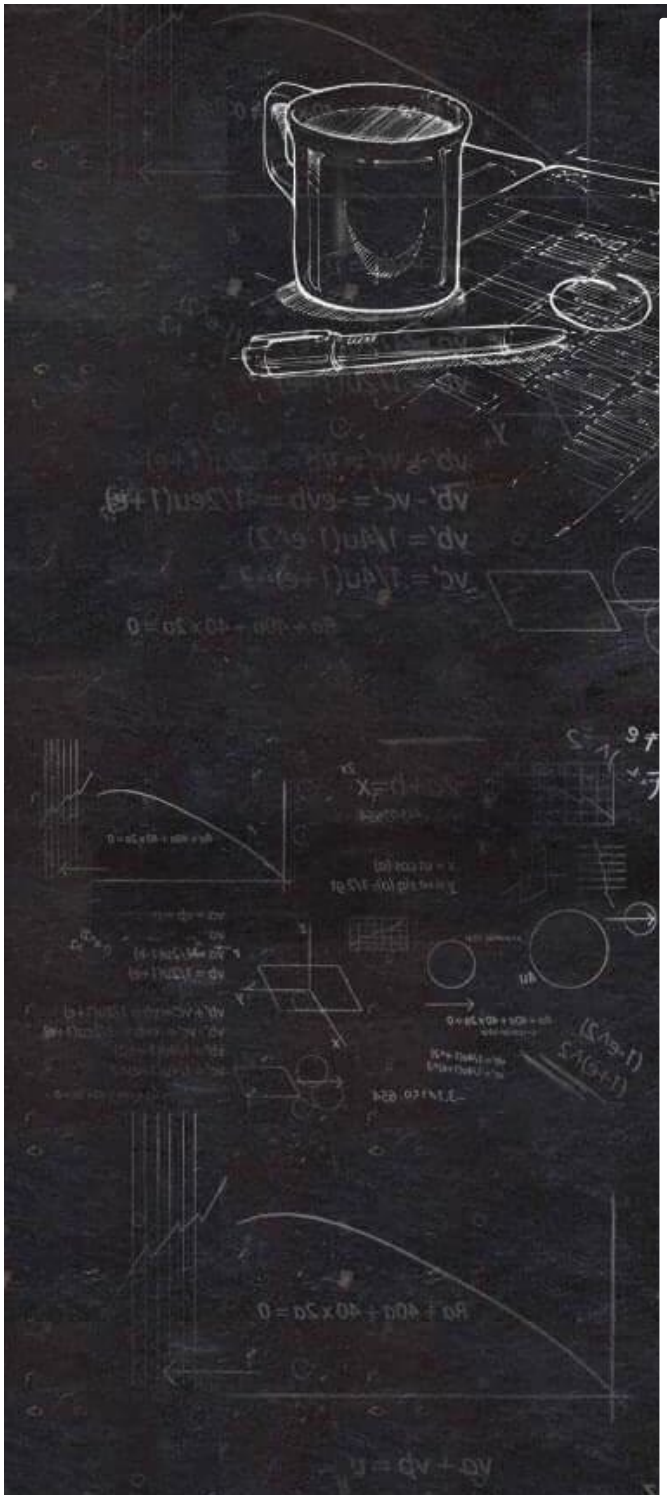


第五届河南省CCPC河南省省赛题解+复盘

第五届河南省CCPC河南省省赛题解+复盘

今年省赛相当有意思的一点，是20级第一次线下省赛，对于部分队也可能是最后一次，看队名就能看出来很多 考研　就业的选手，一群老年人在这PK，氛围挺不错。

A - 小水獭游河南 — 签到

这个题关键点就是 知道a串最多有26个字母，超过26个字母一定会重复
同时注意，一旦发生重复后边也一定会重复，记得break
还要要特判一下s长度为1的情况

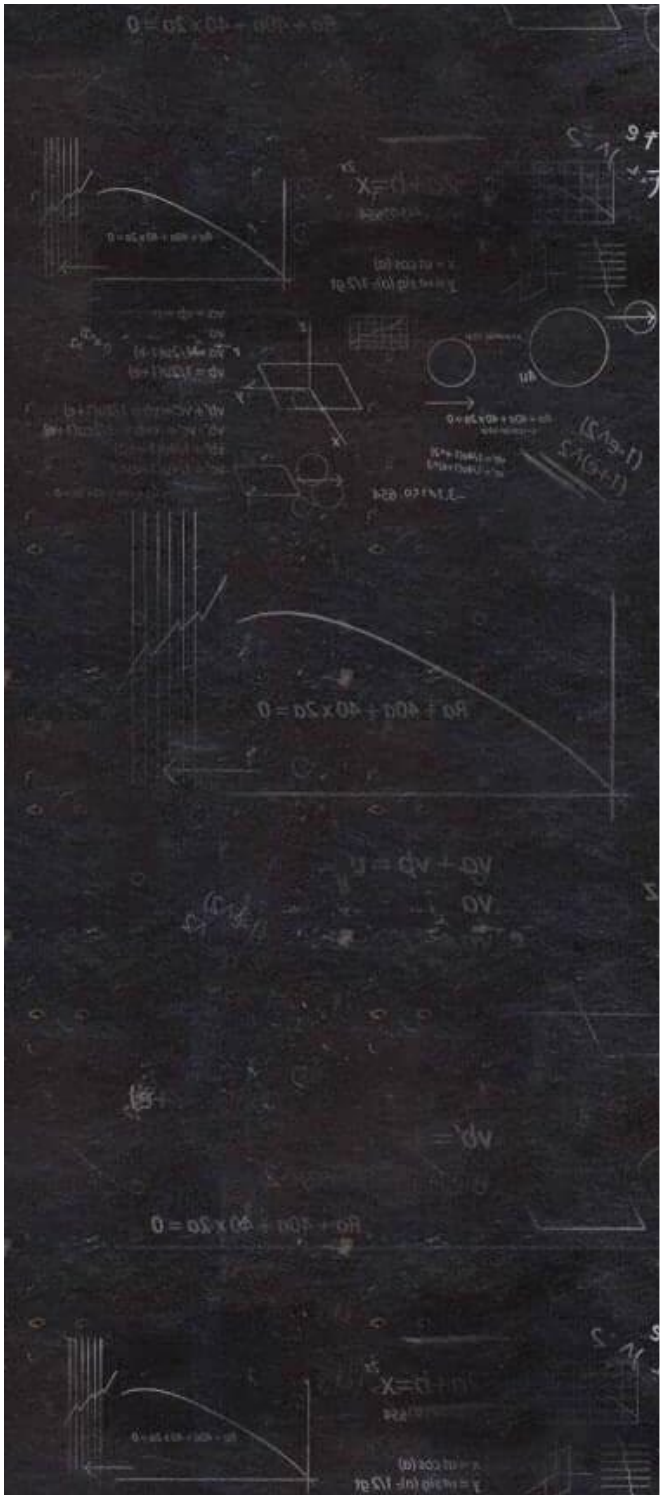
```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;

// 判断回文
bool check(string s, int st)
{
    for(int i = st, j = s.size() - 1; i < j; i ++, j --)
        if(s[i] != s[j]) return false;
    return true;
}

void solve()
{
    string s; cin >> s;
    vector<int> cnt(26, 0);

    if(s.size() == 1) {
        cout << "NaN\n";
        return ;
```

```
    }

    // 最多26个字母，st代表后边回文字符串的开始位置
    for(int st = 1; st <= 26; st ++)
    {
        // 之前的计数
        int t = s[st - 1] - 'a';
        // 判断是否重复，  重复直接break
        if(cnt[t])  {
            break;
        }
        cnt[t] ++;

        // 检查是否回文
        if(check(s, st))
        {
            cout << "HE\n";
            return;
        }
    }
    cout << "NaN\n";
}

int main()
{
    int T; cin >> T;
    while(T --) solve();
}

