# 网洛流

网络流dinic

bool bfs(){

int head=0,tail=1;

for(int i=0;i<=T;i++)h[i]=-1;

q[0]=0;h[0]=0;

while(head!=tail){

int now=q[head];head++;

for(int i=head[now];i;i=e[i].next)

if(e[i].v&&h[e[i].to]==-1){

h[e[i].to]=h[now]+1;

q[tail++]=e[i].to;

}

}

return h[T]!=-1;

}

int dfs(int x,int f){

if(x==T)return f;

int w,used=0;

for(int i=cur[x];i;i=e[i].next)

if(h[e[i].to]==h[x]+1){

w=f-used;

w=dfs(e[i].to,min(w,e[i].v));

e[i].v-=w;e[i^1].v+=w;

if(e[i].v)cur[x]=i;

used+=w;if(used==f)return f;

}

if(!used)h[x]=-1;

return used;

}

void dinic(){

while(bfs()){

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

cur[i]=head[i];

ans+=dfs(0,inf);

}

}

## POJ 1637

给出一张混合图（有有向边，也有无向边），判断是否存在欧拉回路。

首先是对图中的无向边随意定一个方向，然后统计每个点的入度（indeg）和出度（outdeg），如果（indeg - outdeg）是奇数的话，一定不存在欧拉回路；

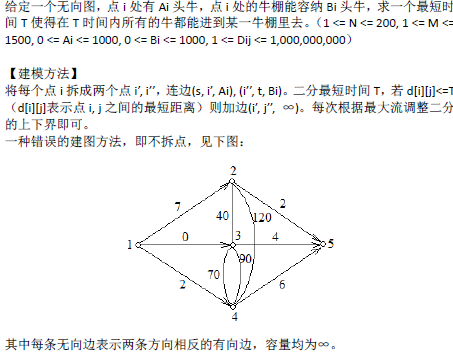
如果所有点的入度和出度之差都是偶数，那么就开始网络流构图：

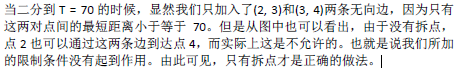
1，对于有向边，舍弃；对于无向边，就按照最开始指定的方向建权值为 1 的边；

2，对于入度小于出度的点，从源点连一条到它的边，权值为（outdeg - indeg）/2；出度小于入度的点，连一条它到汇点的权值为（indeg - outdeg）/2 的边；

构图完成，如果满流（求出的最大流值 == 和汇点所有连边的权值之和），那么存在欧拉回路，否则不存在。

## Poj2391





## POJ2699

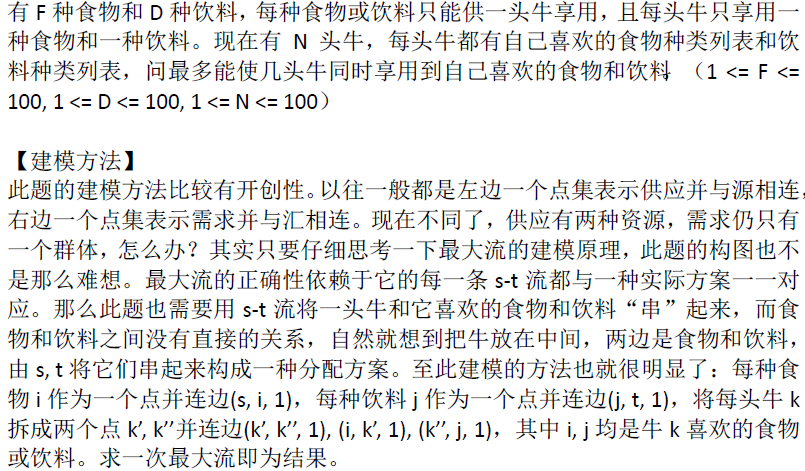
就是有n 个人，两两之间打比赛，每场比赛赢的人加一分，总共呢有n\*(n-1)/2个比赛，然后求这样一种人最多可能的个数，就是能赢所有比自己分高的人或者他就是分最高的人。

