第一题 word

题意简述:给出一个单词,统计其中出现最多的字母出现的次数 maxn,以及出现最少的字母的次数 minn,如果 maxn-minn 是质数的话则作为一个 Lucky Word..否则即为 No Answer.

类型: 模拟水题....

代码:

```
#include <stdio.h>
#include <string.h>


int count[27],max,min;
char s[200];
bool notprime[102];


int main()
{
	freopen("word.in","r",stdin);
	freopen("word.out","w",stdout);
	gets(s);
	memset(count,0,sizeof(count));
	for (int i=0;s[i];i++)
		count[s[i]-96]++;
	min = 2147483647;
	max = 0;
	for (int i=1;i<=26;i++)
	{
		if (!count[i])
			continue;
		if (max<count[i])
			max = count[i];
		if (min>count[i])
			min = count[i];
	}
```

```
        int j;
        notprime[0] = true;
        notprime[1] = true;
        for (int i=2;i<=50;i++)
        {
                j = i*2;
                while (j<=100)
                {
                        notprime[j] = true;
                        j += i;
                }
        }
        max -= min;
        if (notprime[max])
        {
                printf("No Answer\n");
                max = 0;
        }
        else
                printf("Lucky Word\n");
        printf("%d\n",max);
        return 0;
}
```

第二题 matches

题意简述: 给你 n(n<=24)根火柴棒,叫你拼出 "A + B = C"这样的等式,求方案数.

解题思路: 直接枚举 A 和 B(事实证明只到 3 位数),事先预处理 2000 以内各个数所用的火柴数.直接枚举出解

代码:

```
#include <stdio.h>
#include <string.h>


int n,ans;
int c[2002]={6,2,5,5,4,5,6,3,7,6};
```

```
int main()
{
	freopen("matches.in","r",stdin);
	freopen("matches.out","w",stdout);
	scanf("%d",&n);
	for (int i=10;i<=1800;i++)
		c[i] = c[i/10]+c[i%10];
	n -= 4;ans = 0;
	for (int i=0;i<=800;i++)
		for (int j=0;j<=800;j++)
			if (c[i]+c[j]+c[i+j]==n)
				ans++;
	printf("%d\n",ans);
	return 0;
}
```

第三题 message

题意简述: 给一个矩阵(左上角和右下角固定为 0),从左上角走两次到右下角,两次走的路径不能有交集(即一个点不能被走两次),求两次走过的格子上的数的和最大是多少.(类似二取方格数.)

解题思路: 二取方格数很经典的题目了,于是便直接以 f[i][j][k][p] 表示第一条路径走到(i,j),第二条路径走到(k,p)所取到的数的最大值..转移方程就很好办了..同时注意判断两条路不要从同一个点转移过来就好了.

```
#include <stdio.h>
#include <string.h>


int a[52][52],f[52][52][52][52],n,m,ni,nj,nk,np;
int next[4][4]={0,-1,-1,0,-1,0,-1,0,0,-1,0,-1,-1,0,0,-1};


int main()
```

```
{
	freopen("message.in","r",stdin);
	freopen("message.out","w",stdout);
	scanf("%d %d",&n,&m);
	memset(f,0,sizeof(f));
	for (int i=1;i<=n;i++)
		for (int j=1;j<=m;j++)
			scanf("%d",&a[i][j]);
	for (int i=1;i<=n;i++)
		for (int j=1;j<=m;j++)
			for (int k=1;k<=n;k++)
				for (int p=j;p<=m;p++)
					for (int m=0;m<=3;m++)
					{
						ni = i + next[m]
[0];
						nj = j + next[m]
[1];
						nk = k + next[m]
[2];
						np = p + next[m]
[3];
						if ((ni!=nk)||(nj!
=np))
							if (f[i][j]
[k][p] < f[ni][nj][nk][np] + a[i][j] + a[k][p])
								f[i]
[j][k][p] = f[ni][nj][nk][np] + a[i][j] + a[k][p];
					}
	printf("%d",f[n][m][n][m]);
	return 0;
}
```

第四题 twostack

以下引用 saybi 的题解 http://sqybi.com/blog/archives/78

这道题大概可以归结为如下题意:

有两个队列和两个栈,分别命名为队列 1(q1),队列 2(q2),栈 1(s1)和栈 2(s2).最初

的时候,q2,s1 和 s2 都为空,而 q1 中有 n 个数(n<=1000),为 1~n 的某个排列.

现在支持如下四种操作:

a 操作,将 q1 的首元素提取出并加入 s1 的栈顶.

b 操作,将 s1 的栈顶元素弹出并加入 ~~q1~~q2 的队列尾.

c 操作,将 q1 的首元素提取出并加入 s2 的栈顶.

d 操作,将 s2 的栈顶元素弹出并加入 ~~q1~~q2 的队列尾.

请判断,是否可以经过一系列操作之后,使得 q2 中依次存储着 1,2,3,…,n.如果可以,求出字典序最小的一个操作序列.

这道题的错误做法很多,错误做法却能得满分的也很多,这里就不多说了.直接切入正题,就是即将介绍的这个基于二分图的算法.

注意到并没有说基于二分图匹配,因为这个算法和二分图匹配无关.这个算法只是用到了给一个图着色成二分图.

第一步需要解决的问题是,判断是否有解.

考虑对于任意两个数 q1[i]和 q1[j]来说,它们不能压入同一个栈中的充要条件是什么(注意没有必要使它们同时存在于同一个栈中,只是压入了同 一个栈).实际上,这个条件 p 是:存在一个 k,使得 i<j<k 且 q1[k]<q1[i]<q1[j].