```

B - Art for Rest – 找性质+前缀/后缀处理

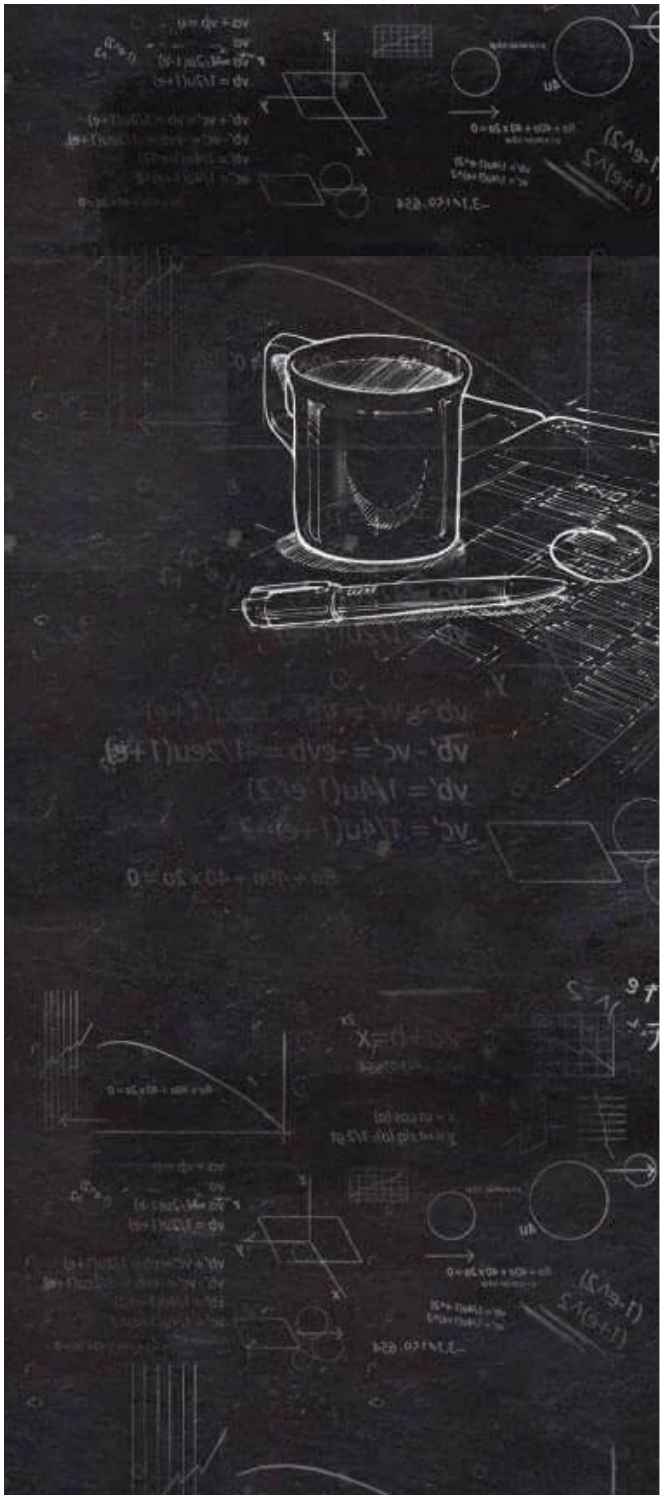
原题意是 把一个长度为n的数组，分为每段长度为k的区间（最后一个区间可能不足k），每个区间单独排序，区间排序后拼接形成整个数组。求满足拼接形成的数组是非严格递增的 k 的 数量

题目可转化为 求符合 每一段长度为k 的区间的最大值 小于等于 后缀数组 所有数的最小值的 k的取值数量
又可以转化为 求符合 每个区间最后一个位点 前缀最大值 小于等于 后缀最小值的 k的取值数量



直接暴力即可，调和级数复杂度nlogn

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;

int main()
{
    int n; cin >> n;
    vector<int> a(n + 1), minv(n + 2), st(n + 2, 0);

    for(int i = 1; i <= n; i ++) scanf("%d", &a[i]);

    // 求后缀最小值
    minv[n + 1] = 2e9;
    for(int i = n; i >= 1; i --)
        minv[i] = min(minv[i + 1], a[i]);

    // st[i] = 1 表示前i个数里的最大值小于 后缀所有数的最小值
    int maxv = 0;
    for(int i = 1; i <= n; i ++)
    {
        maxv = max(maxv, a[i]);
        if(maxv <= minv[i + 1] || i == n)
            st[i] = 1;
    }


    int res = 0;
    // 对于每一个长度为k的区间，看最后一个位置 st[i] 全部等于1 即满足条件，答案 + 1
    for(int k = 1; k <= n; k ++)
    {
        int f = 1;
        for(int i = k; i <= n; i += k)
        {
            if(st[i] == 0)
            {
```



```
                f = 0;
                break;
            }
        }
        res += f;
    }
    printf("%d\n", res);
}
```

C - Toxel 与随机数生成器 — 思维题

因为|s| = 1e6 每段长度1e3到1e4，直接暴力如果重复出现100次以上就判为No

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;

int main()
{
    string s; cin >> s;
    // 记录重复数量
    int cnt = 0;
    string tar = s.substr(0, 1000);
    for(int i = 0; i + 1000 < s.size(); i ++)
    {
        if(tar == s.substr(i, 1000)){
            cnt ++;
            i += 1000;
        }
    }
    if(cnt < 100) cout << "Yes\n";
    else cout << "No\n";
}
```

