AK杯题解

A. 想念Serein

本题是本场比赛的第一题，签到题（意思就是参加这场比赛的人都能写出这道题）

考点与坑点

输出语句

坑点在于要用空行分隔，有可能有部分参赛者读漏题。

同时注意读题，`A1m Miss Using`和`Tension Miss Niannian`不需要输出

题意和思路

输出三遍`I Miss Serein`即可

参考代码

C

```c
#include<stdio.h>

int main()
{
    for(int i=0;i<3;i++)
    {
        printf("I Miss Serein\n");
    }
    return 0;
}
```

Python

```python
print("I Miss Serein\n" * 3)
```

B. Serein的排序

考点与坑点

`abs()`函数，分支条件语句的运用

这题的坑点在于，要输出的是原本的数字

题意

给定三个绝对值不相等的整数x, y, z，请你将它们按照绝对值大小升序排序并输出。

思路

最简单的方法就是把它们读入到各自的变量中，然后各自再用一个变量来存它们的绝对值，最后进行比较

总共有6种可能性

将三个变量从小到大排序的话就是

abc，acb，bac，bca，cab，cba

所以我们把这三种情况写成分支语句即可

注意，这里实际上存在更简单的解法，但这里只是放出了新手比较习惯的写法

参考代码

C

```c
#include<stdio.h>

int main()
{
    int a, b, c;
    scanf("%d%d%d",&a,&b,&c);
    int ta = abs(a);
    int tb = abs(b);
    int tc = abs(c);
    if (ta < tb && tb < tc)
    {
        printf("%d %d %d\n",a,b,c);
    }
    else if (ta < tc && tc < tb)
    {
        printf("%d %d %d\n",a,c,b);
    }
    else if (tb < ta && ta < tc)
    {
       printf("%d %d %d\n",b,a,c);
    }
    else if (tb < tc && tc < ta)
    {
        printf("%d %d %d\n",b,c,a);
    }
    else if (tc < ta && ta < tb)
    {
        printf("%d %d %d\n",c,a,b);
    }
    else if (tc < tb && tb < ta)
    {
        printf("%d %d %d\n",c,b,a);
    }
    return 0;
}
```

Python

```python
a, b, c = map(int, input().split())
ta = abs(a)
tb = abs(b)
tc = abs(c)
if ta < tb < tc:
    print(f"{a} {b} {c}")
elif ta < tc < tb:
    print(f"{a} {c} {b}")
elif tb < ta < tc:
    print(f"{b} {a} {c}")
elif tb < tc < ta:
    print(f"{b} {c} {a}")
elif tc < ta < tb:
    print(f"{c} {a} {b}")
elif tc < tb < ta:
    print(f"{c} {b} {a}")
```

