you can rob = 1 + 3 = 4.
```

Example2:

```
Input: [2,7,9,3,1]
Output: 12
Explanation: Rob house 1 (money = 2), rob house 3 (money = 9)
```

```
and rob house 5 (money = 1).
Total amount you can rob = 2 + 9 + 1 = 12.
```

Idea:

Use dynamic programming. This question is a introductory level DP problem. Build a dynamic array(or dictionary, in python) to hold the current max value, then keep building up until you hit the final answer. In this case, the maximum amount of money robbed when at ith house is either [the money robbed from beginning to (i-2)th house + money in ith house] or [the money robbed from beginning to (i-1)th house]. Then we can just pick the largest one to be the max at ith house.

Answer:

```
def rob(nums):
    dp={};
    dp[0]=nums[0]
    dp[1]=max(nums[0],nums[1])

    for i in range(2,len(nums)):
        dp[i] = max((dp[i-2]+nums[i]),dp[i-1])
    return dp[len(nums)-1]
```

Reference: leetcode #198

3 Word Break

Given a non-empty string s and a dictionary wordDict containing a list of non-empty words, determine if s can be segmented into a space-separated sequence of one or more dictionary words.
Note:
The same word in the dictionary may be reused multiple times in the segmentation.
You may assume the dictionary does not contain duplicate words

Example:

```
Input: s = "leetcode", wordDict = ["leet", "code"]
Output: true
Explanation: Return true because "leetcode" can be segmented as "leet code".
```

Example:

```
Input: s = "applepenapple", wordDict = ["apple", "pen"]
Output: true
Explanation: Return true because "applepenapple" can be segmented as "apple pen
             Note that you are allowed to reuse a dictionary word.
```

Example:

```
Input: s = "catsandog", wordDict = ["cats", "dog", "sand", "and", "cat"]
Output: false
```

Idea:

Another DP problem. Build a boolean dynamic array to hold the bool value of whether the given word contains the ith word in the wordDict. Keep going up until you hit the final anwser.

Answer:

```
class Solution(object):
    def wordBreak(self, s, wordDict):
        """
        :type s: str
        :type wordDict: List[str]
        :rtype: bool
        """
        dp = [False] * (len(s)+1)
        dp[0] = True
        for i in xrange(1, len(s)+1):
            for j in xrange(i):
                if dp[j] and s[j:i] in wordDict:
                    dp[i] = True
                    break
        return dp[-1]
```

Reference: leetcode #202

4 Jump Game

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.
Determine if you are able to reach the last index.

Example 1:

```
Input: [2,3,1,1,4]
Output: true
Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last
```

Example 1:

```
Input: [3,2,1,0,4]
Output: false
Explanation: You will always arrive at index 3 no matter what. Its m
             jump length is 0, which makes it impossible to reach th
```

Idea: Use a value stepsLeft to hold how far you can get from a certain position. Update it in each iteration since you can get further. If it can't go any further and you are not at the end, return false.

Answer:

```
class Solution:
    # @param {integer[]} nums
    # @return {boolean}
    def canJump(self, nums):
        stepsLeft = nums[0]

        if not stepsLeft and len(nums) > 1:
            return False

        for num in nums[1:-1]:
            stepsLeft = max(stepsLeft - 1, num)
            if not stepsLeft:
                return False

        return True
```

Reference: leetcode #55

5 Search in Rotated Sorted Array II

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.
(i.e., [0,0,1,2,2,5,6] might become [2,5,6,0,0,1,2]).
You are given a target value to search. If found in the array return true, otherwise return false.

Example 1:

```
Input: nums = [2,5,6,0,0,1,2], target = 0
Output: true
```

Example 2:

```
Input: nums = [2,5,6,0,0,1,2], target = 3
Output: false
```

Idea: 3 cases: 1. nums[mid] ¡ nums[r]. 2 cases after this: if the answer is between left and mid, or between mid and right? If between left and mid, change right to mid - 1, else change left to mid + 1.
2. nums[mid] ¿ nums[r]: the above 2 cases still apply here.
3. nums[mid] = nums[r]: this is a tricky case where you have like [1,2,3,3,3,3,3,3]. You can solve this by simply making r = r - 1 until you mid gets to 2.

Answer:

```
def search(self, nums, target):
    if not nums:
        return False
    l, r = 0, len(nums)-1
    while l < r:
        mid = l + (r-l)//2
        if nums[mid] == target:
            return True
        if nums[mid] < nums[r]:
```

```
            if nums[mid] < target <= nums[r]:
                l = mid + 1
            else:
                r = mid - 1
        elif nums[mid] > nums[r]:
            if nums[l] <= target < nums[mid]:
                r = mid - 1
            else:
                l = mid + 1
        else:
            r -= 1
    return nums[l] == target
```

Reference: leetcode # 81