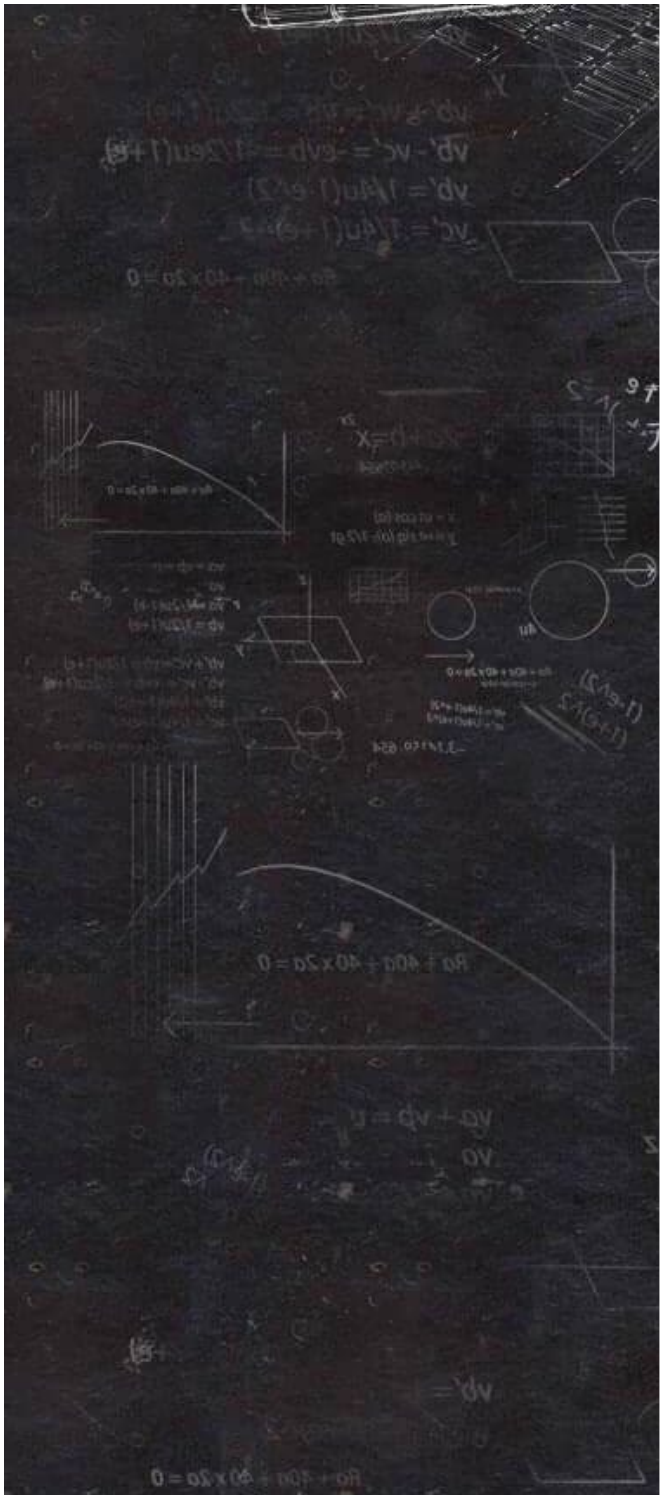
E - 矩阵游戏 — dp

官方题解说的很清楚，不过需要注意的是三维转二维的过程，很容易出现错误，用一个新的数组记录上一个状态

考虑 $f_{i,j,k}$ 表示从 $(1,1)$ 开始走到 (i,j) 恰好替换了 k 个？最多获得的分数，容易得到转移方程：

$$f_{i,j,k}=\begin{cases}\max(f_{i,j-1,k},f_{i-1,j,k}) & s_{i,j}=0\\ \max(f_{i,j-1,k},f_{i-1,j,k})+1 & s_{i,j}=1\\ \max(f_{i,j-1,k},f_{i-1,j,k},f_{i-1,j,k-1}+1,f_{i,j-1,k-1}+1) & k\neq 0\wedge s_{i,j}=?\\ \max(f_{i,j-1,k},f_{i-1,j,k}) & k=0\wedge s_{i,j}=?\end{cases}$$


```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
using namespace std;

void solve()
{
    int n, m, x; cin >> n >> m >> x;
    vector<vector<int>> f(m + 1, vector<int>(x + 1)), g(f);
    vector<string> s(n + 1);
    for (int i = 1; i <= n; i++) cin >> s[i], s[i] = " " + s[i];

    int res = 0;
    // g 对应的其实就是 f[i-1][j][k] f对应 f[i][j][k]
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= m; j++)
        {
            for (int k = 0; k <= x; k++)
            {
                if (s[i][j] == '0') f[j][k] = max(f[j - 1][k], g[j][k]);
                else if (s[i][j] == '1') f[j][k] = max(f[j - 1][k], g[j][k]) + 1;
```



