



# 杂题选讲

算法设计与分析

# P1054 等价表达式

这个选择题中的每个表达式都满足下面的性质：

1. 表达式只可能包含一个变量 'a' 。
2. 表达式中出现的数都是正整数，而且都小于10000。
3. 表达式中可以包括四种运算  $+$ （加）， $-$ （减）， $*$ （乘）， $^$ （乘幂），以及小括号（，）。小括号的优先级最高，其次是 $^$ ，然后是 $*$ ，最后是 $+$ 和 $-$ 。 $+$ 和 $-$ 的优先级是相同的。相同优先级的运算从左到右进行。（注意：运算符 $+$ ， $-$ ， $*$ ， $^$ 以及小括号（，）都是英文字符）
4. 幂指数只可能是1到10之间的正整数（包括1和10）。
5. 表达式内部，头部或者尾部都可能有一些多余的空格。

下面是一些合理的表达式的例子：

$((a^1)^2)^3$ ， $a*a+a-a$ ， $((a+a))$ ， $9999+(a-a)*a$ ， $1+(a-1)^3$ ， $1^{10^9}$  .....

# P1054 等价表达式

## 输入输出样例

输入 #1

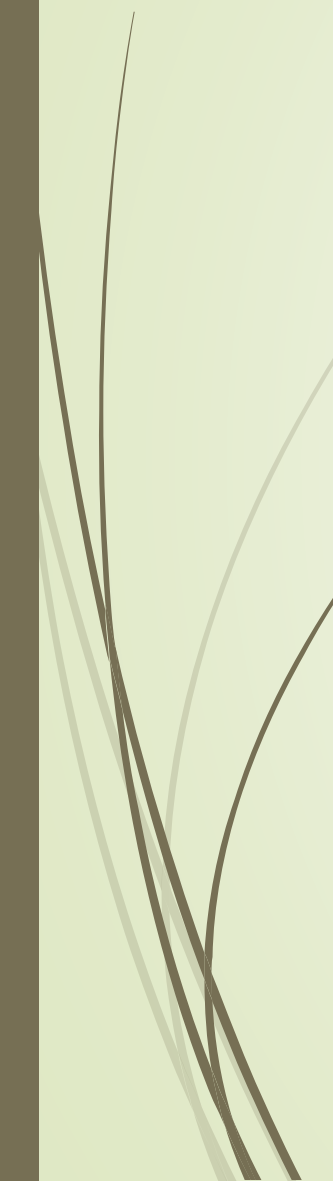

复制

```
( a + 1) ^2
3
(a-1)^2+4*a
a + 1+ a
a^2 + 2 * a * 1 + 1^2 + 10 -10 +a -a
```

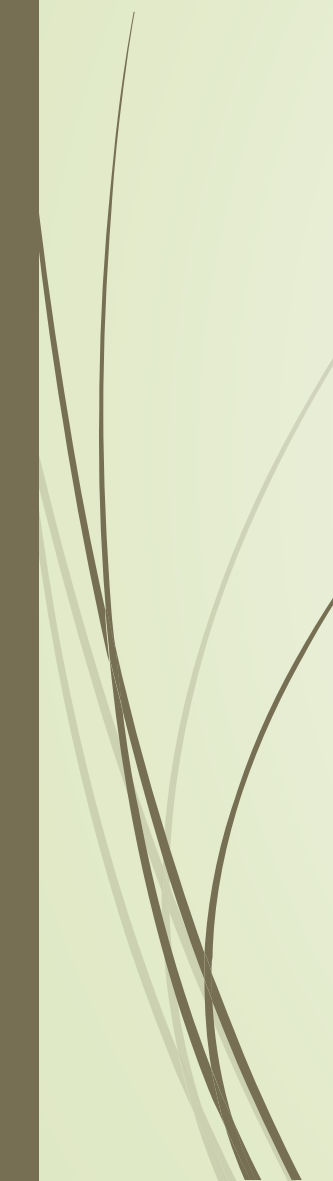

输出 #1

复制

AC





```
#include<bits/stdc++.h>
using namespace std;
const long long mod=1000000007;
//模幂运算
long long Pow(long long a,long long b)
{
    if(a==0)return 0;
    long long s=1;
    long long tt=a;
    while(b>0)
    {
        if(b%2==1)
            s=(s*tt)%mod;
        tt=(tt*tt)%mod;
        b/=2;
    }
    return s;
}
```



