第五次练习赛题解

A. （版像镜）数质个来个质数（镜像版）

解题思路

这道题的题意非常清晰：输出a和b之间所有大于等于5的**回文质数**。很多同学用的方法是本地打表，这样对于做题来说固然没有问题，但是为了锻炼思维，还是建议能发散思维想想不打表的方法。

打表的方法就不说了，不打表的方法大致思路是这样的，考虑到1e8以内的回文数非常少（数量少于2e4），所以可以想办法构造回文数，然后再判断是否是素数，这样结合$\sqrt{n}$判断素数，复杂度就足够了。构造回文数的方法也有很多，比较方便的是枚举前半段，然后直接对称即可。

AC代码

```
#include <stdio.h>
int ans[1000005],tot,l,r,ten[10];

int is_prime(int x)      //判断质数
{
    if(x==1) return 0;
    for(int i=2;i*i<=x;i++)
        if(x%i==0) return 0;
    return 1;
}

int cmp(const void *x,const void *y)
{
    return *(int *)x<*(int *)y?-1:1;
}

int main()
{
    ten[0]=1;
    for(int i=1;i<=8;i++) ten[i]=ten[i-1]*10;
    for(int i=1;i<10000;i++)    //枚举前半段，构造回文数
    {
        int x[6],t=0,q=i,p=0;
        while(q) x[t++]=q%10,q/=10;
        for(int j=0;j<t;j++) p=p*10+x[j];
        int s1=i*ten[t]+p,s2=i*ten[t-1]+p-x[0]*ten[t-1];    //分别对应奇偶两种
        if(is_prime(s1)) ans[tot++]=s1;
        if(is_prime(s2)) ans[tot++]=s2;
    }
    qsort(ans,tot,sizeof(int),cmp);
    scanf("%d%d",&l,&r);
    for(int i=2;i<tot;i++)
        if(ans[i]>=l && ans[i]<=r)
            printf("%d\n",ans[i]);

    return 0;
}
```

那么，我们不枚举回文数可不可以做呢？当然可以！这个要涉及**线性筛素数**的方法，不要求掌握。注意到只要是大于两位数的偶回文数，一定是11的倍数，所以8位数的所有数都不用考虑，我们实际上只用筛出1e7以内的所有素数，然后判断是否是回文数即可。复杂度大概是O(1e7)，本题勉强可过。

AC代码2

```
#include <stdio.h>
int isprime[10000005],a,b;

int ispa(int x)
{
    int c[10],i=0;
    while(x) c[++i]=x%10,x/=10;
    for(int j=1;j<=i/2;j++)
        if(c[j]!=c[i-j+1]) return 0;
    return 1;
}

int main()
{
    for(int i=2;i<=10000000;i++) isprime[i]=1;
    for(int i=2;i*i<=10000000;i++)
        if(isprime[i])
            for(int j=i*i;j<=10000000;j+=i) isprime[j]=0;
    scanf("%d%d",&a,&b);
    if(b>10000000) b=10000000;
    for(int i=a;i<=b;i++)
        if(i>=5 && isprime[i] && ispa(i))
            printf("%d\n",i);

    return 0;
}
```

B. lx买东西

解题思路

暴力枚举每一种硬币出现的次数即可通过本题。

AC代码

```
#include<stdio.h>
int main(){
    int i,j,k,l,m,o,p,q,n,ans=0;
    scanf("%d",&n);
    for(i=0;i<=n;i+=128){
        for(j=0;i+j<=n;j+=64){
            for(k=0;i+j+k<=n;k+=32){
                for(l=0;i+j+k+l<=n;l+=16){
                    for(m=0;i+j+k+l+m<=n;m+=8){
                        for(o=0;i+j+k+l+m+o<=n;o+=4){
                            for(p=0;i+j+k+l+m+o+p<=n;p+=2){
                                for(q=0;i+j+k+l+m+o+p+q<=n;q++){
                                    if(i+j+k+l+m+o+p+q==n)ans++;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    printf("%d",ans);
    return 0;
}
```