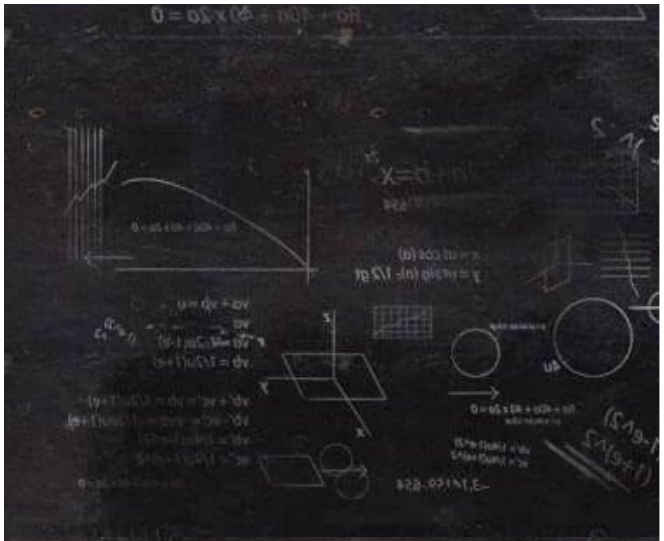
```
                else
                {
                    if (k >= 1) f[j][k] = max(f[j - 1][k - 1] + 1, g[j][k - 1] + 1)
;
                    else f[j][k] = max(f[j - 1][k], g[j][k]);
                }
                if (i == n && j == m) res = max(res, f[j][k]);
            }
        }
        g = f;
    }

    cout << res << "\n";
}

int main()
{
    int T; cin >> T;
    while(T --) solve();
}
```

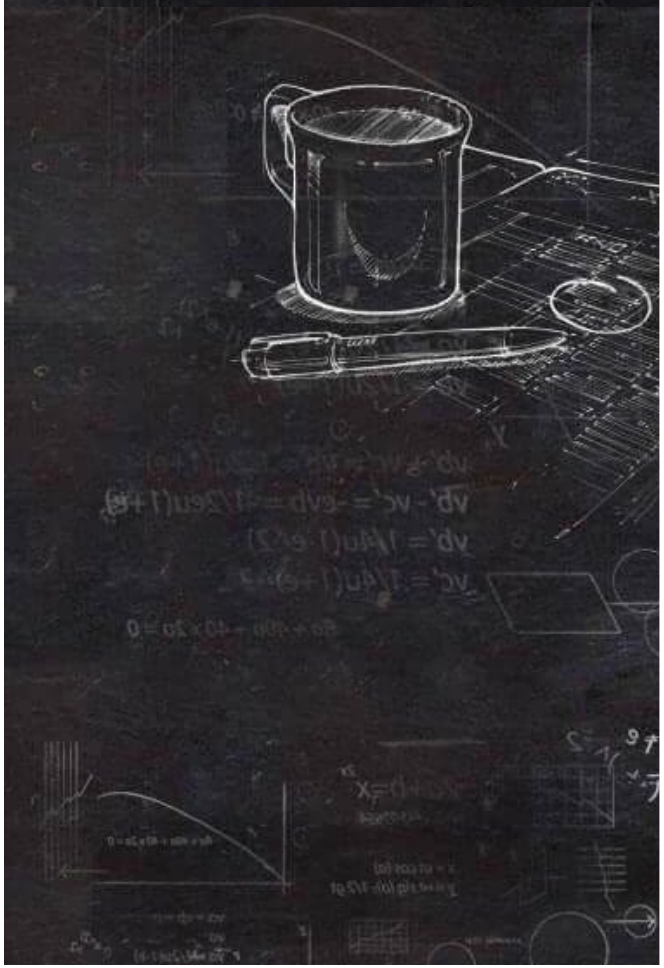
F - Art for Last — 贪心 + 区间求最小值（下边三种写法均可）

题意是求 从n个数中 选k个数 求k个数中（任意2个数之差的最小值）乘 （任意2数的之差最大值） 的最小值
我们要想求最小值，就是尽量让2个数之差的最小值尽可能的小，让任意2数的之差最大值也尽可能的小
直接贪心排序一下，最大值就是长度为k的区间 最后一个-第一个。
然后就是找区间中两个数差值的最小值，差值的最小值一定在排序后相邻的元素中产生
因此找相邻元素差值最小值即可，把相邻元素差值单独取出来，用ST表或线段树或滑动窗口维护都可以，求一下区间最小值即可

