1068. Find More Coins (30)

时间限制

150 ms

内存限制

65536 kB

代码长度限制

16000 B

判题程序

Standard

作者

CHEN, Yue

Eva loves to collect coins from all over the universe, including some other planets like Mars. One day she visited a universal shopping mall which could accept all kinds of coins as payments. However, there was a special requirement of the payment: for each bill, she must pay the exact amount. Since she has as many as 104 coins with her, she definitely needs your help. You are supposed to tell her, for any given amount of money, whether or not she can find some coins to pay for it.

**Input Specification:**

Each input file contains one test case. For each case, the first line contains 2 positive numbers: N (<=104, the total number of coins) and M(<=102, the amount of money Eva has to pay). The second line contains N face values of the coins, which are all positive numbers. All the numbers in a line are separated by a space.

**Output Specification:**

For each test case, print in one line the face values V1 <= V2 <= ... <= Vk such that V1 + V2 + ... + Vk = M. All the numbers must be separated by a space, and there must be no extra space at the end of the line. If such a solution is not unique, output the smallest sequence. If there is no solution, output "No Solution" instead.

Note: sequence {A[1], A[2], ...} is said to be "smaller" than sequence {B[1], B[2], ...} if there exists k >= 1 such that A[i]=B[i] for all i < k, and A[k] < B[k].

**Sample Input 1:**

8 9

5 9 8 7 2 3 4 1

**Sample Output 1:**

1 3 5

**Sample Input 2:**

4 8

7 2 4 3

**Sample Output 2:**

No Solution

Dp的写法。。

到现在为止的设想是先进行排序

然后对于每一位，进行查找，若这位与找到的位数可以得到0，则取零，进行插入判断，若不是，则MAX(dp[current],dp[search]-sequence[current];

[提交代码](https://www.patest.cn/contests/pat-a-practise/1068)

#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

#include<vector>

#include<sstream>

#include<stack>

#include<map>

#define MAX 505

using namespace std;

vector<int>sequence;

struct node {

int id;

int num;

int former;

int deep;

};

vector<node>dp;

vector<node>result;

void output()

{

if (result.size() == 0)

{

cout << "No Solution";

return;

}

stack<int>temp;

int j = result.back().id;

while (dp[j].former != -1)

{

temp.push(sequence[j]);

j = dp[j].former;

}

cout << sequence[j];

while (temp.size())

{

cout <<" "<< temp.top();

temp.pop();

}

return;

}

int main()

{

int num, sum;

cin >> num >> sum;

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

{

int temp;

scanf("%d", &temp);

sequence.push\_back(temp);

}

sort(sequence.begin(), sequence.end());

dp.resize(num);

dp[0].num = sum - sequence[0];

dp[0].former = -1;

dp[0].deep = 0;

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

{

dp[i].id = i;

dp[i].num = -1;

dp[i].former = -1;

dp[i].deep = 0;

for (int j = 0; j < i; j++)

{

if (dp[j].num == sequence[i])

{

dp[i].former = j;

dp[i].deep = dp[j].deep + 1;

if (result.size())

{

if (dp[i].deep >= result.back().deep)

result.push\_back(dp[i]);

}

else

result.push\_back(dp[i]);

break;

}

else if (i != num - 1)

{

if (dp[j].num - sequence[i] >= sequence[i + 1])

{

if (dp[j].num - sequence[i]>dp[i].num)

{

dp[i].deep = dp[j].deep + 1;

dp[i].num = dp[j].num - sequence[i];

dp[i].former = j;

}

}

}

}

}

output();

}