首先证明充分性,即如果满足条件 p,那么这两个数一定不能压入同一个栈.这个结论很显然,使用反证法可证.

假设这两个数压入了同一个栈,那么在压入 q1[k]的时候栈内情况如下:

…q1[i]…q1[j]…

因为 q1[k]比 q1[i]和 q1[j]都小,所以很显然,当 q1[k]没有被弹出的时候,另外两个数也都不能被弹出(否则 q2 中的数字顺序就不是 1,2,3,…,n 了).

而之后,无论其它的数字在什么时候被弹出,q1[j]总是会在 q1[i]之前弹出.而 q1[j]>q1[i],这显然是不正确的.

接下来证明必要性.也就是,如果两个数不可以压入同一个栈,那么它们一定满足条件 p.这里我们来证明它的逆否命题,也就是"如果不满足条件 p,那么这两个数一定可以压入同一个栈."

不满足条件 p 有两种情况:一种是对于任意 i<j<k 且 q1[i]<q1[j],q1[k]>q1[i];另一种是对于任意 i<j,q1[i]>q1[j].

第一种情况下,很显然,在 q1[k]被压入栈的时候,q1[i]已经被弹出栈.那么,q1[k]不会对 q1[j]产生任何影响(这里可能有点乱,因为看起来,当 q1[j]<q1[k]的时候,是会有影响的,但实际上,这还需要另一个数 r,满足 j<k<r 且 q1[r]<q1[j]<q1[k],也就是证明充分性的时候所说的情况…而事实上我们现在并不考虑这个 r,所以说 q1[k]对 q1[j]没有影响).="">

第二种情况下,我们可以发现这其实就是一个降序序列,所以所有数字都可以压入同一个栈.

这样,原命题的逆否命题得证,所以原命题得证.</q1[k]的时候,是会有影响的,但实际上,这还需要另一个数 r,满足 j<k<r 且>

此时,条件 p 为 q1[i]和 q1[j]不能压入同一个栈的充要条件也得证.

这样,我们对所有的数对(i,j)满足 1<=i<j<=n,检查是否存在 i<j<k 满足 p1[k]< p1[i]

二分图的两部分看作两个栈,因为二分图的同一部分内不会出现任何连边,也就相当于不能压入同一个栈的所有结点都分到了两个栈中.
此时我们只考虑检查是否有解,所以只要 O(n)检查出这个图是不是二分图,就可以得知是否有解.

此时,检查有解的问题已经解决.接下来的问题是,如何找到字典序最小的解.

实际上,可以发现,如果把二分图染成 1 和 2 两种颜色,那么结点染色为 1 对应当前结点被压入 s1,为 2 对应被压入 s2.为了字典序尽量小,我们希望让编号小的结点优先压入 s1.

又发现二分图的不同连通分量之间的染色是互不影响的,所以可以每次选取一个未染色的编号最小的结点,将它染色为 1 并从它开始 DFS 染色,直到所有结点都被染色为止.这样,我们就得到了每个结点应该压入哪个栈中.接下来要做的,只不过是模拟之后输出序列啦~

还有一点小问题,就是如果对于数对(i,j),都去枚举检查是否存在 k 使得 p1[k]

MRain:程序是我自己写的(懒得按照格式输出了),已经过了所有标准数据.

```
var    a,b,head,next,point,color:array[0..2001]of longint;
       s:array[1..2,0..1000]of longint;
       n,p,i,j,last:longint;
procedure badend;
begin
       writeln(0);
       close(output);
       halt;
end;
procedure addedge(a,b:longint);
var    t:longint;
begin
       inc(p);
       point[p]:=b;
       if head[a]=0 then
```

```
                head[a]:=p
        else
        begin
                t:=head[a];
                while next[t]<>0 do
                        t:=next[t];
                next[t]:=p;
        end;
end;
procedure dfs(now:longint);
var     t:longint;
begin
        t:=head[now];
        while t<>0 do
        begin
                if color[point[t]]=0 then
                begin
                        color[point[t]]:=3-color[now];
                        dfs(point[t]);
                end;
                if color[point[t]]=color[now] then badend;
                t:=next[t];
        end;
end;
begin
        assign(input,'twostack.in');
        reset(input);
        assign(output,'twostack.out');
        rewrite(output);
        fillchar(s,sizeof(s),0);
        fillchar(a,sizeof(a),0);
        readln(n);
        for i:=1 to n do
                read(a[i]);


        b[n+1]:=maxlongint;p:=0;
        for i:=n downto 1 do
        begin
                b[i]:=b[i+1];
                if a[i]<b[i] then b[i]:=a[i];
        end;
        for i:=1 to n do
                for j:=i+1 to n do
```

```
                if (b[j+1]<a[i])and(a[i]<a[j]) then
                begin
                        addedge(i,j);
                        addedge(j,i);
                end;
        for i:=1 to n do
                if color[i]=0 then
                begin
                        color[i]:=1;
                        dfs(i);
                end;
        last:=1;
        for i:=1 to n do
        begin
                if color[i]=1 then
                        write('a ')
                else
                        write('c ');
                inc(s[color[i],0]);
                s[color[i],s[color[i],0]]:=a[i];
                while (s[1,s[1,0]]=last)or(s[2,s[2,0]]=last)
do
                begin
                        if s[1,s[1,0]]=last then
                        begin
                                write('b ');
                                dec(s[1,0]);
                        end;
                        if s[2,s[2,0]]=last then
                        begin
                                write('d ');
                                dec(s[2,0]);
                        end;
                        inc(last);
                end;
        end;
        close(input);
        close(output);
end.
```