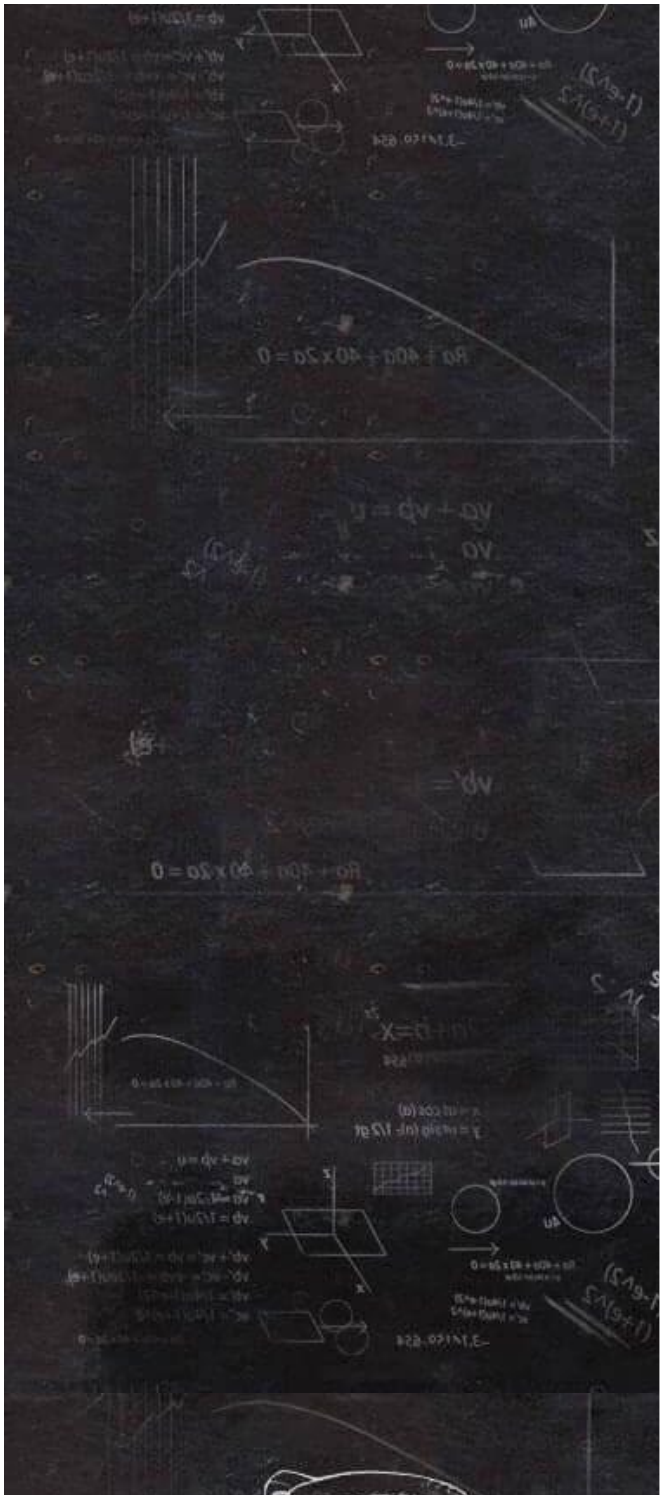
所以sort后遍历一下求每个长度k区间的最大最小值，取乘积最小即可

写法一 ST表写法





```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
using namespace std;
#define int long long
const int N = 500010;
int Fmin[N][20];
int lg2[N];
int a[N];
int n, k;
void init_log()
{
    lg2[0] = -1;
    for(int i = 1; i < N; i ++) lg2[i] = lg2[i >> 1] + 1;
}
void init()
{
    for(int i = 1; i < n; i ++) Fmin[i][0] = a[i + 1] - a[i];
    int k = lg2[n];
    for(int j = 1; j <= k; j ++)
        for(int i = 1; i <= n - (1 << j) + 1; i ++)
        {
            Fmin[i][j] = min(Fmin[i][j - 1], Fmin[i + (1 << j - 1)][j - 1]);
        }

}
int RMQ(int l,int r)
{
    int k = lg2[r - l + 1];
    int minv = min(Fmin[l][k], Fmin[r - (1 << k) + 1][k]);
    return minv;
}

signed main()
{
    init_log();

    scanf("%lld %lld",&n, &k);
```



```
    for(int i = 1; i <= n; i ++) scanf("%lld", a + i);
    sort(a + 1, a + n + 1);

    init();
    int res = 2e18;
    for(int i = k; i <= n; i ++)
    {
        int t = RMQ(i - k + 1, i - 1) * (a[i] - a[i - k + 1]);
        res = min(t, res);
    }
    cout << res << '\n';
}