当然，可以用动态规划的思想，将二进制分解做出此题。有兴趣的同学可以搜索"划分数"。

AC代码2

```
#include<stdio.h>
int f[210]={1,1};
int main(){
    int i,n;
    scanf("%d",&n);
    for(i=2;i<=n;i++)f[i]=f[i-2]+f[i>>1];
    printf("%d",f[n]);
    return 0;
}
```

C. 维修矩阵

解题思路

本题目旨在考查对于数组的应用。题目本身难度不大，但要处理的变量较多，尤其是在刚学二维数组时容易产生混乱，常有同学行列用反，数据读入和处理方向相反，而导致结果错误。

另外标准代码中定义了函数*swap*，用以交换两个数据，这个函数在主函数中多次调用，鲜明的表现出了函数对于减少代码量和方便阅读的优势。

标准代码中在二维数组定义时略大于数据范围，这是为防止可能会有的一些操作导致数组越界，同学们要在自己的代码中注意这个问题，在定义数组时要略大于最大范围。

AC代码

```
#include <stdio.h>
int n, m, q;
int A[1110][1110];
int x[10], y[10], c[10];

void swap(int *a, int *b) {
    int t = *a;
    *a = *b;
    *b = t;
}

int main()
{
    int i, j;
    scanf("%d%d%d", &n, &m, &q);
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= m; ++j)
            scanf("%d", &A[i][j]);
    int tp, pos, l, r;
    while (q--) {
        scanf("%d", &tp);
        if (tp == 3) {
            for (i = 1; i <= 3; ++i) {
                scanf("%d%d", &x[i], &y[i]);
                c[i] = A[x[i]][y[i]];
            }
            for (i = 1; i < 3; ++i)
                for (j = i + 1; j <= 3; ++j)
                    if (c[i] > c[j])
                        swap(&c[i], &c[j]);
            for (i = 1; i <= 3; ++i)
                A[x[i]][y[i]] = c[i];
        } else {
            scanf("%d%d%d", &pos, &l, &r);
            if (tp == 1) {
                while (l < r) {
                    swap(&A[pos][l], &A[pos][r]);
                    ++l; --r;
                }
```

```
            } else {
                while (l < r) {
                    swap(&A[l][pos], &A[r][pos]);
                    ++l; --r;
                }
            }
        }
    }
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= m; ++j)
            printf("%d ", A[i][j]);
        printf("\n");
    }
    return 0;
}
```

D. 小糖与小光的知识问答(2)

解题思路

本题考查二维数组的使用。用一个二维数组模拟比赛过程，最后分别统计占领结果即可。这题要特别注意某个位置**明确归属**的含义，很多同学另外开了一个标记数组来记录某个位置是否被报过数，但是这里报数不等同于占领，所以要明确理解了题意之后再思考如何解答。

另外用一维数组实现二维数组也是可以的，对于这种方法我们可以通过定义函数或者宏定义的方式来建立二维和一维的映射，这样可以简化代码，同时使代码更加清晰易懂。

AC代码

```
#include <stdio.h>
char s[105][105];
int n,m,x,y,suml,sumt;

void change(char c)
{
    scanf("%d%d",&x,&y);
    if(!s[x][y])
        for(int i=1;i<=n;i++)
            s[x][i]=s[i][y]=c;
}

int main()
{
    scanf("%d%d",&n,&m);
    for(int k=1;k<=m;k++)
    {
        change('l');
        change('t');
        printf("Turn #%d\n",k);
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=n;j++)
                printf("%c ",s[i][j]?s[i][j]:'0');
            puts("");
        }
    }
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            suml+=s[i][j]=='l',sumt+=s[i][j]=='t';
    printf("%d:%d",sumt,suml);

    return 0;
}
```