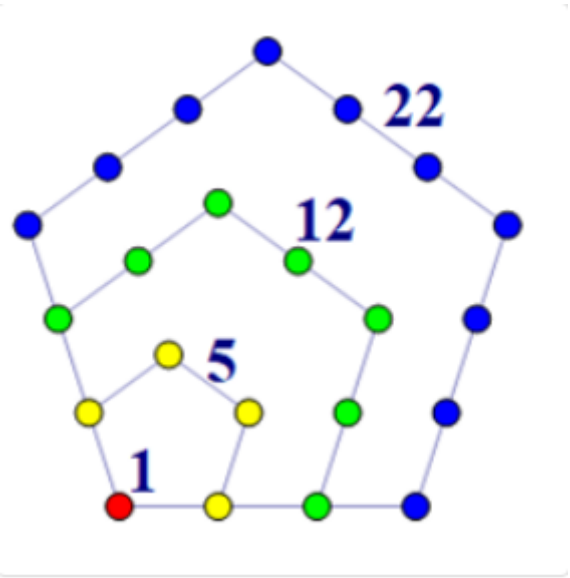
C. 五边形数

考点

for循环的使用，找规律

题意

将无限个点按下图的方式摆放

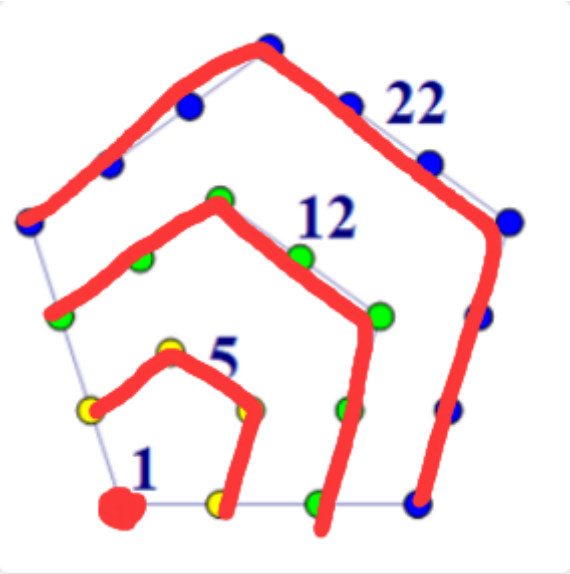


定义第 i 个五边形数是图中第 i 小的五边形的边上及其内部包含的点的个数，在这里我们特殊规定，只有一个点也能算五边形

思路

所以，第一小的五边形是红色的点，第二小的五边形是黄色的点及角落围成的五边形，第三小的五边形是绿色的点及角落围成的五边形，以此类推。

现在我们找到了五边形，需要来确定五边形数。从左下角向右上角看，我们都是给一个“环”包围住的



环就是如图所示的红线所覆盖的点

可以发现，我们每次依次用红线覆盖环后，记本次覆盖是第 i 次，那么，在这前 i 次覆盖后，我们正好可以得到第 i 个第五边形数。

这就引出了我们的结论，第 i 个答案是由前 $i-1$ 次覆盖，再加上本次覆盖得到的。

于是我们用一个变量`curr`来存储之前覆盖的点数，再加上本次覆盖的答案就能得到第 i 个答案。于是，我们可以用for循环来迭代。

现在的问题是，怎么计算本次覆盖的数。

我们观察，第 1 次覆盖是 1，第 2 次覆盖是 4，第 3 次覆盖是 7，第 4 次覆盖是 10，。。。

可以看出通项公式：

第 i 次覆盖是 $(i-1)*3+1$。

实际上其实我们可以推公式，这样无需for循环，可以一次性算出来，但这里会展示较符合新手思路的解法。

参考代码

C

```c
#include<stdio.h>

int main()
{
    int n;
    scanf("%d", &n);
    int curr = 0;
    for(int i = 1; i <= n; i++)
    {
        curr += (i - 1) * 3 + 1;
    }
```

```
    printf("%d", curr);
    return 0;
}
```

Python

```
n = int(input())
curr = 0
for i in range(1, n + 1):
    curr += (i - 1) * 3 + 1
print(curr)
```

D. 排列的排名

考点

数组，模拟

题意

排列指的是，对于$[1,n]$中的整数，它们在数组中出现次数都是1，而且排列中没有除了这n个整数外的数

对于每个排列，它们在字典序的定义下都有一个排名，这个排名可以用以下的方法算出：

1. 设初始排名为 1。
2. 对排列 p 的第 i 个元素 p_i ，找到有多少个整数 x 满足 $x \in [1,p_i]$，且 x 未在 $p_1,p_2,\cdots,p_i$ 出现过，设共有 c_i 个数字满足条件，则对排名增加 $c_i \times (n-i)!$。

那么依次迭代就可以算出答案。

思路

实际上你需要做的就是对这个过程进行模拟。

参考代码

C

```
#include<stdio.h>
int fac[20];
int p[20];
int main()
{
    fac[0] = 1;
    for (int i = 1; i <= 13; i++)fac[i] = fac[i - 1] * i;//预处理阶乘
    int n;
    scanf("%d", &n);
    int res = 1;//最终排名
    for (int i = 1; i <= n; i++)
    {
        scanf("%d", &p[i]);
        int cnt = p[i];//在[1, p[i]]中，有多少个数不在p[1, i]中
```

```
        for (int j = 1; j <= i; j++)
        {
            if (p[j] <= p[i])cnt--;//存在于[1, p[i]]中，那么就cnt--
        }
        res += cnt * fac[n - i];//结果
    }
    printf("%d", res);
    return 0;
}
```

Python

```
fac = [1 for i in range(14)]
for i in range(1, 14):
    fac[i] = fac[i - 1] * i
n = int(input())
res = 1
p = [int(x) for x in input().split()]
p.insert(0, 0)
for i in range(1, n + 1):
    cnt = p[i]
    for j in range(1, i + 1):
        if p[j] <= p[i]:
            cnt -= 1
    res += cnt * fac[n - i]
print(res)
```

E. 鸿门宴

考点

简单博弈，思维

题意

对于一个 $1 \leq a_1 < a_2 < a_3 < \ldots < a_n$ 的序列，两人轮流选择 a_i 减一，但不能破坏 $1 \leq a_1 < a_2 < a_3 < \ldots < a_n$ 的性质，无法操作者输。

思路

我们可以得到以下结论：对于长度为 n 的序列，当序列无法操作时，一定是从 1 到 n 的顺序排列：$[1, 2, 3, \ldots, n]$。

因为如果序列不满足上述的状态，即存在 $a_i - a_{i-1} > 1$，那么显然当前玩家可以选择 a_i 使其减一，所以通过反证可以得知无法操作的情况一定是从 1 到 n 的顺序排列：$[1, 2, 3, \ldots, n]$。

