codeforces 148E report

题目

http://codeforces.com/problemset/problem/148/E

题目大意

一个公主生气了就要摔花瓶。花瓶是放在 n 个架子上，每个架子上每次只能取最左边或者最右边的花瓶来摔。每个花瓶有一个价值，公主要摔 m 个花瓶，问能毁掉的花瓶的最大价值是多少。

算法分析

可以想到不同架子上的花瓶摔掉的先后次序并不影响最后的价值，每个架子的最大毁坏的价值只和它摔了几个花瓶以及这些花瓶的摔的先后次序有关系。如果我们知道了每个架子上取不同个花瓶的最大毁坏价值，那么最终就是要计算一个方案安排每个架子上拿去毁掉的花瓶数，从而得出一个能得到最大毁坏价值的安排。

因此如果把架子 i 摔了 j 个花瓶的毁坏价值是 g[i , j]的话，那么就是要计算一个毁坏花瓶的方案使得价值最大。由于每次只能够取最左边和最右边的花瓶，所以可以将要拿取的部分看作是一段区间，可以设为 gt[l , r , i],意思就是[l,r]这段区间内取 i 个花瓶的最大毁坏价值。根据取花瓶的方式，对于一个架子，可以得出如下的状态转移方程：gt[l , r , i] = max(gt[l + 1 , r , i - 1] + w[l] , gt[l , r - 1 , i - 1] + w[r]);//w[i]为架子上第 i 个花瓶的价值

因此又有 g[i , j] = gt[1，number[i] , j];//number[i]为第 i 个架子的花瓶数

接下来对于所有的架子，可以设 f[i , j]是前 i 个架子上选择 j 个花瓶打掉的最大毁坏价值。这里容易想到一个类似背包模型的转移方程 f[i , j] = max(f[i - 1 , j - k] + g[i , k]);f[n，m]即是最终的答案。

参考代码

```
/*
 CF148E
 2012-02-17
 ACCEPTED
*/
#include<iostream>
#include<cstring>
#define MAXN 102
#define MAXM 10002
using namespace std;

int n , m ;
int f[ MAXN ][ MAXM ];
int g[ MAXN ][ MAXN ] , gt[ MAXN ][ MAXN ][ MAXN ];
int w[ MAXN ][ MAXN ] , v[ MAXN ];

void init()
{
 cin>>n>>m;
 for( int i = 1 ; i <= n ; ++ i )
  {
   cin>>v[ i ];
```

```
    for( int j = 1 ; j <= v[ i ] ; ++ j )
      cin>>w[ i ][ j ];
   }
 return ;
}

int max( int a , int b ) { return a > b ? a : b ; }

void dp1()
{
 int i , j , k , t ;

 for( k = 1 ; k <= n ; ++ k )
  {
   for( t = 1 ; t <= v[ k ] ; ++ t )
     {
      for( i = 1 ; i <= v[ k ] ; ++ i )
        {
         gt[ i ][ i ][ t ] = w[ k ][ i ];
         for( j = 1 ; i + j <= v[ k ] ; ++ j )
           {
            gt[ i ][ i + j ][ t ] = max( gt[ i + 1 ][ i + j ][ t - 1 ] + w[ k ][ i ] , gt[ i ][ i + j -
1 ][ t - 1 ] + w[ k ][ i + j ] );
            //g[ k ][ t ] = g[ k ][ t ] > gt[ i ][ i + j ][ t - 1 ] ? g[ k ][ t ] : gt[ i ][ i + j ][ t ];
           }
        }
      g[ k ][ t ] = gt[ 1 ][ v[ k ] ][ t ];
     }
  }
 //test
 /*
 for( i = 1 ; i <= n ; ++ i )
  {
   for( j = 1 ; j <= v[ i ] ; ++ j )
     cout<<g[ i ][ j ]<<" ";
   cout<<"\n";
  }
 */
 return ;
}

void dp2()
{
```

```
 int i , j , k ;
 memset( f , 0 , sizeof( f ) );

 for( i = 1 ; i <= n ; ++ i )
  {
   for( j = 1 ; j <= m ; ++ j )
     for( k = 0 ; k <= v[ i ] ; ++ k )
       if( j - k >= 0 and f[ i ][ j ] < f[ i - 1 ][ j - k ] + g[ i ][ k ] )
         f[ i ][ j ] = f[ i - 1 ][ j - k ] + g[ i ][ k ];
  }
 cout<<f[ n ][ m ]<<"\n";
 return ;
}

int main()
{

 init();

 dp1();

 dp2();

 return 0;
}
```