E. 搜索引擎

解题思路

这道题只需要按照题目要求一步一步进行操作即可，可以使用库函数 $isalnum()$ 、 $isdigit()$ 、 $isalpha()$ 判断字符的类型，使用库函数 $strcmp()$ 判断字符串是否和排除关键字完全相同。

主要的问题包括：

1、看到匹配的括号只想到了小括号

2、和排除关键字完全相同的内容才被忽略，即“110”仅和“110”完全相同，与“11”、“1100”都不算完全相同。

AC代码

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
int main()
{
    int n,i,j,k,isnokey=0,flag=0;
    char c[51050]={'\0'},keyword[10005]={'\0'},nokeyword[10005]={'\0'};
    fgets(c,51050,stdin);
    scanf("%d",&n);
    for(i=0;i<strlen(c)&&i<n;i++)//先遍历字符串确认有没有排除关键字
    {
        if(c[i]=='-')
        {
            isnokey=1;
            for(j=0,k=0,i++;(isdigit(c[i+j])||isalpha(c[i+j])||c[i+j]
<32||c[i+j]==127)&&i+j<strlen(c)&&i+j<n;j++)//是否是数字、字母，可合并为函数
isalnum()。0-31,127为不可见字符。
            {
                if(c[i+j]<32||c[i+j]==127) continue;
                nokeyword[k]=c[i+j];
                k++;
            }
            break;
        }
    }
    for(i=0;i<strlen(c)&&i<n;i++)
    {
        if(isdigit(c[i])||isalpha(c[i]))//按题目要求分割字符串，如果有排除关键字判断是否
需要排除
        {
            for(j=0,k=0;(isdigit(c[i+j])||isalpha(c[i+j])||c[i+j]
<32||c[i+j]==127)&&i+j<strlen(c)&&j+i<n;j++)
            {
                if(c[i+j]<32||c[i+j]==127) continue;
                keyword[k]=c[i+j];
                k++;
            }
            keyword[k]='\0';
            i+=j;
```

```
            if(isnokey==1&&strcmp(keyword,nokeyword)!=0) printf("%s\n",keyword);
            else if(isnokey==0) printf("%s\n",keyword);
        }
        if(c[i]=='(')//遇到括号则寻找有没有匹配的括号，并输出括号中所有的内容，请注意括号中的内容也有可能恰好是排除关键字。
        {
            for(j=i;j<strlen(c)&&j<n;j++)
            {
                if(c[j]==')')
                {
                    flag=1;
                    break;
                }
            }
            if(flag==1)
            {
                for(i++,k=0;i<j;i++)
                {
                    if(c[i]<32||c[i]==127) continue;
                    keyword[k]=c[i];
                    k++;
                }
                keyword[k]='\0';
                if(isnokey==1&&strcmp(keyword,nokeyword)!=0) printf("%s\n",keyword);
                else if(isnokey==0) printf("%s\n",keyword);
                flag=0;
            }
        }
        else if(c[i]=='[')
        {
            for(j=i;j<strlen(c)&&j<n;j++)
            {
                if(c[j]==']')
                {
                    flag=1;
                    break;
                }
            }
            if(flag==1)
            {
                for(i++,k=0;i<j;i++)
                {
                    if(c[i]<32||c[i]==127) continue;
                    keyword[k]=c[i];
                    k++;
                }
                keyword[k]='\0';
                if(isnokey==1&&strcmp(keyword,nokeyword)!=0) printf("%s\n",keyword);
                else if(isnokey==0) printf("%s\n",keyword);
                flag=0;
            }
        }
        else if(c[i]=='{')
        {
            for(j=i;j<strlen(c)&&j<n;j++)
            {
```

```
                if(c[j]=='}')
                {
                    flag=1;
                    break;
                }
            }
            if(flag==1)
            {
                for(i++,k=0;i<j;i++)
                {
                    if(c[i]<32||c[i]==127) continue;
                    keyword[k]=c[i];
                    k++;
                }
                keyword[k]='\0';
                if(isnokey==1&&strcmp(keyword,nokeyword)!=0)
printf("%s\n",keyword);
                else if(isnokey==0) printf("%s\n",keyword);
                flag=0;
            }
        }
    }
    return 0;
}
```

AC代码2