由于初始的序列和结束的序列是确定的，那么可操作的次数也显然是确定的，即为 $\sum_{i=1}^{n} a_i - i$，分析出这个结论离真相就不远了。

因为轮流操作，先手都是在第奇数次操作，后手都是在第偶数次操作。所以，若可操作奇数次，则先手胜利；若可操作偶数次，则后手胜利。

最后，AC代码：

参考代码

C

```c
#include <stdio.h>
int n, s;
int main()
{
    scanf("%d",&n);
    for (int i = 1, a; i <= n; ++i)
    {
        scanf("%d",&a), s += a - i;
    }
    printf(s % 2 == 0 ? "baicha" : "guodong");
    return 0;
}
```

Python

```python
n = int(input())
a = [int(i) for i in input().split()]
s = sum([a[i] - (i + 1) for i in range(n)])
print('guodong' if s % 2 else 'baicha')
```

F. 数学王子Serein

考点与坑点

分析能力，分类讨论

坑点是小细节较多

题意

对于给定的正整数 s， s 是否能被表示为 n 个连续正偶数的和

思路

需要注意的是，我们的评测机一秒只能进行大约 10^8 次运算，而本题的时限为 1 秒，然后，数据范围为 $1 \leq n \leq s \leq 1 \times 10^{12}$。所以，我们不能进行简单的的枚举，因为这会超出时限。

假设问题有解

$\frac{s}{n}$ 是这 n 个连续正偶数的平均值，可证，问题的解是唯一的。

考虑 $\frac{s}{n}$ 不为整数，那就意味着，这 n 个连续正偶数的平均值不为偶数，这很明显是错误的，因为偶数之和仍然还是偶数，而偶数一定可以整除 2。

所以 $\frac{s}{n}$ 不为整数的话，即 n 不能整除 s 的话，说明问题无解。

假设 n 为奇数，那么 $\frac{s}{n}$ 就正好等于这些连续正偶数的中位数

它，这个中位数，相对于第一个正偶数的距离应该是相差的偶数个数（项数）$\frac{n-1}{2}$乘上间距（公差）2，所以，它相对于第一个正偶数的距离是$n-1$

假设 n 为偶数，那么$\frac{s}{n}$就是中间两个偶数的平均值，也就是说它是位于中间两个偶数的中间，那么它相对于第一个正偶数的距离应该是左边的偶数和第一个正偶数相差的偶数个数（项数）$\frac{n}{2}-1$乘上间距（公差）2 再加上平均值离左边偶数的距离 1，所以它相对于第一个正偶数的距离是 $n-2+1=n-1$

综上，可以发现，无论 n 的奇偶性，$\frac{s}{n}$这个平均值距离第一个正偶数的距离都是 $n-1$，由对称性，我们可以知道

第一个正偶数，也就是最小的偶数是 $\frac{s}{n}-(n+1)$

最后一个正偶数，也就是最大的偶数是 $\frac{s}{n}+(n+1)$

答案呼之欲出，然而，我们还需要检验以下情况：

1、第一个正偶数是否是正数

2、第一个正偶数是否是偶数

一旦有情况是否定的，那么我们的答案都是不合法的。

参考代码

C

```c
#include<stdio.h>

int main()
{
    long long s, n;
    scanf("%lld%lld",&s,&n);
    long long lans = s / n - (n - 1), rans = s / n + (n - 1);
    if (s % n == 0 && lans % 2 == 0 && lans > 0)
    {
        printf("%lld\n",rans);
    }
    else printf("-1\n");
    return 0;
}
```

Python

```python
s, n = map(int, input().split())
lans = s // n - (n - 1)
rans = s // n + (n - 1)
if s % n or lans % 2 or lans <= 0:
    print("-1")
else:
    print(rans)
```