二进制位枚举strong kings，那么复杂度大概是1000\*60 \* 60

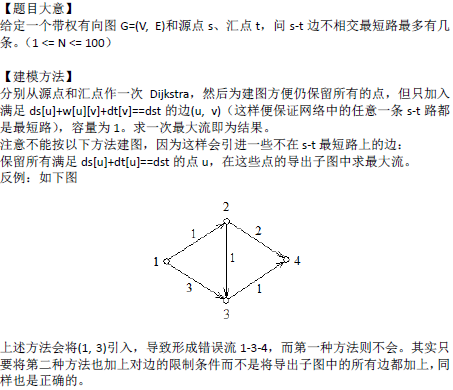
建图的话，就要分两种点，人和比赛，很显然，源点到所有的人建边，容量是该人的得分，所有比赛与汇点建边，容量为1。那么现在要处理的就是人与比赛的关系了

对每个strong kings，设其编号为i，某个比他分高的人为j，那么他俩对应的比赛应该是i赢，那么直接让i去连接这个比赛点就行了，容量为1，代表这是一个必要的边，但是其他的比赛我们不知道结果，就要把比赛双方i,j都连一个容量为1的边到对应的比赛点上.

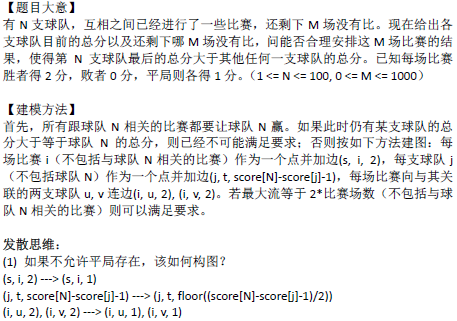
## Poj 3281



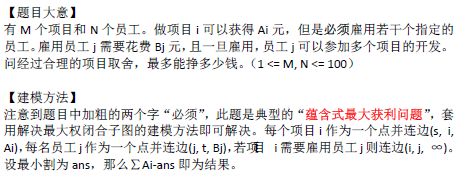
## Zoj2760



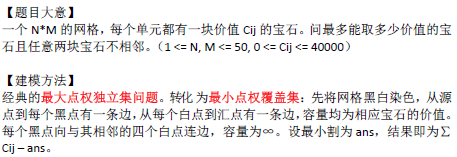
## WOJ1124



## HOj 2634



## HOj 2713

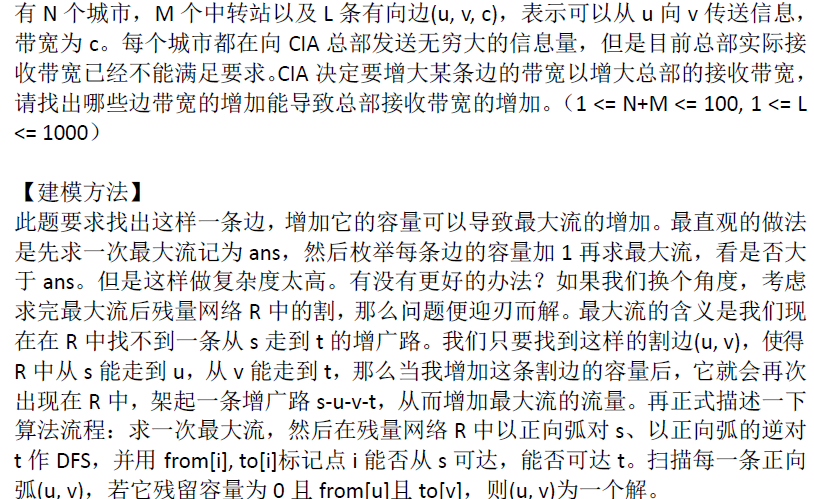


