给你一个整数数组 coins 表示不同面额的硬币，另给一个整数 amount 表示总金额。
请你计算并返回可以凑成总金额的硬币组合数。如果任何硬币组合都无法凑出总金额，返回 0 。
假设每一种面额的硬币有无限个。
题目数据保证结果符合 32 位带符号整数。

分析： 典型的0-1背包问题： 但与常规0-1背包问题不同，在常规0-1背包问题中计算最大价值其状态转移方程为： `dp[i][j] = std::max(dp[i-1][j], dp[i-1][j-weight[i-1]]+val[i])` 对本题的完全背包问题： **状态转移方程为： `dp[i][j] = dp[i-1][j] + dp[i][j-weight[i-1]]`** 因为其状态为i放入背包和不放入背包，但要求存在的组合总数，所以状态方程为放入和不放入之和。

```
class Solution {
public:
  int change(int amount, std::vector<int>& coins) {
    int                             len = coins.size();
    std::vector<std::vector<int>> dp =
        std::vector<std::vector<int>>(len + 1, std::vector<int>(amount +
1, 0));

    // base case
    // 当amount = 0时，都有一种方法
    for (int i = 0; i <= len; i++) {
      dp[i][0] = 1;
    }

    // 计算dp数组
    for (int i = 1; i <= len; i++) {
      for (int j = 1; j <= amount; j++) {
        if (j - coins[i - 1] < 0) {
          // 不可放入
          dp[i][j] = dp[i - 1][j];
        } else {
          dp[i][j] = dp[i - 1][j] + dp[i][j - coins[i - 1]];
        }
      }
    }

    return dp[len][amount];
  }
};
```