\115. Distinct Subsequences

Hard

88345Add to ListShare

Given a string **S** and a string **T**, count the number of distinct subsequences of **S** which equals **T**.

A subsequence of a string is a new string which is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (ie, "ACE" is a subsequence of "ABCDE" while "AEC" is not).

**Example 1:**

Input: S = "rabbbit", T = "rabbit"  
Output: 3  
Explanation:  
  
As shown below, there are 3 ways you can generate "rabbit" from S.  
(The caret symbol ^ means the chosen letters)  
  
rabbbit  
^^^^ ^^  
rabbbit  
^^ ^^^^  
rabbbit  
^^^ ^^^

**Example 2:**

Input: S = "babgbag", T = "bag"  
Output: 5  
Explanation:  
  
As shown below, there are 5 ways you can generate "bag" from S.  
(The caret symbol ^ means the chosen letters)  
  
babgbag  
^^ ^  
babgbag  
^^ ^  
babgbag  
^ ^^  
babgbag  
 ^ ^^  
babgbag  
 ^^^

**Solution**

动态规划。定义dp数组，dp[i][j]表示S[1, … , i ]中，等于T[1,…,j]的字串的个数

计算dp[i+1][j+1]时，考察字符S[i+1]和字符T[j+1]

case1 S[i+1] == T[j+1]时，可以用S[1, … , i + 1]匹配T[1, … , j]或者用S[1, … , i]匹配T[1 , … , j]

case2 S[i+1] != T[j+1]时，维持原状，dp[i+1][j+1] = dp[i][j+1]

class Solution {  
public:  
 int numDistinct(string s, string t) {  
 long long int dp[s.size() + 1];  
 for(int i = 0; i <= s.size(); ++i)dp[i] = 1;  
 for(int i = 0; i < t.size(); ++i){  
 int tmp = dp[0];  
 dp[0] = 0;  
 for(int j = 0; j < s.size(); ++j){  
 int tmp2 = dp[j+1];  
 if(s[j] == t[i]){  
 dp[j+1] = dp[j] + tmp;  
 }else{  
 dp[j+1] = dp[j];  
 }  
 tmp = tmp2;  
 }  
 }  
 return dp[s.size()];  
 }  
};