## POJ 1815

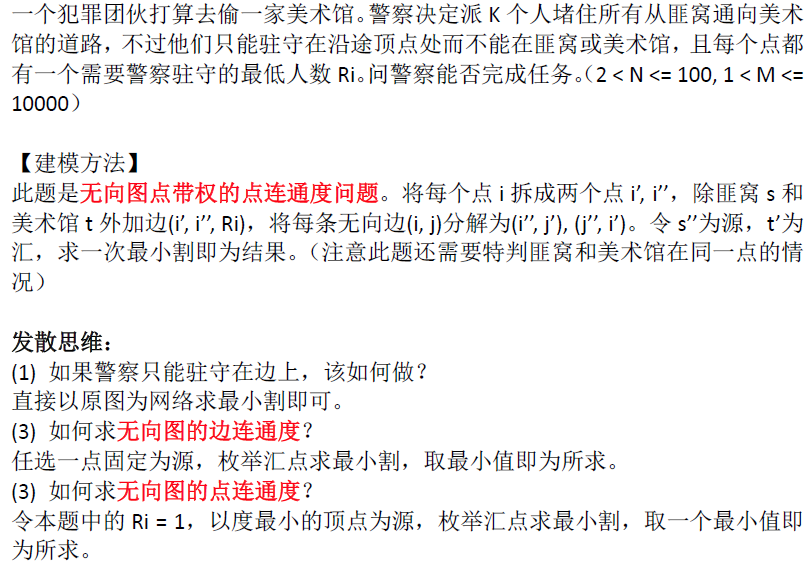
题意就是求s点到t点，最少去掉几个点使得他们不连通。

 因为最小割只能求割掉几条边的解，我们要求的是割掉几个点。那么我们可以这样考虑：把每个点拆成入点和出点。入点->出点权值为1。那么割掉这条边就相当于割掉这个点了，就能把这题转化成最小割。那么原来的边，我们是不希望去割它的，所以我们将原来的边的权值设置为INF。例如，原来边为u->v那么就变成out(u)->in(v)。这样跑一次最大流，跑出来的答案就是最小割掉的点数。

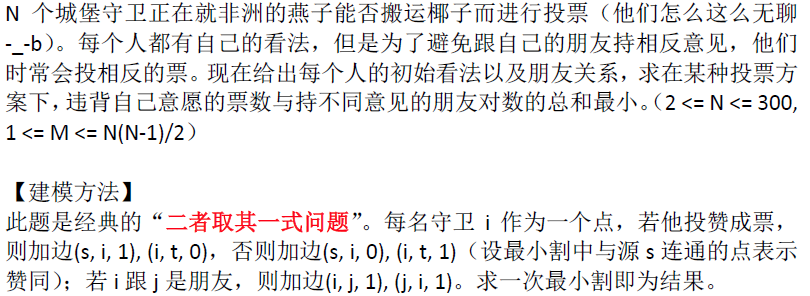
## Zoj2532



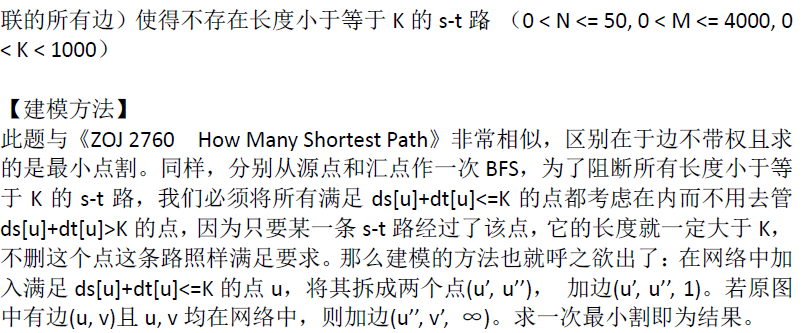
## Ural1277

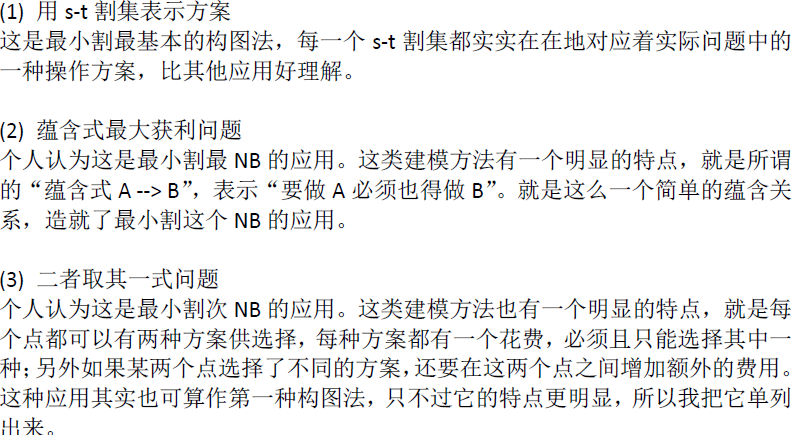


## Spoj 1693



## Acm2008

C:\Users\acm\AppData\Roaming\Tencent\Users\740103204\QQ\WinTemp\RichOle\)N8@_8PVZYPYR4NK6P5U}OE.png