```

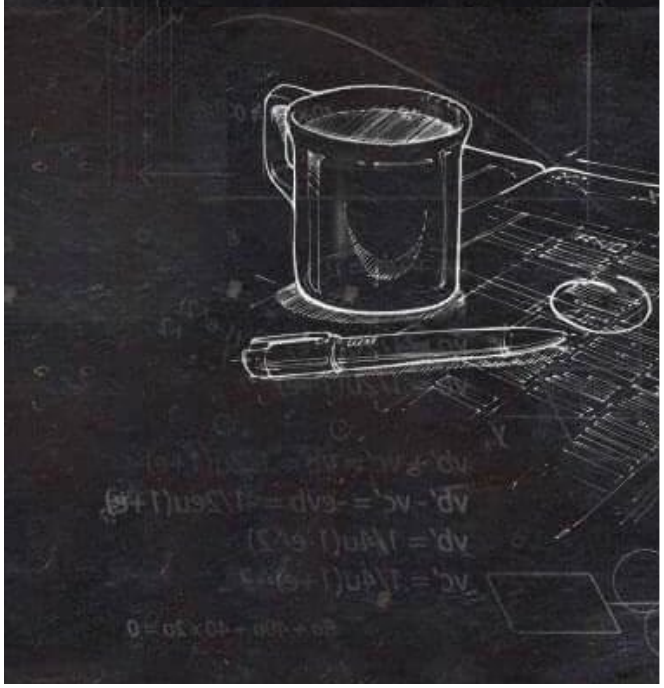
写法二 滑动窗口

```
#include <iostream>
#include <cstring>
#include <algorithm>
#define int long long
using namespace std;

signed main()
{
    int n, k; cin >> n >> k;
    vector<int> a(n + 1), d(n + 1), q(n + 1);
    for(int i = 1; i <= n; i ++) cin >> a[i];
    sort(a.begin() + 1, a.end());

    for(int i = 1; i < n; i ++)
        d[i] = a[i + 1] - a[i];

    int hh = 0, tt = -1;
    int res = 2e18;
    for(int i = 1; i <= n; i ++)
    {
        if(hh <= tt && q[hh] < i - k + 1) hh ++;
        if(i >= k)
        {
```

```
23
24              int t = d[q[hh]] * (a[i] - a[i - k + 1]);
25              res = min(res, t);
26          }
27          while(hh <= tt && d[i] <= d[q[tt]]) tt --;
28          q[++ tt] = i;
29      }
30      cout << res << '\n';
31  }
```

写法三 multiset写法

```
1   #include <iostream>
2   #include <cstring>
3   #include <algorithm>
4   #include <set>
5   #define int long long
6   using namespace std;
7
8   signed main()
9   {
10      ios::sync_with_stdio(0);
11      cin.tie(0);cout.tie(0);
12      int n, k; cin >> n >> k;
13      vector<int> a(n + 1);
14      for(int i = 1; i <= n; i ++) cin >> a[i];
15      sort(a.begin() + 1, a.end());
16
17      multiset<int> S;
18      for(int i = 1; i < k; i ++)
19          S.insert(a[i + 1] - a[i]);
20
21      int res = *S.begin() * (a[k] - a[1]);
22      for(int i = k; i <= n; i ++)
23      {
24          int t = *S.begin() * (a[i] - a[i - k + 1]);
25          res = min(res, t);
26          S.erase(S.find(a[i - k + 2] - a[i - k + 1]));
            S.insert(a[i + 1] - a[i]);
```

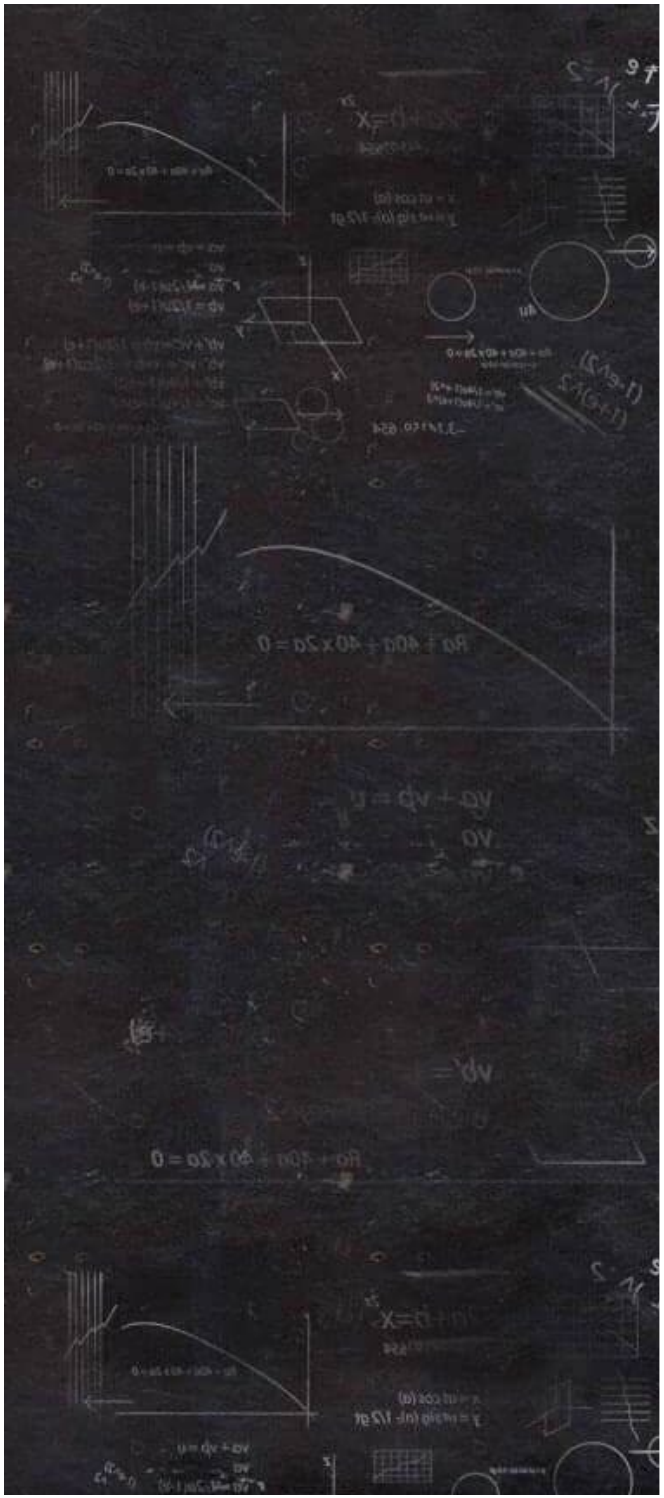


```
    }
    cout << res << '\n';
}
```