G. 只因你太美

考点与坑点

考点：模拟

坑点：数组长度需开两倍大小，因为可能出现全为 `ji` 的情况

思路

本题思路没有难点，跟着题目要求做就好，难点在于实现较复杂

参考代码

C

```c
#include <stdio.h>
#include <string.h>
#define ll long long
ll n, sumj, sumzy, m, a[105], b[105];
char s[105][105][23], t[105][105*2][23];
signed main()
{
    scanf("%lld", &n);
    for (ll i = 1; i <= n; ++i)
    {
        scanf("%lld", &a[i]);
    }
    for (ll i = 1; i <= n; ++i)
    {
        for (ll j = 1; j <= a[i]; ++j)
        {
            scanf("%s", s[i][j]);
        }
        int beauty = 0;
        for (int j = 1; j <= a[i]; ++j)
        {
            if (!strcmp(s[i][j], "ji"))
            {
                ++sumj;
                beauty = 1;
            }
            if (j < a[i] && !strcmp(s[i][j], "zhi") && !strcmp(s[i][j + 1],
"yin"))
```

```
            {
                ++sumzy;
                beauty = 1;
            }
        }
        if (beauty)
        {
            ++m;
            for (ll j = 1; j <= a[i]; ++j)
            {
                if (!strcmp(s[i][j], "ji"))
                {
                    strcpy(t[m][++b[m]], "zhi");
                    strcpy(t[m][++b[m]], "yin");


                }
                else if (j < a[i] && !strcmp(s[i][j], "zhi") && !strcmp(s[i][j + 1], "yin"))
                {
                    strcpy(t[m][++b[m]], "ji");
                    ++j;
                }
                else
                {
                    strcpy(t[m][++b[m]], s[i][j]);
                }
            }
        }
    }
    printf("%lld\n%lld\n", sumj * sumj + sumzy * sumzy, m);
    for (ll i = 1; i <= m; ++i)
    {
        for (ll j = 1; j <= b[i]; ++j)
        {
            printf("%s ", t[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

C++

```
#include <bits/stdc++.h>

using namespace std;

using ll = long long;

int main()
{
    int n;
    cin >> n;
```

```
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];

    ll ans = 0;
    vector<string> vs;

    vector<vector<string>> g;
    int x = 0, y = 0;
    for (int i = 0; i < n; i++)
    {
        vs = vector<string>(a[i]);
        for (int j = 0; j < a[i]; j++)
        {
            cin >> vs[j];
        }

        vector<string> f;
        int h = 0; //记录是否出现ji或zhi yin
        for (int j = 0; j < a[i]; j++)
        {
            if (vs[j] == "ji") x++, h = 1, f.push_back("zhi"),
f.push_back("yin");  //出现ji，替换成zhi yin
            else if (j + 1 < a[i] && vs[j] == "zhi" && vs[j+1] == "yin")//出现zhi
yin，替换成ji
             y++, h = 1, f.push_back("ji"), j++;
            else f.push_back(vs[j]);  //存入普通单词
        }


        if (h > 0) g.push_back(f); //出现过ji或zhi yin，保存
    }

    ans = x * x + y * y;

    cout << ans << "\n";
    cout << g.size() << "\n";
    for (auto&& v : g)
    {
        for (auto&& s : v)
        {
            cout << s << " ";
        }
        cout << "\n";
    }

    return 0;
}
```

Python

普通解法:

```
n = int(input())
a = [int(i) for i in input().split()]
```

```
t = []
sumj = sumzy = 0
for i in range(n):
    s = input().split()
    beauty = 'ji' in s
    beauty += sum(
        [s[i] == 'zhi' and s[i + 1] == 'yin' for i in range(a[i] - 1)])
    if beauty:
        t.append([])
        j = 0
        while j < a[i]:
            if s[j] == 'ji':
                t[-1].extend(['zhi', 'yin'])
                sumj += 1
            elif s[j] == 'zhi' and j + 1 < a[i] and s[j + 1] == 'yin':
                t[-1].append('ji')
                sumzy += 1
                j += 1
            else:
                t[-1].append(s[j])
            j += 1
print('%d\n%d' % (sumj**2 + sumzy**2, len(t)))
for i in t:
    for j in i:
        print(j, end=' ')
    print()
```

利用正则表达式：

```
import re

n = int(input())
a = [int(i) for i in input().split()]
r = []
sumj = sumzy = 0
for i in range(n):
    s = input()
    newj = len(re.findall(r'\bji\b', s))
    newzy = len(re.findall(r'\bzhi yin\b', s))
    if newj + newzy == 0:
        continue
    sumj, sumzy = sumj + newj, sumzy + newzy
    s = re.sub(r'\bji\b', '#', s)
    s = re.sub(r'\bzhi yin\b', 'ji', s)
    s = re.sub('#', 'zhi yin', s)
    r.append(s)
print('%d\n%d' % (sumj**2 + sumzy**2, len(r)))
for i in r:
    print(i)
```

H. Serein的概率论

考点与坑点

考点：分析能力，位运算，概率

坑点就是只需要用到第一个考点

题意

对 n 个 01 串，有 q 次操作，每次选择两个 01 串进行按位与的操作，并将结果赋值给这两个数字串，每次有 $\frac{p}{100}$ 操作成功。求最终所有 01 串按位与得到的数字串中 1 的个数期望。

思路

这道题看起来很像期望dp，但显然AK杯并不会考察这么超前的知识，所以我们对题目进行分析：

我们可以发现，如果在第 i 个数位上有 0，那么 $0\&1=0$，$0\&0=0$，也就是说无论操作成功与否这个数位上一定有 0，那么最终按位与的操作后，第 i 数位就一定是 0；反之如果第 i 个数位上全是 1，那么无论怎么操作，这个数位都不会出现 0，所以按位与操作后，第 i 个数位一定是 1。

经过分析，我们可以发现这题其实与操作成功的概率无关，甚至与这 q 次操作都无关，最终 1 的个数期望就是最初所有数字串按位与得到的 1 的个数。

参考代码

C

