SJTU OJ 1379 解题报告

F1503024　515030910585　金陆骅

2017 年 5 月 23 日

目录

1. Description

三国杀中的关羽是一个很厉害的武将，而武圣是个很牛逼的技能。

武圣——你可以将你的任意一张红桃或方片牌当杀使用或打出。

可见武圣这个技能如果用得好那么是可以轻松杀死别人的。但是这样就有了一个问题:他应该先杀死谁后杀死谁呢?

玩三国杀的人都是坐成一圈的,每个人到牌堆的距离相等,也就是说存在一个所有人都在边界上的圆。而现在，关羽想怒杀 4 个反贼。根据他的推理,这一盘的情况应该是这样的: 4 个反贼所在的位置肯定构成一个矩形。现在求这一局可能有多少种反贼的组合。

现在请编程回答这个问题

2. Input Format

第一行包含一个整数 n, 表示除关羽外的游戏人数。

第二行包括 n 个整数, 表示玩家之间的间隔弧长。

3. Output Format

输出共 1 行，表示最多有多少种可能的组合。

4. Solution Method

这题其实表述可能有那么一丢丢不清楚（也可能是我语文太差），反正我用的理解 AC 了。

首先有两点需要注意，第一，如果能够在圆上形成矩形，那么矩形对角线必然为两条直径，也就是对角线上的人应该相隔半个圆周长；第二，如果有 n 对相隔半个圆周长的选手，那么不必模拟，直接通过 nC2，即 n * (n – 1) / 2 即可得到可能的组合数。

那么来讲一下思路，以样例中的数据为例，8 位选手的间隔为：

1　2　2　3　1　1　3　3

那么他们的位置可表示为：

1　3　5　8　9　10　13　16

以最后一名选手的位置为圆周长，那么有 1　9、5　13、8　16 这三对选手在圆的直径上，因此由 3C2 = 3 种可能的组合。

另外，还有一个小技巧可以简化算法。当需要找位于直径上的选手时，不需要遍历进行寻找。将所有位置大于圆周长一半的选手位置减去圆周长一半，再重新进行排序，只需进行一次遍历统计有多少对相同位置的选手即可。

5. Source Code

```
#include<iostream>
using namespace std;

void quickSort(int s[], int l, int r)
{
    if (l< r)
    {
        int i = l, j = r, x = s[l];
        while (i < j)
        {
            while (i < j && s[j] >= x)
                j--;
            if (i < j)
                s[i++] = s[j];
            while (i < j && s[i]< x)
                i++;
            if (i < j)
                s[j--] = s[i];
```

```
        }
        s[i] = x;
        quickSort(s, l, i - 1);
        quickSort(s, i + 1, r);
    }
}

int main()
{
    int i,n,tmp,cnt = 0;
    cin >> n;
    int *p = new int[n];
    cin >> p[0];
    for (i = 1;i < n;++i)
    {
        cin >> tmp;
        p[i] = tmp + p[i - 1];
    }

    int half = p[n - 1] / 2;  //得到半圆周长

    i = n - 1;
    while (p[i] > half)   //所有位置大于半圆周长的选手位置减去半圆周长
    {
        p[i] -= half;
        i--;
    }

    quickSort(p,0,n - 1);//重新排序
    for (i = 0;i < n - 1;++i)//统计重复数字
    {
        if (p[i] == p[i + 1])
        {
            cnt++;
            ++i;
        }
    }

    cout << cnt * (cnt - 1) / 2 << endl; //nC2

    return 0;
}
```