```
/*
 Author: 高世伟(35686)
 Result: AC Submission_id: 2042296
 Created at: Mon Nov 11 2019 12:47:59 GMT+0800 (CST)
 Problem: 2667  Time: 12    Memory: 1768
*/

#include<stdio.h>
#include<math.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
#define swap(a,b) ({int _temp=a;a=b;b=_temp;})
const double eps =1e-8;
int count=0;
char b[100000],temp[100000],a[100000],out[100000];
int x[100000];
int num;
int main()
{
    int n;
    gets(b);
    scanf("%d",&n);
    strncpy(a,b,n);
    int len=strlen(a);
    int l1=-1,l2=-1,l3=-1;
    for(int i=0;i<len;i++)
    {
        if (a[i]<=31||a[i]==127)continue;
```

```
        if(a[i]=='(')
        l1=i;
        if(a[i]==')'&&l1!=-1)
        {
            x[l1]=1,x[i]=2;
            l1=-1;
        }
        if(a[i]=='[')
        l2=i;
        if(a[i]==']'&&l2!=-1)
        {
            x[l2]=1,x[i]=2;
            l2=-1;
        }
        if(a[i]=='{')
        l3=i;
        if(a[i]=='}'&&l3!=-1)
        {
            x[l3]=1,x[i]=2;
            l3=-1;
        }
    }
    int in=0,u=0;
    for(int i=0;i<len;i++)
    {
        if (a[i]<=31||a[i]==127)continue;
        if (a[i]=='-')
        {
            i++;
            while(isalnum(a[i]))
            {
                out[count++]=a[i];
                i++;
            }
        }
    }
    out[count]=0;
    int ans=0;
    for(int j=0;j<len;j++)
    {
        if (a[j]<=31||a[j]==127)continue;
        if(x[j-1]==2)in=0;if(x[j]==1)in=1;
        if(in){
            if(x[j]==1)
            {
                    temp[ans]=0;
                    if(strcmp(temp,out)!=0&&ans!=0)
                    printf("%s\n",temp);
                    ans=0;
            }
            if(x[j]!=1&&x[j]!=2)
            temp[ans++]=a[j];
            if(x[j]==2)
            {
                if(ans>0)
                {
                    temp[ans]=0;
                    printf("%s\n",temp);
```

```
                    ans=0;
                }
            }
        }
        else if(isalnum(a[j])){
            temp[ans++]=a[j];
        }
        else if(ans){
            temp[ans]=0;
            if(strcmp(temp,out)!=0)
                printf("%s\n",temp);
            ans=0;
        }
    }
    if(ans){
        temp[ans]=0;
        if(strcmp(temp,out)!=0)
            printf("%s\n",temp);
        ans=0;
    }
    return 0;
}
```

F. 投喂cwd-1

解题思路

对于每一次加入饼干的操作，通过下列步骤模拟：

1. 插入一个新饼干；
2. 从头到尾检查，有没有连续的x个饼干：如果有，删掉它，回到2开头；否则进行下一次操作；
3. 检查到结尾，本次操作结束，进行下一次操作。

AC代码

```
#include<stdio.h>
#include<string.h>
char a[2010];
int main(){
    int i,j,n,x,len,ans=0;
    scanf("%d%d%s",&n,&x,a);
    len=strlen(a);
    while(n--){
        char ch;int p;
        scanf("%d %c",&p,&ch);
        len++;
        if(p>len-1)p=len-1;
        for(i=len;i>=p;i--)a[i+1]=a[i];
        a[p]=ch;
        for(i=0;i<len;i++){
            j=0;
            while(i+j<len&&j<x&&a[i+j]==a[i])j++;
            if(j==x){
                len-=x;
                ans+=x;
                for(j=i;j<len;j++)a[j]=a[j+x];
                i=-1;
            }
        }
    }
    printf("%d",ans);
    return 0;
}
```

G. 我完全懂了

解题思路

本题旨在考查对快速排序和二分法的使用。相信不少同学在第一眼看到这题时选择了直接用数组存下了两次读入的内容直接进比较（比较部分如下）

