**★★★★☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：10313: Pay the price**

**解題者：李謙**

**解題日期：2023年5月24日**

**題意：**

使用面額1~300(皆為整數且可重複使用)的硬幣，依據指定數量湊出剛好的價值且最高不超過300，注意0元也被設定為可湊出，輸出應為0種方法。

 輸出為恰好湊出總額的方法數，輸出會根據輸入有所不同。

 輸入有三種情況:

1. t輸出組成t的方法數

2. t a輸出在使用不多於a個硬幣的情況下，組成t的方法數

3. t a b輸出在使用介於a, b的硬幣數量的情況下，組成t的方法數

**題意範例：**

Input : 6

Output: 11

Methods: 6, 5+1, 4+2, 3+3, 4+1+1, 3+2+1, 2+2+2, 3+1+1+1, 2+2+1+1,

 2+1+1+1+1, 1+1+1+1+1+1(共11種)

Input : 6 3

Output : 7

Methods: 6, 5+1, 4+2, 3+3, 4+1+1, 3+2+1, 2+2+2 (共7種)

**解法：**

此問題為knapsack problem的變異版，可用DP處理。

**解法範例：**

DP建表，從題意可發現，湊出6的方法數就是(6,1)…(6,5), (6,6)的方法數相加。DP處理的方式便是dp[i+j][k] += dp[j][k-1]

**討論：**

因為有time limit不可使用暴力解。(4星原因)

動態規劃可輕鬆解，須注意最高總額可達300，方法數會極大要用LL

**程式：**

#include<iostream>

#include<string>

#include<sstream>

using *namespace* std;

*long* *long* dp[301][1001] = {0};  //dp[i][j] 表示在i元最多用j個硬幣的方法數

*int* main()

{

    dp[0][0] = 1;

    for(*int* i=0; i<=300; ++i){              //i元的硬幣

        for(*int* j=0; j+i<=300; ++j){        //i+j元由j元的方法數而來

            for(*int* k=1; k<=1000; ++k){     //用了k個硬幣

                dp[i+j][k] += dp[j][k-1];   //用k個硬幣組成總和為j+i的方法數是由k-1組成j元的方法數而來

            }

        }

    }

    string str;

    *int* n[3];

    while(getline(cin, str)){

        stringstream ss(str);

        *int* i=0;

        while(ss >> n[i]) ++i;

        switch(i){

            case 1:

                printf("%lld\n", dp[n[0]][1000]);

                break;

            case 2:

                printf("%lld\n", dp[n[0]][n[1]]);

                break;

            case 3:

                if(n[1]==0)

                    printf("%lld\n", dp[n[0]][n[2]]);

                else

                    printf("%lld\n", dp[n[0]][n[2]] - dp[n[0]][n[1]-1]);

                break;

        }

    }

    return 0;

}