**★★☆☆☆**

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

**題號：11264: Coin Collector**

**解題者：鄭博元**

**解題日期：2024年5月2日**

**題意：**

給定不同面額的硬幣，可以提取任意金額，銀行將使用以下算法將錢給你。求一次提款最多可以拿到幾種不同面額的硬幣。

withdraw(X) {
    if(X == 0)
        return;
    令 Y 為其值不超過 X 且面額最大的硬幣。
    給客戶一個 Y 元的硬幣。
    withdraw(X-Y);
}

**題意範例：**

輸入:

2

6

1 2 4 8 16 32

6

1 3 6 8 15 20

輸入的第一行包含一個整數T，代表測資數量。
每組測資第一行有一個整數n (1 ≤ n ≤ 1000)，代表不同類型硬幣的數量。
下一行包含n個整數C1、C2、...、Cn，分別代表每種硬幣的面額。
其中C1 < C2 < C3 < ... < Cn < 1000000000，且C1 = 1。

輸出:

6

4

**解法：**

將硬幣面額由小到大開始取，如果已經取的硬幣總額大於等於目前看的硬幣價值，那就將取的硬幣中上一個取的挑掉，然後取目前正在看的，第一個硬幣一定要取(面額為1)。

**解法範例：**

討論給定1 3 6 8 15 20:

1一定要拿，然後看3。1<3，取3。接下來看6，1+3<6，取6。接下來看8，1+3+6>8，把6挑掉改取8。目前拿了1,3,8。接下來看15，1+3+8<15，取15。最後看20，1+3+8+15>20，挑掉15改取20。最後拿了1,3,8,20，共4種，輸出4。

**討論：**

(greedy)

以 1,3,6,8,15,20 為例

領 5 元時可拿到 3,1,1 => 比 5 小的硬幣都拿的到

領 10 元時可拿到 8,1,1 => 拿不到3,6

領 12 元時可拿到 8,3,1 => 拿不到6

領 14 元時可拿到 8,6 => 拿不到3,1

考慮一個問題:有可能同時拿到3,6嗎?

不可能，因為3+6>8，根據題目提供換錢的算法，會先取8直到不能取，也就是說取完8以後的餘額一定<8。

假如領9~14元(因為領15元會先取面額15的硬幣)，那麼根據演算法會先取1個8，剩1~6元，都不可能再取到一個3及一個6。

換句話說，如果 A+B<=C，那麼 A,B 硬幣沒有辦法同時出現，因為會被 C 取代。

因此對於此測資所有領錢情況都不可能同時拿到3跟6。

那麼要留3還是6?又或者是說有多組時呢?例如1+3+8+15>20時，要丟掉最大的還是最小的?

答:丟當前取的最大的。因為題目要求是最多能取幾種不同的硬幣。

為甚麼不是丟目前看的?

會違反題目給的找錢演算法，我們總是要給能給的中最大的。

如果丟掉次小的會發生甚麼事?(1一定會取

再參考這筆測資

1 3 6 8 13

假如丟掉的是3，那就會變成取了1+6+8>13，最後輸出是3。(1 6 8)

但假如丟掉6，那就會是1+3+8<13，最後可以輸出4。(1 3 8 13)

也就是說要盡量讓目前取的硬幣總額小一點，這樣才能多取幾個硬幣。

為甚麼題目給的算錢是從大到小，但解法是從小到大?

題目的由大到小是限制，即要求找出A+B<=C這個限制下的

依照題目從大到小算，最大的問題是你要提取多少?要找到一個值完全符合題目算法非常困難，更何況對於某些測資存在多組相同的組合都有正確解

例如一開始的1 3 6 8 15 20

至少存在

1 3 6 15

1 3 8 20

1 3 15 20 等多組解。

因此使用greedy，從小到大來確保總是符合A+B<=C

而從小到大也在上一頁中提到了能保證取的總額比較小，可以多取幾個

**程式：**

#include <iostream>

#include <vector>

using namespace std;

int count\_all(vector<int> l){

 int n=0;

 for(int a:l)

 n+=a;

 return n;

}

int main(int argc, const char \* argv[]) {

 int t;

 cin>>t;

 int num,temp;

 vector<int> l;

 while(t>0){

 cin>>num;

 for(int i=0;i<num;i++){

 cin>>temp;

 l.push\_back(temp);

 }

 vector<int> change;

 change.push\_back(l[0]);

 for(int i=1;i<num;i++){

 if(count\_all(change)>=l[i]){

 change.pop\_back();

 change.push\_back(l[i]);

 }

 else{

 change.push\_back(l[i]);

 }

 }

 cout<<change.size()<<endl;

 t--;

 change.clear();

 l.clear();

 }

 return 0;

}