```
for(int i=1;i<=n;i++){
    for(int j=1;j<=m;j++)if(a[i]==b[j])
    {
        printf("%d ",a[i]);
        break;
    }
}
```

但是结果会是`TLE`。这是因为，如果我们假定运行一个语句的时间是1，这个两个语句运行的时间会是nm；而如果我们使用`C`语言内置函数`qsort`先进行排序，再使用二分法进行查找，我们花费的时间会是$2m\log_2 m+n\log_2 m$。

当数据很大时，我们可以很明显的看出，正确方法的时间会比直接查找快很多，当你们学习了数据结构之后会了解什么是时间复杂度，你们会理解的更加深刻。

而了解了上面两种情况的区别后，我们解题方法就很容易理解为什么这样选择了。首先我们需要利用`qsort`函数对两个数组进行排序，之后先选定一个数组，将其中每一个元素分别在另一个数组中进行二分查找，这样我们就可以得到结果并记录了。

AC代码

```
#include<stdio.h>
#include<stdlib.h>
int n,m,a[100005],b[100005];
int cmp(const void *x,const void *y){return *(int*)x-*(int*)y;}
int main()
{
    while(~scanf("%d%d",&n,&m))
    {
        for(int i=1;i<=n;i++)scanf("%d",&a[i]);
        for(int i=1;i<=m;i++)scanf("%d",&b[i]);
        qsort(b+1,m,sizeof(b[0]),cmp);
        for(int i=1;i<=n;i++)
        {
            int l=1,r=m;
            while(l<=r)
            {
                int mid=(l+r)>>1;
                if(b[mid]>a[i]) r=mid-1;
                else if(b[mid]<a[i]) l=mid+1;
                else {
                    printf("%d ",a[i]);
                    break;
                }
            }
        }
```

```
        putchar('\n');
    }
    return 0;
}
```

H. Prince and Princess

解题思路

如果我们要保证一定能找到公主在哪个房间，就必须以最坏的情况来考虑，所以可以直接把**立场不确定**的全都当成**说假话**的，这样就只有**说真话**的和**说假话**的两类人，假设这两类人的数量分别是 a 和 b，我们来考虑 a 和 b 的数量关系：

如果 $a \leq b$：那么只要所有说假话的人对每一个问题都众口一词地回答同一个假的答案，我们就无法确定任何信息。

如果 $a > b$：那么只要我们对每个人都询问公主在哪个房间，如果同一个答案出现的次数超过了 b 次，那么我们就可以确定这个答案就是公主的房间。需要询问 $2b+1$ 次。那么为什么这个次数一定是最少的询问呢？因为我们要确认任何信息，都必须经过 $2b+1$ 次的询问才能真正确认，所以直接问公主的房间一定是最少次数的。

最后要注意特殊情况，如果只有一个人，那么这个人只能是公主，所以不用询问。

AC代码

```
#include <stdio.h>

int main()
{
    int n,a,b,c;
    scanf("%d",&n);
    while(n--)
    {
        scanf("%d%d%d",&a,&b,&c);
        if(a<=b+c) puts("NO");
        else if(a==1) puts("YES\n0");
        else printf("YES\n%d\n",2*(b+c)+1);
    }