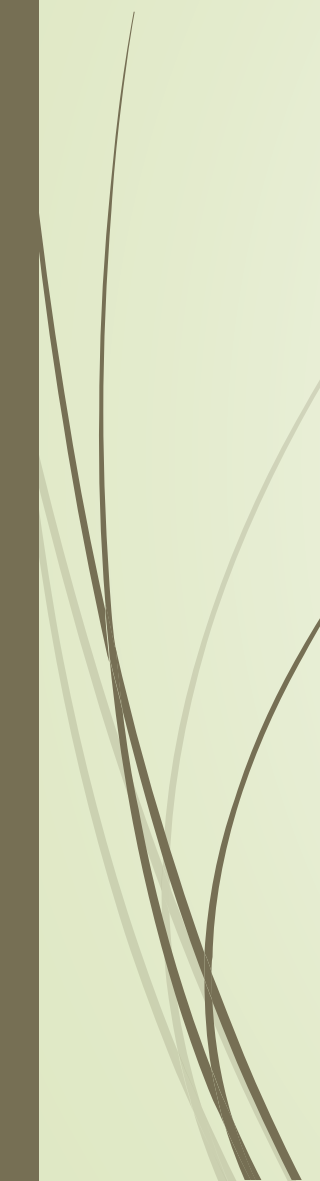

- unsigned int lev[256];//运算符优先级
- stack<long long>num;//数字
- stack<char>op;//运算符
- void calc()//从栈顶拿出两个元素，计算结果，然后再压入栈中
- {
- char ts=op.top();
- op.pop();
- long long t1=num.top();
- num.pop();
- long long t2=num.top();
- num.pop();
- long long ans=0;
- if(ts=='+')ans=t1+t2;
- if(ts=='-')ans=t2-t1+mod;
- if(ts=='\*')ans=t1\*t2;
- if(ts=='^')ans=Pow(t2,t1);
- num.push(ans%mod);
- }

```
➤ long long solve(char s[],long long xx)//计算
  一个表达式的值，假设变量取值为xx
➤ {
➤   int k=0;//保存(的个数
➤   op.push('#');//运算终止符
➤   long long temp=0;
➤   for(unsigned int i=0;s[i]!=0;i++)
➤   {
➤     //一次处理一个字符
➤     if(s[i]=='a')num.push(xx);
➤     else if(s[i]<='9'&&s[i]>='0') {
➤       temp=(temp*10+s[i]-'0')%mod;///
➤       if(s[i+1]<'0' || s[i+1]>'9') {
➤         num.push(temp);
➤         temp=0;
➤       }
➤     }
➤   }
➤   else if(s[i]=='('){op.push('(');k++;}
```

```
➤   else if(s[i]==')') {
➤     if(k<=0)return -1;//error!
➤     while(op.top()!='(')calc();
➤     op.pop();
➤     k--;
➤   } else {
➤     //运算符+-*^
➤     char tp=op.top();
➤     while(lev[(unsigned int)s[i)]<= lev[tp])
➤     {
➤       //栈顶优先级高，就先算栈顶
➤       calc();
➤       tp=op.top();
➤     }
➤     op.push(s[i]);
➤   }
➤ }
➤ while(op.top()!='#')calc();
➤ return num.top();
➤ }
```



```
char line[1024];
void getline()
{
    int i=0;
    char ch;
    while(1)
    {
        ch=getchar();
        if(ch==' ')continue;
        if(ch=='\r' || ch=='\n'){
            break;
        }
        line[i++]=ch;
    }
    while(ch!='\n')ch=getchar();
    line[i]=0;
}
```



```
int main(){
    lev[int('#')]=0;//定义优先级, 终止符保证继续输入
    lev[int('+')]=1;
    lev[int('-')]=1;
    lev[int('*')]=2;
    lev[int('^')]=3;
    long long a=137;
    getline();
    long long anss=solve(line,a);
    getline();
    int n;
    sscanf(line,"%d",&n);
    string lastans="";
    for(int i=1;i<=n;i++) {
        getline();
        if(solve(line,a)==anss)
            lastans+=char('A'+i-1);
    }
    cout<<lastans;
    return 0;
}
```