```c
#include <stdio.h>
int n,m,q,num[1010];
int ans;
char s[110][1010];
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
    {
        scanf("%s",s[i]);
        for(int j=1;j<=m;j++)
        {
            if(s[i][j-1]=='1') num[j]++;
        }
    }
    scanf("%d",&q);
    while(q--)
    {
        int ii,jj,l,r,p;
        scanf("%d%d%d%d%d",&ii,&jj,&l,&r,&p);
    }
    for(int i=1;i<=m;i++)
    {
        if(num[i]==n) ans++; //该数位上全是1
    }
    printf("%d\n",ans);
    return 0;
}
```

Python

```python
n, m = [int(i) for i in input().split()]
t = [1] * m
for i in range(n):
    s = [int(j) for j in input()]
    t = [t[j] & s[j] for j in range(m)]
print(sum(t))
```

I. Ustinian的围骰

题解

如果直接暴力枚举每个人，一共需要枚举除自己外的 $n-1$ 个人，且每个人的骰子有 $6^5-5!$ 种选择，那么枚举总状态数为 $(6^5-5!)^{n-1}=7656^4\approx 3.4\times 10^{15}$ 种状态，显然直接暴力 DFS 是会超时的。

考虑将上述 DFS 拆分为两个 DFS。

我们首先预处理出每种骰子点数 $[1,6]$ 的所有状态数，设 $c_{r,x,m}$ 代表一个人的五个骰子里，r(斋/飞)时点数为 x 的骰子有 m 个的方案数。可以用 x 次 DFS 枚举所有 6^5 种状态，求出所有 $c_{r,x,m}$，复杂度为 $O(x\cdot 6^5)=O(6\cdot 6^5)\approx 4.6\times 10^4$。这一步也可以用 x 进制枚举求解，复杂度同样。或者可以直接手算组合数学/打表，则复杂度 $O(1)$。

实际上可以发现，除了 $x=1$ 外，所有 $x\geq 2$ 的方案数 c 都是相同的，可以用分类讨论法将复杂度的 x 消为 2。(题解略去这一步)

有了 $c_{r,x,m}$ 之后，我们对每次询问，再枚举 $n-1$ 个人，每个人有 i 个 x，方案数有 $c_{r,x,i}$ 个。对于每次搜索得到的 $n-1$ 个人的方案数，根据乘法原理将它们乘起来，就能得到总方案数。此时复杂度为 $O(6^{n-1})$。这一步也可以用多项式乘法，则暴力预处理多项式复杂度为 $O(n\cdot x\cdot 6n)$，每次询问复杂度为 $O(6(n-1))$。

因此，考虑两次 DFS 的话，总复杂度为 $O(t(6^5+6^{n-1}))$，可以预处理第一个 DFS 优化为 $O(6^6+6^{n-1}t)$。

如果第二次 DFS 用暴力多项式优化，则总复杂度为 $O(6^6+6^4n+6tn)$。学习过多项式算法的选手可以尝试该解法。(本题的 hard-version 已公布，欢迎挑战)

> 对多项式方法，本质是乘法原理与加法原理。例如假设第一个人拿 k 个物品有 a_k 种方案，第二个人拿 k 个物品有 b_k 种方案。例如有 $a=3,4,5$，即第一个人有三种方案可以只拿一个物品，四种方案可以拿两个物品，五种方案拿三个物品。又有 $b=6,7,8$。则两个人共拿 4 个物品有 $3\times 8+6\times 5+4\times 7$ 种方案。将 a 表示为多项式 $3x+4x^2+5x^3$，b 表示为多项式 $6x+7x^2+8x^3$，那么两多项式相乘后 x^4 这一项的系数正好是 a,b 对应系数相乘再相加即 $3\times 8+6\times 5+4\times 7$。即 $(3x+4x^2+5x^3)(6x+7x^2+8x^3)$ 展开后 x^4 为所求。
>
> 具体到这一题，一个人点数加起来为 k 的方案数为多项式 $c_{r,x,1}x+c_{r,x,2}x^2+\cdots$ 里的 x^k 系数 $c_{r,x,k}$，两个人加起来为 k 的方案数可以看成是 $(c_{r,x,1}x+c_{r,x,2}x^2+\cdots)^2$ 展开后的 x^k 系数。同理，如果要求 n 个人加起来，就是求 $(c_{r,x,1}x+c_{r,x,2}x^2+\cdots)^n$ 的 x^k 系数，暴力模拟求之即可。

参考代码

DFS+DFS

C++