    return 0;
}
```

I. 寻找大橙羊

解题思路

本题旨在考查对于C语言内置函数库`string.h`的掌握。首先将数据读入并储存（记得去掉第一个数据组数之后的`\n`），再利用函数`tolower`将读入字符串中字符小写，之后`strstr`寻找子串，找到后输出从开始到替换位置的字符串，以及`grass`，再从下个位置继续重复上诉操作即可。

需要注意的是，`C`语言中内置的字符串函数处理字符串时，`\0`会被认为是一个字符串的结尾，标准代码中多处使用这种方法对字符串进行处理。

AC代码

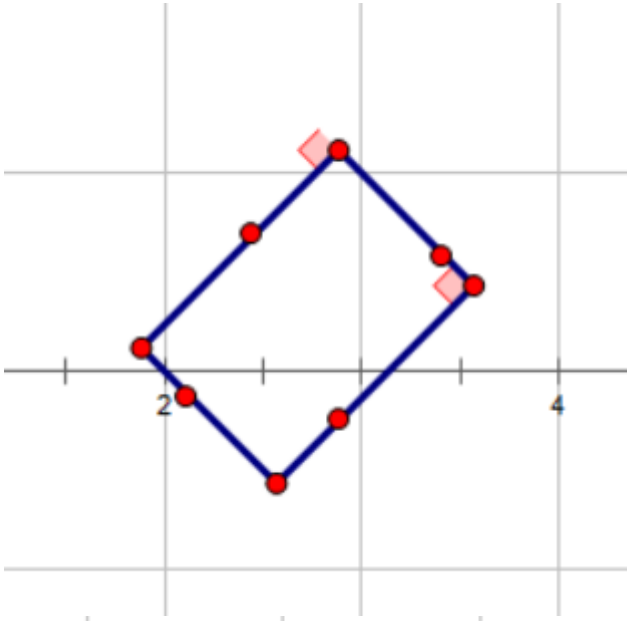
```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main ()
{
    char txt_little[1024],txt_big[1024],os[]="orangesheep",gs[]="grass";
    int oslong = 11,k = 0,n,num = 0;
    char *judge,*judge1;
    scanf("%d\n",&n);
    while(n--) {
        gets(txt_big);
        while (txt_big[k] != '\0') {
            txt_little[k] = tolower(txt_big[k]);
            k++;
        }
        txt_little[k] = '\0';
        judge = txt_little;
        judge1 = txt_big;
        while(strstr(judge, os)!=NULL) {
            txt_big[strstr(judge, os) - txt_little] = '\0';
            num++;
            printf("%s",judge1);
            printf("%s",gs);
            judge1 = strstr(judge, os) - txt_little + oslong + txt_big;
            judge =  strstr(judge, os) + oslong;
        }
        printf("%s\n",judge1);
        k = 0;
        memset(txt_little, 0, sizeof(txt_little));
        memset(txt_big, 0, sizeof(txt_big));
    }
    printf("%d",num);
    return 0;
}
```

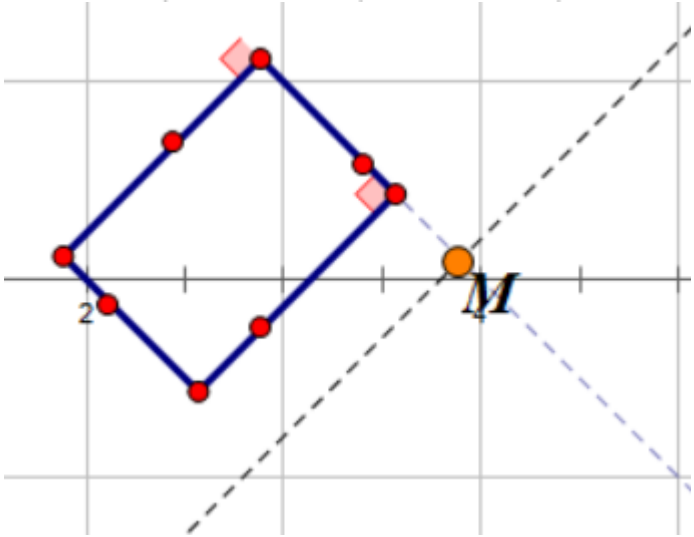
J. 选房间

解题思路

根据曼哈顿距离的性质可以得出，二维的情况下，若所有点在$x+y<=a$，$x+y>=b$，$x-y<=c$，$x-y>=d$围成的范围内，$a>b, c>d$,则两点距离最大值D不会超过$max(a-b, c-d)$。这与欧几里得距离对应的是一个圆类似。

所以找到恰好满足条件的临界系数$abcd$即可。对于所有区域内的点，有$2x<=a+c$。如使x取得最大值且两点间最大距离D不变，则可使用计算出的D调整临界系数，使$a=D+b$，$c=D+d$，最大可取的x即为$(a+c)/2$。





三维的情况下，相当于有8个平面进行约束，使用同样的方法确定D和各平面的临界系数，并对临界系数进行调整。最后可推导出对于最大的z有$x+y+z<=a, x-y+z<=b, -x+y+z<=c, -x-y+z<=d$四个约束方程，$abcd$为使用$D$调整过后的临界系数。这时$2z<=a+d$及$b+c$，取$z=min(a+d, b+c)/2$。

对于移动时间的调整，直接使用线性变换，对点的位置进行缩放即可。

AC代码

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