# 欧拉回路

(一笔画, 图中要有0或2个度为奇数的点)

## 题目描述

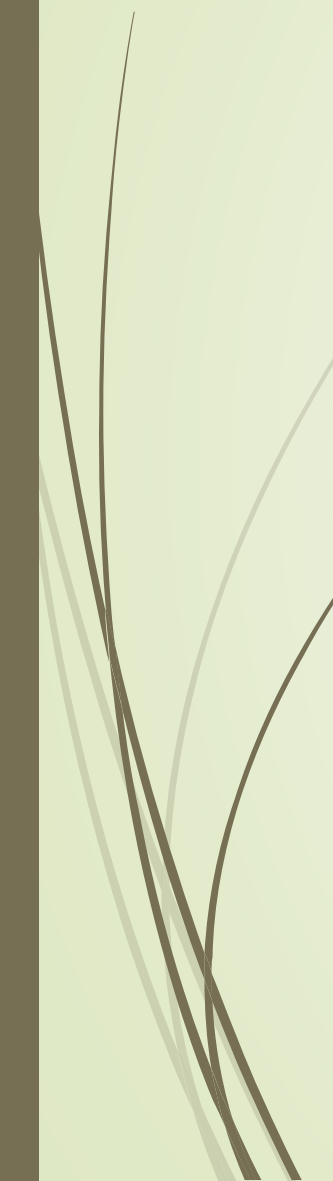

一个图能否一笔画成, 如果能请你每次都从当前最小编号的点开始画起, 如果不行, 则输出“no answer”

## 输入



```
4
0 1 0 0
1 0 1 1
0 1 0 1
0 1 1 0
```

## 输出

```
1 2 3 4 2
```



```
#include<iostream>
#include<cstdio>
using namespace std;
int n,tot,d[1001],fa[1001],bg,ed;
bool mp[1001][1001],vis[1001][1001];
int find(int x){return x==fa[x]?x:fa[x]=find(fa[x]);}
void dfs(int x)
{
    printf("%d ",x);
    for(int i=1;i<=n;i++)
        if(mp[x][i]&&!vis[x][i])
            {vis[x][i]=vis[i][x]=1;dfs(i);}
}
```



```
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)fa[i]=i;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
        {
            scanf("%d",&mp[i][j]);
            if(mp[i][j]==1)
            {
                d[i]++;//度
                int p=find(i),q=find(j);
                if(p!=q){fa[p]=q;tot++;}
            }
        }
}
```

```
//判断图是否连通
if(tot!=n-1){printf("no answer");return 0;}
for(int i=1;i<=n;i++)
    if(d[i]&1)
    {
        if(!bg)bg=i;//先找到开始节点
        else if(!ed)ed=i;//再找到结束节点
        else {printf("no answer");return 0;}
        //多于两个奇数度则不合法
    }
if(bg)dfs(bg);//如果找到奇数度
else dfs(1);//0个奇数度
return 0;
}
```

