第五周学习总结

动态规划（DP）总结:

1. 一维 DP

   f(n) = f(n-1) + f(n-2)

   Fibonacci 数列

   爬楼梯问题

2. 二维 DP

   最长公共子序问题：

   给定两个字符串 text1 和 text2，返回这两个字符串的最长公共子序列。

   一个字符串的 子序列 是指这样一个新的字符串：它是由原字符串在不改变字符的相对顺序的情况下删除某些字符（也可以不删除任何字符）后组成的新字符串。

   例如，"ace" 是 "abcde" 的子序列，但 "aec" 不是 "abcde" 的子序列。两个字符串的「公共子序列」是这两个字符串所共同拥有的子序列。

   若这两个字符串没有公共子序列，则返回 0。

```
def longestCommonSubsquence(self, text1, text2):
    m = len(text1)
    n = len(text2)
    dp = [ [0] * len(n+1) for _ in range(m+1) ]
    for i in range(1, m+1):
        for j in range(1, n+1):
            if text[i-1] == text[j-1]:
                dp[i][j] = dp[i-1][j-1] + 1
            else:
                dp[i][j] = max(dp[i-1][j], dp[i][j-1])
    return dp[-1][-1]
```

3. DP 的本质: 递归 + 递推 （至底向上规划）

   DP 的两种形式：

   爬楼梯问题：你正在爬楼梯。需要 *n* 阶你才能到达楼顶。

   每次你可以爬 1 或 2 个台阶。你有多少种不同的方法可以爬到楼顶呢?

   - 自顶向下（Top down）

```
def climbStairs1(self, n):
    if n == 1:
        return 1
    if n == 2:
```

```
            return 2
        return self.climbStairs(n-1)+self.climbStairs(n-2)
```

- 自底向上（Bottom up）（熟练 DP 形式）

```
    def climbStairs2(self, n):
        if n == 1:
            return 1
        res = [0 for i in xrange(n)]
        res[0], res[1] = 1, 2
        for i in xrange(2, n):
            res[i] = res[i-1] + res[i-2]
        return res[-1]
```

res[i] = res[i-1] + res[i-2] 类似 Fibonacci 数列 f(n) = f(n-1) + f(n-2)可以将一个 n 维数组简化为两个变量的空间复杂度形式：

```
    class Solution(object):
        def climbStairs(self, n):
            if n == 1:
                return 1
            a, b = 1, 2
            for i in xrange(2, n):
                a, b = b, a+b
        return b
```