E1-A题题解

> 作者：余绍函

题目描述

此题与 Hard version 只有数据范围不同，其他均一致。

`hls` 有一天迷上了神奇的概率论，于是他去商店买了三个骰子。这三个不同的骰子的面数分别为 a, b, c。

对于一个有S面的骰子，每个面上的数字是$1, 2, 3, 4, \cdots, S$。每个面（上的数字）出现的概率均等。`hls` 想要求出**所有三个面上的数字的和**中，哪个和的值出现的概率最大。

现在给出每个骰子的面数，需要求出**三个面上的数字的和**出现得最频繁的那个数。如果有很多个和出现的概率相同，那么只需要输出最小的那个。

输入

第一个数为数据组数 $T(1 \leq T \leq 20)$。

接下来 T 行，每行 3 个整数 $a, b, c(1 \leq a, b, c \leq 200)$。

输出

对于每组数据，输出 T 行，代表出现的概率最大的那个值的最小值。

输入样例

```
2
3 2 3
1 4 2
```

输出样例

```
5
4
```

样例解释

当三个骰子的面数分别为 $3, 2, 3$ 时，考虑枚举三个骰子能掷出的全部情况：$(1,1,1)$，$(1,1,2)$，$(1,1,3)$，$(1,2,1)$，$(1,2,2)$，$(1,2,3)$，$(2,1,1)$，$(2,1,2)$，$(2,1,3)$，$(2,2,1)$，$(2,2,2)$，$(2,2,3)$，$(3,1,1)$，$(3,1,2)$，$(3,1,3)$，$(3,2,1)$，$(3,2,2)$，$(3,2,2)$，共计 1818 种情况，全部求和后有：

求和	3	4	5	6	7	8
出现次数	1	3	5	5	3	1

因此出现次数最多的最小的数为 55。第二组数据同理可得。

解题思路

不难看出本题的数据范围非常小，并且时间限制非常宽松。在这种情况下，直接用暴力的方法解题是最简单的。我们可以通过遍历三个骰子的点数，模拟投骰子共$a * b * c$次，求三个骰子的点数之和并记录，在模拟结束后统计点数和出现次数最大且点数和最小的点数和。

方法1：暴力枚举

```
#include<stdio.h>
int t, a, b, c, count[601], maxVal, maxCnt, maxSum;
int main()
{
    scanf("%d", &t);
    while(t--)
    {
        scanf("%d%d%d", &a, &b, &c); // 三个骰子的最大点数
        maxSum = a + b + c; // 点数的最大合
        for(int i = 3; i <= maxSum; i++) // 初始化计数数组
            count[i] = 0;
        for(int i = 1; i <= a; i++)
            for(int j = 1; j <= b; j++)
                for(int k = 1; k <= c; k++)
                    count[i + j + k]++; // 遍历骰子点数，计数每组数据
        maxVal = maxCnt = -1; // 即将找到的点数合和点数合的出现次数
        for(int i = 3; i <= maxSum; i++)
        {
            if(count[i] > maxCnt)
            {
```

```
                maxCnt = count[i];
                maxVal = i;
            }
        }
        printf("%d\n", maxVal);
    }
    return 0;
}
```

方法2：O(1)

暴力枚举已经可以通过这道题了，但是不能通过数据点强测`E1-I`题。这说明这题的步骤可以进一步简化，让我们再仔细想想骰子点数合的分布：

首先考虑两个骰子点数合中出现概率最大的点数和

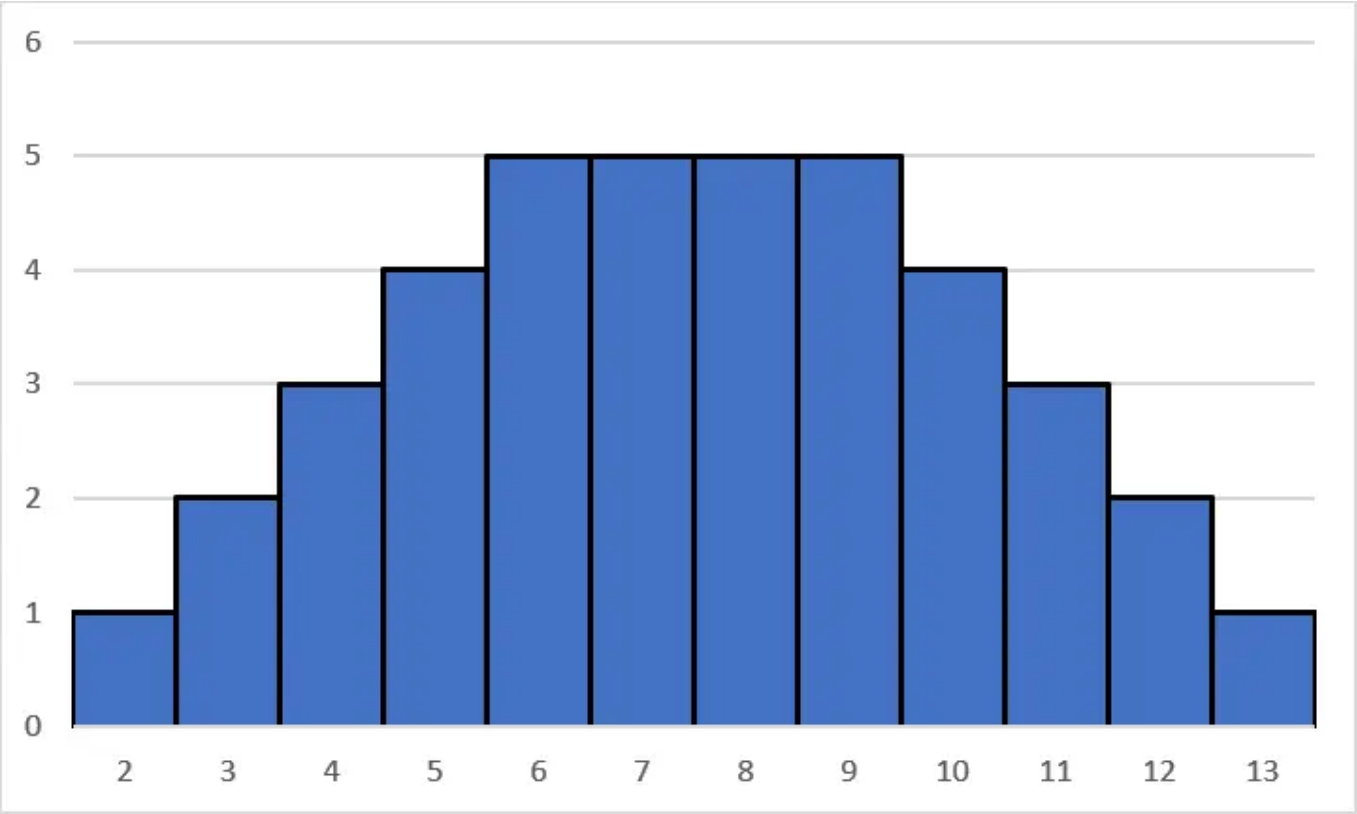
	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9	10
3	4	5	6	7	8	9	10	11
4	5	6	7	8	9	10	11	12
5	6	7	8	9	10	11	12	13