# (Code Forces) Graph and String

## 二分图染色问题

- 一个无向图，每个顶点取值为a,b,c,满足两个顶点相连当且仅当他们相等或者相邻（a,b相邻，b,c相邻），现给出所有边，判断图是否合法。
- 分析：如果一个顶点标号为b，那么它一定和所有顶点都是相连的；a和c一定是不相连的。
- 解题思路：首先找出b，剩下的二分

```

➤ #include <stdio.h>
➤ #include <algorithm>
➤ #define F(a,b,c) for(int a=b;a<=c;a++)
➤ #define N 505
➤ using namespace std;
➤ int n,m,a[N][N],l[N],u,v,ans,ok;
➤ char s[N];
➤ int main()
➤ {
➤     scanf("%d%d",&n,&m);
➤     F(i,1,m){
➤         scanf("%d%d",&u,&v);
➤         a[u][v]=a[v][u]=1;
➤         l[u]++;
➤         l[v]++;
➤     }
➤     F(i,1,n)
➤         if(l[i]==n-1) s[i]='b';

```

```

➤ int i=1;
➤     while(s[i]=='b'&&i<=n) i++;
➤     F(j,1,n)
➤         if(!a[i][j]) s[j]='c';
➤         else if(s[j]!='b') s[j]='a';
➤     s[i]='a';
➤     F(i,1,n&&ok!=-1)
➤         F(j,i+1,n)
➤             if(!a[i][j]&&(s[i]==s[j] || s[i]=='b')
➤                 || a[i][j]&&(s[i]=='a'&&s[j]=='c'))
➤                 ok=-1;
➤ if(ok===-1)
➤     printf("No\n");
➤ else {
➤     printf("Yes\n");
➤     F(j,1,n) printf("%c",s[j]);
➤ }
➤ return 0;
➤ }

```

# 2-sat

- 对于条件  $x_1=1$  或  $x_2=0$ 。
- 也就是说：
- 若  $x_1 \neq 1$ ，那么一定满足  $x_2=0$ 。
- 即  $x_1=0$  若为真，则  $x_2=0$  为真。 ———— (1)
- 若  $x_2 \neq 0$ ，那么一定满足  $x_1=1$ 。
- 即  $x_2=1$  若为真，则  $x_1=1$  为真。 ———— (2)
- 考虑如何把“若A为真，则B为真”的关系转化为边：
- 对于每个  $i$  ( $1 \leq i \leq n$ )，将  $x_i=0$  看作一个点，编号为  $i$ ， $x_i=1$  看作一个点，编号为  $i+n$ 。
- 然后对于 (1)，从  $1$  向  $2$  连边；对于 (2)，从  $n+2$  到  $n+1$  连边。
- 每个要求都连边一次，就把所有要求转化成一个图。

## 2-sat

- 然后会发现，如果点 $a$ 和点 $b$ 在同一个强连通分量里面，那么 $a$ 和 $b$ 的真假是相同的。
- 证明： $a$ 和 $b$ 在同一个强连通分量中，说明 $a$ 可以有一条路径通向 $b$ 。
- 若 $a$ 为真，则 $a$ 到 $b$ 路径上的所有点都为真， $b$ 也为真。
- 若 $a$ 为假，同理 $b$ 也为假。所以同一个强连通分量里所有点的真假性都相同。
- 因此，如果 $i$ 和 $i+n$ 在同一个强连通分量里，说明 $x_i=0$ 和 $x_i=1$ 的真假性相同，显然是不成立的，因此这时不存在解。反之则存在解。
- 如果存在解，考虑怎样找到一组解：发现拓扑序靠后的一定是有拓扑序靠前的推出来的。于是让拓扑序靠后的点为真即可。