```
#define MAXN 10000
#define LL long long
#define max(a, b) (a > b ? a : b)
#define min(a, b) (a < b ? a : b)
int main(){
    int t = 0, i, j;
    scanf("%d", &t);
    while(t--){
        int xt, yt, zt, x, y, z, j;
        int maxa = 1<<31, maxb = 1<<31, maxc = 1<<31, maxd = 1<<31;
        int mina = maxa - 1, minb = maxa - 1, minc = maxa - 1, mind = maxa - 1;
        int n;
        scanf("%d%d%d%d", &xt, &yt, &zt, &n);
        for (i = 0; i < n; ++i) {
            scanf("%d%d%d", &x, &y, &z);
            x *= xt;
            y *= yt;
            z *= zt;
            maxa =  max(maxa, x + y + z);
            maxb =  max(maxb, x - y + z);
            maxc =  max(maxc, -x + y + z);
            maxd =  max(maxd, -x - y + z);
            mina =  min(mina, x + y + z);
            minb =  min(minb, x - y + z);
            minc =  min(minc, -x + y + z);
            mind =  min(mind, -x - y + z);
        }
        int a = maxa - mina, b = maxb - minb, c = maxc - minc, d = maxd - mind;
        int maxx = max(max(a, b), max(c, d));
        maxa += maxx - a, maxb +=  maxx - b,
        maxc +=  maxx - c, maxd +=  maxx - d;
        z = min(maxa+maxd, maxb+maxc) / (2 * zt);
        printf("%d\n", z);
    }
}
```

还有一种稍慢但更易想出的做法是二分。

首先计算出最大距离D。最高高度的范围在已有的最高的点之上长度为D的区间内。对于一个高度h，将所有已知点投影到高度h的平面上。对于每一个高度为a的点在高度h的平面上投影，可以得到一个斜正方形的符合条件的区域，该区域内点与投影点的距离小于$D-(h-a)$。所有点在平面上产生的区域求交集，如果不为空集则h符合要求。

AC代码2

```
/*
 Author: 傅云濠(34705)
 Result: AC Submission_id: 2038161
 Created at: Sat Nov 09 2019 20:16:29 GMT+0800 (CST)
 Problem: 2670  Time: 363   Memory: 2820
*/