观察上图，不难得出这样的结论：

- 不同的点数之和的情况总数为$a+b-1$，从小到大具体是$2,3,4,\ldots,a+b$;
- 点数之和出现次数最多的是$1+a,1+a+1,\ldots,1+b$，共$b-a+1$个

接下来加入第三个骰子

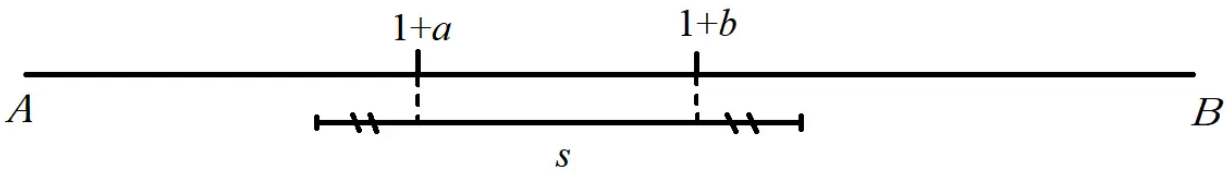
如果直接在上图中加入第三个骰子，那么图像将变为一个长方体，比较难观察思考。因为我们只关心骰子和，而不是骰子的组合，那么上图中的行列坐标显然是多余的，我们先把上图整合为一个柱状图：



设第三个骰子有c面，那么对于上述柱状图的横轴，我们加上$1, 2 \ldots c$就组成了三个骰子的点数之和。那在什么情况下三个骰子的点数之和相等呢？我们可以想象一条长度为c的线段，上面的数从右到左依次为$c, \ldots, 2, 1$。将这个线段水平放在上图横轴上，则横轴上的数与线段上的数的和始终保持一致。由于横坐标对应的频率不都一致，那么当线段尽量靠近中间的时候，所得到的点数之和频率最高。进一步讲，依据上文中的结论，我们应该尽量把线段落在区间$[1+a, 1+b]$内。我们先根据线段的长度进行分类讨论：

- 当线段的长度**不足以完全覆盖或恰好完全覆盖**时，为了保证三点数之和最小，我们需要把线段靠左放置在点$1+a$上。在这种情况下，答案显然是$1+a+c$。
- 当线段的长度**比区间长度更大**时，我们应当使得线段放置在区间中间（尽量靠左）处

<a href="https://cdn.luogu.com.cn/upload/image_hosting/dfh3xcxc.png"target="_blank">



当然$(c-(b-a+1))/2$不一定是整数，所以可以根据尽量靠左的原则向左分配一点。这种情况下答案是$1+b+\lfloor(c-(b-a+1))/2\rfloor+1$，可简化为$2+b+\lfloor(a+c-b-1))/2\rfloor$。

在大功告成之前，我们还应该考虑下a,b,c之间的大小关系。在上文中，我们按照$b>a$的前提条件进行讨论，因此在三个骰子的情况下$b>a$仍然满足。由于在答案$1+a+c$和$1+b+\lfloor(a+c-b-1))/2\rfloor+1$之中，$a$和$c$都只以$a+c$的形式出现，因此两个变量等价。由此可以得到大小关系为$b>a,c$。

代码实现

```
#include<stdio.h>
void swap(int *n1, int *n2)
{
    int tmp = *n1;
    *n1 = *n2;
    *n2 = tmp;
}
int t, a, b, c;
int main()
{
    scanf("%d", &t);
    while(t--)
    {
        scanf("%d%d%d", &a, &b, &c);
        if(a > b)
            swap(&a, &b);
```

```
        if(c > b)
            swap(&c, &b);
        if(b - a + 1 >= c)
            printf("%d\n", 1 +  a + c);
        else
            printf("%d\n", 2 + b + (a + c - b - 1) / 2);
    }
    return 0;
}
```

参考资料

本文大部分参考了洛谷P2911题解，删去了其中的冗余部分并加入了对于a, b, c大小关系的讨论。