```
#include <bits/stdc++.h>
using namespace std;
#define sc(x) scanf("%lld", &x)
typedef long long ll;
ll c[3][7][10], t[10], t1[10], bp[10];
void dfs0(ll i, ll x)
{
    if (i == 5)
    {
        memcpy(t1, t, sizeof t);
        sort(t1, t1 + 5);
        if (t1[0] == 2 && t1[1] == 3 && t1[2] == 4 && t1[3] == 5 && t1[4] == 6)
        {
            return;
        }
        ll num1 = 0, num2 = 0; //斋,飞
        for (ll i = 0; i < 5; ++i)
        {
            num1 += t[i] == x;
            num2 += t[i] == x || t[i] == 1;
        }
        num1 += num1 == 5;
        num2 += num2 == 5;
        c[1][x][num1]++, c[2][x][num2]++;
        return;
    }
    for (ll y = 1; y <= 6; ++y)
    {
        t[i] = y;
        dfs0(i + 1, x);
    }
}
void dfs(ll rem, ll sum, ll need, ll &a, ll *c)
{
    if (rem == 0)
    {
        a += sum * (need <= 0);
        return;
    }
    for (ll i = 0; i <= 6; ++i)
    {
        dfs(rem - 1, sum * c[i], need - i, a, c);
    }
}
signed main()
{
    bp[0] = 1;
    for (ll i = 1; i <= 5; ++i)
    {
        bp[i] = bp[i - 1] * 7656;
    }
```

```
    for (ll i = 1; i <= 6; ++i)
    {
        dfs0(0, i);
    }
    ll t, n, r, x, tot;
    for (sc(t); t; --t)
    {
        sc(n), sc(r), sc(x), sc(tot);
        ll numx = 0;
        if (x == 1)
        {
            r = 1;
        }
        for (ll i = 1, v; i <= 5; ++i)
        {
            sc(v);
            numx += v == x || (v == 1 && r == 2);
        }
        if (numx == 5)
        {
            numx++;
        }
        ll u = max(0LL, tot - numx), a = 0, b = bp[n - 1];
        dfs(n - 1, 1, u, a, c[r][x]);
        ll c = __gcd(a, b);
        printf("%lld/%lld\n", a / c, b / c);
    }
    return 0;
}
```

Python

```
t = [0 for i in range(5)]  #当前dfs枚举的5个骰子状态
c = [[[0 for i in range(7)] for j in range(7)] for k in range(3)]  #斋/飞x总数


def dfs1(i, x):  #统计斋和飞时x下的方案数,O(6^5)=7776,也可以组合数学/六进制枚举
    if i == 5:
        b = t[:]
        b.sort()
        if b == [2, 3, 4, 5, 6]:
            return
        num1 = len(list(filter(lambda y: y == x, t)))  #斋
        num1 += num1 == 5
        c[1][x][num1] += 1
        num2 = len(list(filter(lambda y: y == x or y == 1, t)))  #飞
        num2 += num2 == 5
        c[2][x][num2] += 1
        return
    for y in range(1, 7):
        t[i] = y
        dfs1(i + 1, x)


for i in range(1, 7):
```

```
    dfs1(0, i)
for _ in range(int(input())):
    n, r, x, tot = [int(i) for i in input().split()]
    r = 1 if x == 1 else r
    a = [int(i) for i in input().split()]
    num = len(list(filter(lambda y: y == x or (y == 1 and r == 2), a)))
    num += num == 5
    tot = max(0, tot - num)
    fz, fm = 0, 7656**(n - 1)  #len(c1)==len(c2)==7656

    def dfs2(i, s, p):  #前i个人共s个x的方案数有p个,O(6^n);也可以用多项式
        global fz
        if i == n:
            if s >= tot:  #断言正确
                fz += p
            return
        for j in range(0, 7):
            dfs2(i + 1, s + j, p * c[r][x][j])

    def gcd(a, b):
        return gcd(b, a % b) if b > 0 else a

    dfs2(1, 0, 1)
    g = gcd(fz, fm)
    print('%d/%d' % (fz // g, fm // g))
```

x进制枚举+多项式

C++

```
#include <bits/stdc++.h>
using namespace std;
#define sc(x) scanf("%lld", &x)
typedef long long ll;
ll c1[7][10], c2[7][10], bp[10], f1[5][7][100], f2[5][7][100];
void init(ll x) //对x计算每个人的期望值
{
    ll c[10] = {}, b = 0; // b:所有可能性
    for (ll i = 0; i < 6 * 6 * 6 * 6 * 6; ++i)
    {
        for (ll j = 0, k = i; j < 5; ++j, k /= 6)
        {
            c[j] = k % 6 + 1;
        }
        sort(c, c + 5);
        if (c[0] == 2 && c[1] == 3 && c[2] == 4 && c[3] == 5 && c[4] == 6)
        {
            continue;
        }
        ++b;
        ll a1 = 0, a2 = 0; // a1:斋; a2:飞
        for (ll j = 0; j < 5; ++j)
        {
            if (c[j] == x)
            {
                ++a1, ++a2;
            }
```