# HDU 6118费用流

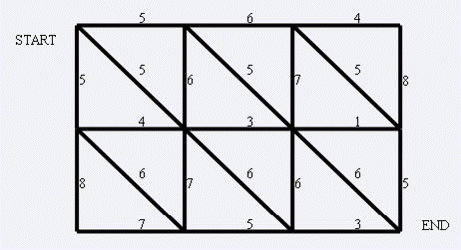
度度熊参与了喵哈哈村的商业大会，但是这次商业大会遇到了一个难题：  
喵哈哈村以及周围的村庄可以看做是一共由n个片区，m条公路组成的地区。第i个片区能够花费a[i]元生产1个商品，但是最多生产b[i]个。第i个片区也能够以c[i]的价格出售最多d[i]个物品。据测算，每一个商品运输1公里，将会花费1元。  
那么喵哈哈村最多能够实现多少盈利呢？

明显费用流，只不过将费用流终止条件改为费用为>=0，这样说明以后找到的增光路就都不赚钱了

## BZOJ1002狼爪兔子

平面网格图网络流点数非常多，求解最大流dinic已经不能解决了。

如求下网格图（1000\*1000大小）的最小割



<http://www.lydsy.com/JudgeOnline/problem.php?id=1001>

因为最大流==最小割

而平面图最小割可以通过其对偶图跑最短路求得

这个时候我们应该将它转化为对偶图

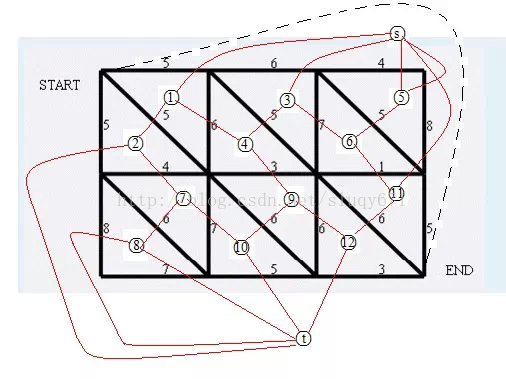
对偶图转化方法如下：

1起点和终点在外圈连一条无穷大的无向边，新得到一个s域和一个t域

2原图的域变为对偶图上一个点

3原图的边假设为a域和b域的公共边，则在对偶图连一条a到b的无向边，权值仍为原图的权值（注意s,t不连或者为无穷大）

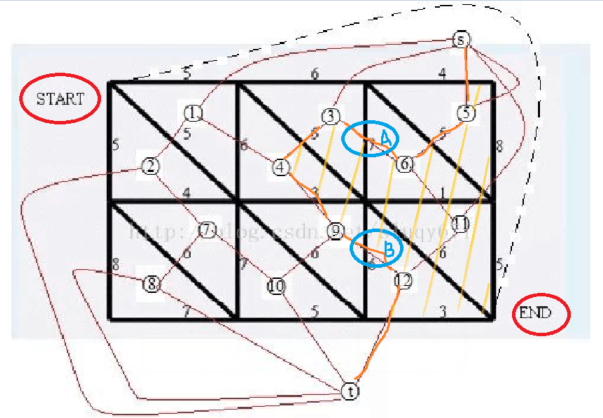
连好后如下



**这样每一条从s到t的路径都对应一个割**

**求最小割即为为最短路**

**而对于同一条路径上的两个割来说，如果他们是同一种割边（比如都是竖着的边（如下图蓝色圈圈）），而从s到t的路径通过这两条边的方向相反，则这这两条边在原图中的网络流流量方向一定相反。**



理解如下：假设我割掉橙色的边，那么黄色的区域是没有流量的，所以流量只能从外面流向黄色区域，所以A边流量从上往下，B边流量从下往上