2012 Chengdu Regional Online E Food Solution

write by Gestalti Lur
2012-09-18
題目鏈接：http://acm.hdu.edu.cn/showproblem.php?pid=4292

題目大意

有 F 種食物，D 種饮料,给出每種食物和飲料的數量，給出 N 個人對於每種食物和飲料是否能够接受的列表。一份食物和一份飲料可以给一個人配餐，每个人只能接受他们能接受的食物和飲料，问最多可以给多少人配餐。N，F，D 不超过 200.

算法分析

可以用最大流建模。從源點向每種食物連一條流量爲食物數量的邊，從每種飲料出發向匯點連一條流量爲飲料數量的邊，再將每個人轉化成一條邊和兩個點(將第 i 個人轉化成兩個點 i 和 i'，他們之間連一條流量爲 1 的邊)，然後從第 i 個人可以接受的食物到點 i 連一條流量爲 1 的邊，從 i'出發向他能夠接受的飲料連一條流量爲 1 的邊，計算最大流即可。

ps:這種將點轉化成邊以便控制流量的方法在網絡流問題中經常可以見到。

參考代碼

dinic 實現：

```
/*
 2012 Chengdu Oline E
 gestapolur
 2012-09-17
 ACCEPTED
*/
#include<cstdio>
#include<cstring>
#define MAXV 832
#define MAXE 90002
#define INF 20000000

struct edge{//边
 short int u , v ;//e( u , v )
 int cap , flow , next; //残量网络的容量、流量、指向下一条从 u 出发的边的序号的指针(forward star 表示法)
};

short int n , vs , vt;
int m;
int cnt;
int head[ MAXV ];
edge ap[MAXE * 2];
```

```
short int h[ MAXV ], Q[ MAXE * 2 ];//Q 是用于构造分层图的一个队列
int G[ MAXV ];

//构造分层图
inline bool build(int s, int t){
  int p , x , y , front , rear;
  memset(h,255,sizeof(h));
  memset(G,0,sizeof(G));
  front = rear = h[ s ] = 0;
  Q[ rear ++ ] = s;
  G[ s ] = head[ s ];
  while( front < rear )
    {
     x = Q[ front ++ ];
     p = head[ x ];
     while( p )
     {
       y = ap[ p ].v;
       if( ap[ p ].cap and h[ y ] == -1 )
         {
           h[ y ] = h[ x ] + 1;
           G[ y ] = head[ y ];
           if( y == t ) return true;
           Q[ rear ++ ] = y;
         }
       p = ap[ p ].next;
     }
    }
  return false;
}

inline int min( int a , int b ){ return ( a < b ? a : b );}

//augumenting，dfs 增广过程 x 是当前节点，t 是汇点,low 是当前增广路上的流量。
int find(int x, int t, int low=0xffffffff)
{
  if(x==t)return low;
  int ret=0,y;
  //遍历所有与 x 相连的点 y，如果 y 在当前的分层图当中且仍然有剩余流量，则增广这条边并调整正向边
和反向边的剩余流量
  for(int &p=G[x];p;p=ap[p].next)
    {
     y = ap[ p ].v;
     if(ap[p].cap && h[y] == h[x]+1 && ( ret = find( y , t , min( low , ap[p].cap ) )
) )
     {
       ap[ p ].cap-=ret;
```

```
      ap[ (p - 1 ^ 1 ) + 1 ].cap+=ret;//p == odd ? p + 1 : p - 1;
      return ret;
     }
   }
 return 0;
}

inline void maxflow( int s, int t )
{
 int flow;
 cnt = 0;
 while( build(s, t) )
  while ( flow = find( s , t ) )
   cnt+=flow;
 return ;
}

inline void add_edge( int i , int u , int v , int c )
{
 ap[ i ].u = u;
 ap[ i ].v = v;
 ap[ i ].cap = c;
 ap[ i ].next = head[ u ];
 head[ u ] = i;
 return ;
}

bool init()
{
 int np , nf , nd;
 if( scanf("%d%d%d" , &np , &nf , &nd ) not_eq EOF )
  {
   int i , j , c;
   char str[ 202 ];
   n = nf + nd + np + np + 2;
   m = 0;
   vs = n - 1;
   vt = n;
   for( i = 1 ; i <= nf ; ++ i )
   {
    scanf( "%d" , &c );
    add_edge( ++ m , vs , i , c );
    add_edge( ++ m , i , vs , 0 );
   }
   for( i = 1 ; i <= nd ; ++ i )//
   {
    scanf( "%d" , &c );
    add_edge( ++ m , nf + np + np + i , vt , c );
```

```
      add_edge( ++ m , vt , nf + np + np + i , 0 );
    }
    for( i = 1 ; i <= np ; ++ i )
    {
      scanf( "%s" , str );
      for( j = 1 ; j <= nf ; ++ j )
        if( str[ j - 1 ] == 'Y' )
          {
          add_edge( ++ m , j , nf + i , 1 );
          add_edge( ++ m , nf + i , j , 0 );
          }
    }
    for( i = 1 ; i <= np ; ++ i )
    {
      add_edge( ++ m , nf + i , nf + np + i , 1 );
      add_edge( ++ m , nf + np + i , nf + i , 0 );
    }
    for( i = 1 ; i <= np ; ++ i )
    {
      scanf( "%s" , str );
      for( j = 1 ; j <= nd ; ++ j )
        if( str[ j - 1 ] == 'Y' )
          {
          add_edge( ++ m , nf + np + i , nf + np + np + j , 1 );
          add_edge( ++ m , nf + np + np + j , nf + np + i , 0 );
          }
    }
    return true;
   }
  return false;
}

int main()
{
  while ( init() )
    {
     maxflow( vs , vt );
     printf( "%d\n" , cnt );
     memset( head , 0 , sizeof( int ) * ( n + 1 ));
    }
  return 0;
}
```