```
            else if (c[j] == 1)
            {
                ++a2;
            }
        }
        if (a1 == 5)
        {
            ++a1;
        }
        if (a2 == 5)
        {
            ++a2;
        }
        c1[x][a1]++, c2[x][a2]++;
    }
    bp[0] = 1;
    for (ll i = 1; i <= 5; ++i)
    {
        bp[i] = bp[i - 1] * b;
    }
    f1[0][x][0] = 1, f2[0][x][0] = 1; // f^0(x)=1
    //暴力多项式乘法，可以分治FFT优化为O(nlogn)
    for (ll i = 1; i < 5; ++i)
    {
        for (ll j = 0; j < 90; ++j)
        {
            for (ll k = 0; k <= 6; ++k)
            {
                f1[i][x][j + k] += f1[i - 1][x][j] * c1[x][k];
                f2[i][x][j + k] += f2[i - 1][x][j] * c2[x][k];
            }
        }
    }
}
signed main()
{
    for (ll i = 1; i <= 6; ++i)
    {
        init(i);
    }
    ll t, n, r, x, tot;
    for (sc(t); t; --t)
    {
        sc(n), sc(r), sc(x), sc(tot);
        ll numx = 0;
        if (x == 1)
        {
            r = 1;
        }
        for (ll i = 1, v; i <= 5; ++i)
        {
            sc(v);
            numx += v == x || (v == 1 && r == 2);
        }
        if (numx == 5)
        {
            numx++;
        }
```

```
        ll u = max(0LL, tot - numx), a = 0, b = bp[n - 1];
        for (ll i = u; i <= 6 * n; ++i)
        {
            a += r == 1 ? f1[n - 1][x][i] : f2[n - 1][x][i];
        }
        ll c = __gcd(a, b);
        printf("%lld/%lld\n", a / c, b / c);
    }
    return 0;
}
```

Python

```
t = [0 for i in range(5)]  #当前dfs枚举的5个骰子状态
c = [[[0 for i in range(7)] for j in range(7)] for k in range(3)]  #斋/飞x总数
f = [[[[0 for h in range(40)] for i in range(7)] for j in range(5)]
     for k in range(3)]  #多项式
for x in range(1, 7):
    for i in range(6**5):  #六进制枚举
        for j in range(5):
            t[j] = i // 6**j % 6 + 1
        t.sort()
        if t == [2, 3, 4, 5, 6]:
            continue
        num1 = len(list(filter(lambda y: y == x, t)))  #斋
        num1 += num1 == 5
        c[1][x][num1] += 1
        num2 = len(list(filter(lambda y: y == x or y == 1, t)))  #飞
        num2 += num2 == 5
        c[2][x][num2] += 1
    f[1][0][x][0] = f[2][0][x][0] = 1
    for i in range(1, 5):
        for j in range(32):
            for k in range(7):
                for r in range(1, 3):
                    f[r][i][x][j + k] += f[r][i - 1][x][j] * c[r][x][k]
for _ in range(int(input())):
    n, r, x, tot = [int(i) for i in input().split()]
    r = 1 if x == 1 else r
    a = [int(i) for i in input().split()]
    num = len(list(filter(lambda y: y == x or (y == 1 and r == 2), a)))
    num += num == 5
    tot = max(0, tot - num)
    fz, fm = 0, 7656**(n - 1)
    for i in range(tot, 6 * n + 1):
        fz += f[r][n - 1][x][i]

