CF round 662 div2

A

题意：略

思路：找规律

```
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
    {
        int n;
        cin >> n;
        cout << n / 2 + 1 << endl;
    }
}
```

B

题意：给定一些长度不一的木棒q个操作，每次操作增加或者删除一根木棒，问操作之后的木棒能否围成一个正方形或者长方形。

题解：直接按照长度分类成四根以上的，两根以上的，暴力维护即可。

```
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
#include<map>
#include <stdio.h>
using namespace std;
const int MAX = 1e5 + 200;
int num[MAX];
map <int,int> pic2;
map<int, int> pic4;
int main()
{
```

```
    int n,m;
    cin >> n;
    for (int i = 1; i <= n;i++)
    {
        int tem;
        scanf("%d", &tem);
        num[tem]++;
    }
    for (int i = 1; i <= n;i++)
    {
        if (num[i] >= 4)
            pic4[i] = num[i];
        else if (num[i] >= 2)
            pic2[i] = num[i];
    }
    cin >> m;
    for (int i = 1; i <= m;i++)
    {
        char S[5];
        int tem = 0;
        scanf("%s%d", S, &tem);
        if (S[0] == '+')
        {
            num[tem]++;
            if (num[tem] == 4)
                pic2.erase(tem),pic4[tem] = num[tem];
            else if (num[tem] == 2)
                pic2[tem] = num[tem];
            else if (pic4.count(tem))
                pic4[tem] = num[tem];
            else if (pic2.count(tem))
                pic2[tem] = num[tem];
        }
        else
        {
            num[tem]--;
            if (num[tem] == 3)
                pic4.erase(tem),pic2[tem] = num[tem];
            else if (num[tem] == 1)
                pic2.erase(tem);
            else if (pic4.count(tem))
                pic4[tem] = num[tem];
            else if (pic2.count(tem))
                pic2[tem] = num[tem];
        }
        if (pic4.size() == 0)
                puts("NO");
        else if (pic2.size() >= 2)
        {
```

```
            puts("YES");
        }
        else if (pic2.size() == 1)
        {
            map<int, int>::iterator it;
            int flag = 0;
            for (it = pic4.begin(); it != pic4.end();it++)
            {
                if (it->second >=6)
                {
                    puts("YES");
                    flag = 114;
                    break;
                }
                else if (flag == 1)
                {
                    puts("YES");
                    flag = 114;
                    break;
                }
                else
                    flag = 1;
            }
            if (flag != 114)
                puts("NO");
        }
        else
        {
            map<int, int>::iterator it;
            int flag4 = 0,flag2 = 0;
            for (it = pic4.begin(); it != pic4.end();it++)
            {
                if (it->second >= 8)
                {
                    puts("YES");
                    flag2 = 114;
                    break;
                }
                if (flag4 == 0)
                {
                    flag4 = 1;
                    if (it->second >= 6)
                        flag2++;
                    continue;
                }
                else
                {
                    puts("YES");
                    flag2 = 114;
                    break;
                }
```

```
            }
            if (flag2 != 114)
                puts("NO");
        }
    }
    return 0;
}
```

C

题意：给定一些数字，要求对这些数字重新排列，是的两个相同数字之间的最小间隔最大。

题解：看起来像二分答案，但是二分答案显然不太好处理（可以做，但是不好想而且麻烦），考虑直接贪心，找出出现次数最多的数字，使其均匀分布，然后将其他的数字插入空隙即可。

```
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
const int maxn = 1e5 + 5;
int pic[maxn];

int main()
{
    int t;
    scanf("%d", &t);
    while (t--)
    {
        int n, m = 0, tot = 0;
        scanf("%d", &n);
        for (int i = 1; i <= n; i++)
            pic[i] = 0;
        for (int i = 1; i <= n; i++)
        {
            int x;
            scanf("%d", &x);
            pic[x]++;
            m = max(m, pic[x]);
        }
        for (int i = 1; i <= n; i++)
            if (pic[i] == m)
                tot++;
        printf("%d\n", (n - m - tot + 1) / (m - 1));
    }
    return 0;
```

```
}
```

D

题意：给定一个矩阵，每个格子都有一种颜色，问从中能找到多少个有多少个相同颜色的旋转45度的正方形。

题解：dp□$dp[i][j]$表示以(i,j)为最下面的那个格子一共有多少个斜正方形，所以，只要考虑$(i-2,j),(i-1,j),(i-1,j-1),(i-1,j+1)$这几个格子即可。

```
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <map>
#include <vector>
using namespace std;
typedef long long LL;
const int MAX = 2005;
int dp[MAX][MAX];
char s[MAX][MAX];
LL ans = 0;
int main()
{
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        scanf("%s", s[i] + 1);
    for (int i = 1; i <= 2; i++)
        for (int j = 1; j <= m; j++)
            dp[i][j] = 1;
    for (int i = 3; i <= n; i++)
        for (int j = 1; j <= m; j++)
        {
            if (j == 1 || j == m)
            {
                dp[i][j] = 1;
                continue;
            }
            if (s[i][j] != s[i - 1][j - 1] || s[i][j] != s[i - 1][j] ||
s[i][j] != s[i - 1][j + 1] || s[i][j] != s[i - 2][j])
            {
                dp[i][j] = 1;
                continue;
            }
            int x = 0x3f3f3f3f;
            x = min(min(min(min(x, dp[i - 2][j]), dp[i - 1][j + 1]), dp[i -
1][j - 1]), dp[i - 1][j]) + 1;
```

```
            dp[i][j] = x;
        }
    for (int i = 1; i <= n;i++)
        for (int j = 1; j <= m;j++)
            ans += dp[i][j];
    printf("%lld", ans);

    return 0;
}
```