T1 喷泉

根据初中数学知识，可以知道，一个定点到圆上点的最大（小）距离等于其到圆心的距离加（减）半径的长度，而一个定点到一条线段的最大距离显然是到线段两端之一，最小距离是垂线段长度（题目保证垂足在线段上）。

于是输出圆心到线段距离减半径，到两端点的最大距离加半径即可。

别溢出了。

具体证明问初中数学老师，剩下的就是解析几何计算了。

```
#include <bits/stdc++.h>
using namespace std;
signed main(){
    #define int long long
    int t, x1, y1, x2, y2, x3, y3, r;
    cin >> t;
    while(t--){
        cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> r;
        int A=y1-y2,B=x2-x1;
        int C=-(A*x1+B*y1);
        long double minm=abs(A*x3+B*y3+C)/sqrt((long double)A*A+B*B)-r;
        long double maxm=sqrt((long double)(x3-x1)*(x3-x1)+(y3-y1)*(y3-y1));
        maxm=max(maxm,sqrt((long double)(x3-x2)*(x3-x2)+(y3-y2)*(y3-y2)))+r;
        // printf("%.2lf %.2lf\n",(double)minm,(double)maxm);
        printf("%.2Lf %.2Lf\n",minm,maxm);
    }
    return 0;
}
```

T2 红绿灯

其实本题暴力就大致能通过了，只不过有一些大优化。

题目简述：维护一个长度为 n 的序列，初始为 1 至 n。之后有 m 次操作，每次输入一个整数 a，将序列中每一个元素 x 变成 $\lceil \frac{x}{a} \rceil \times a$，问最终序列。

可以发现在一系列操作之后整个序列中有很多相同的元素，那么我们去重一下，并且记录一下每一个数字出现在序列中的哪些位置，显然是一个区间。

这样，我们记录的数字个数就大大减少，可以拿到约 $70pts$。

可以发现在 $2, 3$ 循环的数据中跑的很慢，而我们输出序列发现在之后的操作中，序列甚至一点都没变。

这是因为序列中的每一个元素**都整除了** a，所以根本不会变。

这样我们可以在每一次操作后，求一下整个序列所有元素的 gcd，一旦操作的 a 是 gcd 的约数，直接跳过不用修改。

由于本题数据较为随机，所以可以通过。

```
#include<bits/stdc++.h>
#define int long long
```

```cpp
using namespace std;
int b[300001][2];
int a[300001][2],n,m;
int gcd(int a,int b){
    return b?gcd(b,a%b):a;
}
signed main(){
    cin >> n >> m;
    a[0][0]=n;
    for(int i=1;i<=n;i++)a[i][0]=b[i][0]=i;
    int G=1, x;
    int sum=0;
    for(int i=1;i<=m;i++){
        cin >> x;
        if(G%x==0)continue;
        int now=i&1,las=now^1;
        sum+=a[0][las];
        a[0][now]=0;
        for(int j=1;j<=a[0][las];j++)a[j][las]=((a[j][las]-1)/x+1)*x;
        for(int j=1;j<=a[0][las];j++){
            if(j==1||a[j][las]!=a[j-1][las])
                a[++a[0][now]][now]=a[j][las];
            b[a[0][now]][now]=b[j][las];
        }
        G=a[1][now];
        for(int j=1;j<=a[0][now];j++)G=gcd(G,a[j][now]);
    }
    sum+=a[0][m&1];
    int j=0;
    for(int i=1;i<=n;i++){
        while(b[j][m&1]<i)j++;
        cout << a[j][m&1] << " ";
    }
}
```

T3 子集

注意到一个结论：

$$F_k(S)=\sum_{T\subseteq S}F_0(T)\cdot k^{|S|-|T|} \tag{1}$$

其中 $k>0$。

证明：$k=1$ 时显然成立。

若对任意 $k\le r$ 均有 (1) 成立，那么当 $k=r+1$ 时：

$$\begin{aligned}F_{r+1}(S)&=\sum_{T\subseteq S}F_r(T)=\sum_{T\subseteq S}\sum_{C\subseteq T}F_0(C)\cdot r^{|T|-|S|}\\&=\sum_{C\subseteq S}F_0(C)\cdot\sum_{i=0}^{|S|-|C|}\binom{|S|-|C|}{i}r^i\\&=\sum_{C\subseteq S}F_0(C)\cdot(r+1)^{|S|-|C|}\end{aligned}$$

因此 (1) 同样成立。从而，对任意正整数 k 均有 (1) 成立。

因此我们现在要做的就是：对每个和为 M 的子集 $T \subseteq S$，计算其贡献 $k^{|S|-|T|}$，最后加到一起。

可以考虑一个显然的 dp：设 $F(i,j,w)$ 表示前 i 个数中选了 j 个，其和为 w 的方案数。

枚举最后一个数选不选，不难得到转移方程

$$F(i,j,w) = F(i-1,j,w) + F(i-1,j-1,w-a_i)$$

那么答案就是

$$\sum_{j=0}^{n} F(n,j,M) \cdot k^{n-j}$$

时间复杂度为 $O(n^2M)$，可以获得 60 分。

考虑优化一下状态设计：设 $F(i,w)$ 表示 $\{a_1, a_2, \cdots, a_i\}$ 的所有和为 w 的子集的贡献之和。

同样考虑最后一个数选不选：

- 如果没有选 a_i，那么相当于式 (1) 中的 $|S|$ 加了 1，因此 k 的指数也会 $+1$；
- 反之，则 $|T|$ 也会 $+1$，那么原封不动加进来就行了。

因此，我们得到

$$F(i,w) = F(i-1,w) \cdot k + F(i-1,w-a_i)$$

时间复杂度 $O(nM)$，可以通过。

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int MN=5005;
const int mod=1e9+7;
int a[MN],dp[MN][MN],n,M,k;
void solve(){
    memset(dp,0,sizeof(dp));
    cin >> n >> M >> k;
    for(int i=1;i<=n;i++) cin >> a[i];
    dp[0][0]=1;
    for(int i=1;i<=n;i++){
        for(int j=0;j<=M;j++){
            dp[i][j]=dp[i-1][j]*k%mod;
            if(j>=a[i])dp[i][j]=(dp[i][j]+dp[i-1][j-a[i]])%mod;
        }
    }
    cout<<dp[n][M]%mod<<endl;
}
signed main(){
    int tt;
    cin >> tt;
    while(tt--)solve();
}
```

T4 光

20pt

n^4 枚举每一盏灯的耗电量，然后按照公式计算每一个格子的亮度是否符合要求。

70pt

n^3 枚举三盏灯的耗电量，计算出此时每个格子的亮度，算出每个格子当前亮度与需求亮度的差值，这个差值由第四盏灯来填补。即枚举好三盏灯之后，我们可以 O(1) 算出第四盏灯的最低耗电量。

100pt

本题显然具有二分性，考虑二分答案。假设总共有 mid 的耗电量可以分配给四盏灯，我们考虑如何分配：n^2 枚举对角线的两盏灯的耗电量，假设枚举的是左上角 a 和右下角 b。那么我们可以发现，右上角这个格子当前亮度为 $\lfloor \frac{a}{2} \rfloor + \lfloor \frac{b}{2} \rfloor$，左下角的格子的亮度也一样。我们可以算出这两个格子的亮度与需求的差值，再将 $mid - a - b$ 的耗电量分配给这两盏灯。 一种减少写代码细节的小技巧是：算出左下角格子的大致耗电量，然后用 for 循环在估计值的附近枚举即可，这样就不需要判断边界条件了。