    def gcd(a, b):
        return gcd(b, a % b) if b > 0 else a

    g = gcd(fz, fm)
    print('%d/%d' % (fz // g, fm // g))
```

J. 疲惫的Serein

题解

我们有 n 个非负整数（题面中 $a[]$ 是 $\frac{n(n-1)}{2}$ 个数的和，这里为了方便就姑且设为 $a[1\sim n]$ ），答案要求又从小到大输出，我们就从小到大求解($a[1]\le a[2]\le a[3]\le a[4].....$)

这就有了以下数对和

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}a[i]+a[j]$$

这就是输入，我们把它们装在数组 $sum[]$ 里

展开上式可以得到

$$a[1]+a[2]\ ,a[1]+a[3]\ ,a[1]+a[4]\ ,a[1]+a[5]\ \dots\ a[1]+a[n]$$

$$a[2]+a[3]\ ,a[2]+a[4]\ ,a[2]+a[5]\ \dots\ a[2]+a[n]$$

$$.....$$

$$a[n-1]+a[n]$$

这个略丑， 我们稍加整理

$$\begin{array}{ccccccc} a[1]+a[2] & a[1]+a[3] & a[1]+a[4] & a[1]+a[5] & \dots & a[1]+a[n] \\ & a[2]+a[3] & a[2]+a[4] & a[2]+a[5] & \dots & a[2]+a[n] \\ & & a[3]+a[4] & a[3]+a[5] & \dots & a[3]+a[n] \\ & & & a[4]+a[5] & \dots & a[4]+a[n] \\ & & & & \dots & \\ & & & & & a[n-1]+a[n] \end{array}$$

这样子我们可以得到一个倒三角形，

且发现每一行，每一列都有一个相同的元素，又由于我们求的 $a[]$ 单调递增（单调不下降），可知:

每一行，从左到右，值依次递增

每一列，从上到下，值依次递增

~~元素周期表的感觉~~

可以进一步推出当前图最小的那个点一定在左上角（突破口）

意味着将 $sum[]$ 排个序后，我们知道了最小的 $sum(sum[1])$ ，这其实就是 $a[1]+a[2]$

观察一下可以发现，$a[1\sim n]$ 是递增的 n 个数，所以可以得到 $sum[1]=a1+a2$，$sum[2]=a1+a3$，那么只要找出 a_2+a_3，就可以求出 a_1 了

如何找出 a_2+a_3 呢，显然它不一定是 $sum[3]$. 但是我们可以遍历 $sum[3]$ 直到 $sum[n(n-1)/2]$，假设这个 sum 是 a_2+a_3 ，从而求解出 a_1

那么假设我们已经求出了 a_1，有什么用? 我们可以求出 a_2，因为 $sum[1]$ 是 a_1 和 a_2 的和, 我们在所有和中剔除已知数两两之和 (a_1, a_2)，剩下的最小 sum 就是 $a_1 + a_3$, 那么可求出 a_3，再利用 a_3 将可以求出的 sum 剔除掉, 剩下的 sum 的最小就是 $a_1 + a_4$, 同理, 利用这个规则可以求解出所有的 a

同时又因为题目数据保证如果有解仅有唯一解，所以当遇到满足条件的解直接输出即可。

参考代码

C++

```cpp
#include <bits/stdc++.h>
using namespace std;
const int mn=110;
int a[mn],sum[mn*(mn-1)/2],n,k;
bool vis[mn*(mn-1)/2],flag;
int main()
{
    cin>>n;
    int m=n*(n-1)/2;
    for(int i=1; i<=m; i++) cin>>sum[i];
    for(int i=3; i<=m; i++)
    {
        a[2]=(sum[1]-sum[2]+sum[i])/2;
        a[1]=sum[1]-a[2];
        a[3]=sum[2]-a[1];
        if(a[2]+a[3]!=sum[i]) continue;
        memset(vis,0,sizeof(vis));
        vis[i]=1;
        flag=1;
        k=3;
        for(int j=4; j<=n; j++)
        {
            while(vis[k]) k++;
            a[j]=sum[k]-a[1];
            vis[k]=1;
            for(int g=2; g<j; g++)
            {
                bool f=0;
                for(int x=k+1; x<=m; x++)
                {
                    if(!vis[x]&&a[j]+a[g]==sum[x])
                    {
                        vis[x]=1;
                        f=1;
                        break;
                    }
                }
                if(!f)
                {
                    flag=0;
                    break;
                }
            }
            if(!flag) break;
        }
```

```
        if(flag) break;
    }
    if(flag)
    {
        for(int i=1; i<=n; i++)
        {
            if(i!=1) cout<<" ";
            cout<<a[i];
        }
    }
    else cout<<-1;
    return 0;
}
```

Python

```
n = int(input())
a = [int(i) for i in input().split()]
if sum(a) % (n - 1) != 0:
    print(-1)
    exit(0)
m = n * (n - 1) // 2
for i in range(2, m):
    y = (a[0] - a[1] + a[i]) // 2
    z = a[i] - y
    x = a[0] - y
    if x + z != a[1] or x + y != a[0] or y + z != a[i]:
        continue
    vis = [False for i in range(m)]
    r = [x, y, z]
    vis[0] = vis[1] = vis[i] = True
    ok = True
    vi = 2
    for j in range(3, n):
        while vis[vi]:
            vi += 1
        v = a[vi] - x
        vj = vi
        for u in r:
            while vj < m:
                if not vis[vj] and u + v == a[vj]:
                    vis[vj] = True
                    break
                vj += 1
            if vj >= m:
                ok = False
                break
        r.append(v)
        if not ok:
            break
    if not ok:
        continue
    for j in r:
        print('%d ' % j, end='')
    exit(0)
```

```
print(-1)
```