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

```
#include<math.h>
typedef long long LL;
#define mem(a,b) memset(a,b,sizeof(a))
int MAX(int a,int b){return a>b ? a : b;}
int MIN(int a,int b){return a<b ? a : b;}
int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
int T,a,b,c,n,p[100010][3],MaxL,maxH;
const int oo=2147483647;
struct Rec{int X1,Y1,X2,Y2;};
struct Rec get(struct Rec a,struct Rec b)
{
    struct Rec C;
    C.X1=MAX(MIN(a.X1,a.X2),MIN(b.X1,b.X2));
    C.Y1=MAX(MIN(a.Y1,a.Y2),MIN(b.Y1,b.Y2));
    C.X2=MIN(MAX(a.X1,a.X2),MAX(b.X1,b.X2));
    C.Y2=MIN(MAX(a.Y1,a.Y2),MAX(b.Y1,b.Y2));
    return C;
}
int calc()
{
    int ans=0,Min,Max;
    int i,j,k;
    for(i=0;i<8;i++)
    {
        Min=oo,Max=-oo;
        for(j=0;j<n;j++)
        {
            double sum=0;
            for (k=0;k<3;k++)
            {
                int t=i&1<<k;
                if(t)sum+=p[j][k];
                else sum-=p[j][k];
            }
            Max=MAX(Max,sum);Min=MIN(Min,sum);
        }
        ans=MAX(ans,Max-Min);
    }
    return ans;
}
struct Rec r[100010];
int judge(int index)
{
    int i;
    struct Rec C;
    for(i=0;i<n;i++)
    {
        int L=MaxL-index+p[i][2];
        if(L<0)return 0;
        int X1=p[i][0],Y1=p[i][1]+L,X2=p[i][0],Y2=p[i][1]-L;
        r[i].X1=(X1-Y1);r[i].Y2=(X1+Y1);
        r[i].X2=(X2-Y2);r[i].Y1=(X2+Y2);
```

```
        if(i==0)C=r[0];
        else
        {
            C=get(C,r[i]);
            if(!(C.X2>=C.X1 && C.Y2>=C.Y1))return 0;
        }
    }
    maxH=MAX(maxH,index);
    return 1;
}
int main()
{
    int i,j;
    T=read();
    while(T--)
    {
        MaxL=0;maxH=0;
        mem(p,0);
        a=read();b=read();c=read();n=read();
        for(i=0;i<n;i++)p[i][0]=read()*a,p[i][1]=read()*b,p[i][2]=read()*c;
        MaxL=calc();
        int L=0,R=0;
        for(i=0;i<n;i++)L=MAX(L,p[i][2]);
        R=(L+MaxL)/c;L=L/c;
        while(R-L>1)
        {
            int mid=L+R>>1;
            if(judge(mid*c))L=mid;
            else R=mid;
        }
        judge(L*c);judge(R*c);
        printf("%d\n",maxH/c);
    }
    return 0;
}
```

K. Gauss

解题思路

高斯消元法解方程，根据高代所学知识一步步模拟即可。这里提供了一个方程数和未知数不一定相同的高斯消元模板。

AC代码

```
#include<stdio.h>
#include<math.h>
double eps=1e-9;
double a[220][220],x[220];
int equ,var;//方程数 未知数
void swap(double *a,double *b){double t=*a;*a=*b;*b=t;}
int Gauss(){
    int i,j,k,col,max_r;
    for(k=0,col=0;k<equ&&col<var;k++,col++){
        max_r=k;
        for(i=k+1;i<equ;i++)
            if(fabs(a[i][col])>fabs(a[max_r][col]))
                max_r=i;
        if(fabs(a[max_r][col])<eps)return 0;
        if(k!=max_r){
            for(j=col;j<var;j++)
                swap(&a[k][j],&a[max_r][j]);
            swap(&x[k],&x[max_r]);
        }
        x[k]/=a[k][col];
        for(j=col+1;j<var;j++)a[k][j]/=a[k][col];
        a[k][col]=1;
        for(i=0;i<equ;i++)
            if(i!=k){
                x[i]-=x[k]*a[i][col];
                for(j=col+1;j<var;j++)a[i][j]-=a[k][j]*a[i][col];
                a[i][col]=0;
            }
    }
    for(i=0;i<var;i++)printf("%.2f ",x[i]);
    return 1;
}
int main(){
    int i,j;
    scanf("%d",&equ);var=equ;
    for(i=0;i<equ;i++){
        for(j=0;j<var;j++){
            scanf("%lf",&a[i][j]);
        }
        scanf("%lf",&x[i]);
    }
    if(!Gauss())printf("No Solution!");
    return 0;
}
```