## 2-sat

- 奶牛议会
- $M$ 只到场的奶牛 ( $1 \leq M \leq 4000$ ) 会给 $N$ 个议案投票( $1 \leq N \leq 1,000$ )。每只奶牛会对恰好两个议案  $B_i$  and  $C_i$  ( $1 \leq B_i \leq N$ ;  $1 \leq C_i \leq N$ )投出“是”或“否” (输入文件中的'Y'和'N')。他们的投票结果分别为 $VB_i$  ( $VB_i \in \{'Y', 'N'\}$ ) and  $VC_i$  ( $VC_i \in \{'Y', 'N'\}$ )。最后，议案会以如下的方式决定：每只奶牛投出的两票中至少有一票和最终结果相符合。例如Bessie给议案1投了赞成'Y'，给议案2投了反对'N'，那么在任何合法的议案通过 方案中，必须满足议案1必须是'Y'或者议案2必须是'N' (或者同时满足)。给出每只奶牛的投票，你的工作是确定哪些议案可以通过，哪些不能。
- 如果不存在这样一个方案， 输出” IMPOSSIBLE”。
- 如果至少有一个解，输出： Y 如果在每个解中，这个议案都必须通过； N 如果在每个解中，这个议案都必须驳回； ? 如果有的解这个议案可以通过，有的解中这个议案会被驳回。



## 2-sat

- ▶ 议案 1 通过（满足奶牛1, 3, 4） \* 议案 2 驳回（满足奶牛2） \* 议案 3 可以通过也可以驳回（这就是有两个解的原因） 事实上，上面的问题也只有两个解。所以，输出的答案如下： YN?
- ▶ 3 4  
1 Y 2 N  
1 N 2 N  
1 Y 3 Y  
1 Y 2 Y
- ▶ 4头奶牛给3个议案投票
- ▶ 第一头牛：同意1，反对2。。。
- ▶ 结果：YN ?



```
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
const int N=5000;
const int M=10000;
int
tot,point[N],v[M],nxt[M],low[N],df
n[N],nn,stack[N],top,id,belong[N]
,fin[N],n,m;bool vis[N],can1,can2;
void addline(int x,int y){++tot;
nxt[tot]=point[x]; point[x]=tot;
v[tot]=y;}
```

```
void tarjan(int x)
{
    low[x]=dfn[x]=++nn; vis[x]=1; stack[++top]=x;
    for (int i=point[x];i;nxt[i])
        if (!dfn[v[i]])
        {
            tarjan(v[i]); low[x]=min(low[x],low[v[i]]);
        }
        else if (vis[v[i]]) low[x]=min(low[x],dfn[v[i]]);
    if (low[x]==dfn[x])
    {
        int now=0;++id;
        while (now!=x)
        {
            now=stack[top--];
            vis[now]=0;
            belong[now]=id;
        }
    }
}
```

```

void dfs(int x,bool &can)
{
    if (fin[x+n]) {can=0;return;}//
    if (fin[x]) return;
    stack[++top]=x;fin[x]=1;//
    for (int i=point[x];i && can;i=nxt[i])
        dfs(v[i],can);
}

int main()
{
    scanf("%d%d",&n,&m);
    for (int i=1;i<=m;i++)//x Y x+n N
    {
        char s1[5],s2[5];int x=0,y=0,z,l;
        scanf("%d%s%d%s",&x,s1,&y,s2);
        if (s1[0]=='Y') z=0;else z=1;
        if (s2[0]=='Y') l=0;else l=1;
        addline(x+(!z)*n,y+l*n);
        addline(y+(!l)*n,x+z*n);
    }
}

```

```

for (int i=1;i<=2*n;i++)
    if (!dfn[i]) tarjan(i);
for (int i=1;i<=n;i++)
    if (belong[i]==belong[i+n])
        {printf("IMPOSSIBLE");return 0;}
for (int i=1;i<=n;i++)
{
    top=0;can1=1;dfs(i,can1);
    for (int j=1;j<=top;j++)
        fin[stack[j]]=fin[stack[j]+n]=0;
    top=0;can2=1;dfs(i+n,can2);
    for (int j=1;j<=top;j++)
        fin[stack[j]]=fin[stack[j]+n]=0;
    if (can1 && !can2) printf("Y");
    if (!can1 && can2) printf("N");
    if (can1 && can2) printf("?");
}
}

```



■ 题目分类查找：<http://hzwer.com/>