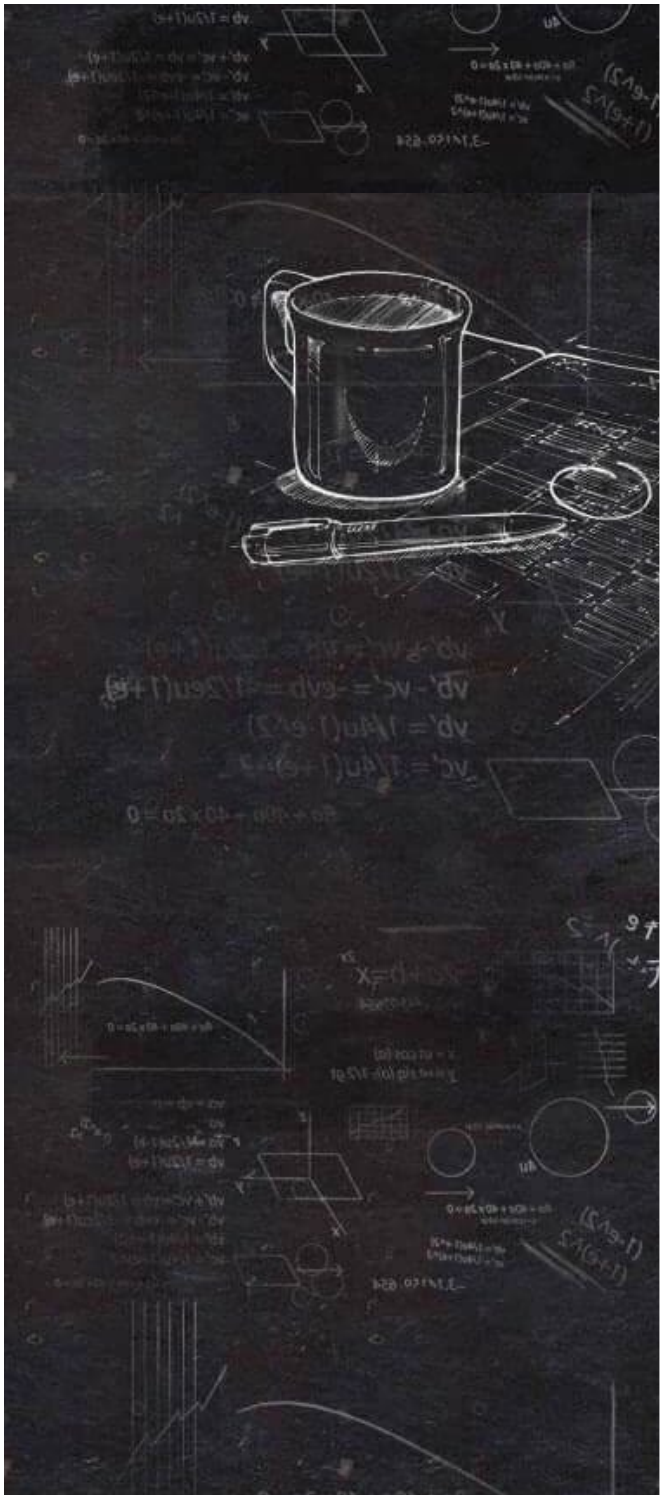
G - Toxel 与字符画 — 大模拟

大模拟，容易出错的地方是判断结果是否超过1e18输出INF
这里不需要用什么快速幂，特判一下如果x = 1结果为1
否则只要x是比2大的数，2的60次方超过1e18，因此遍历不会超过60次
直接暴力判断即可，需要注意暴力乘的时候可能爆longlong，直接用__int128就好了

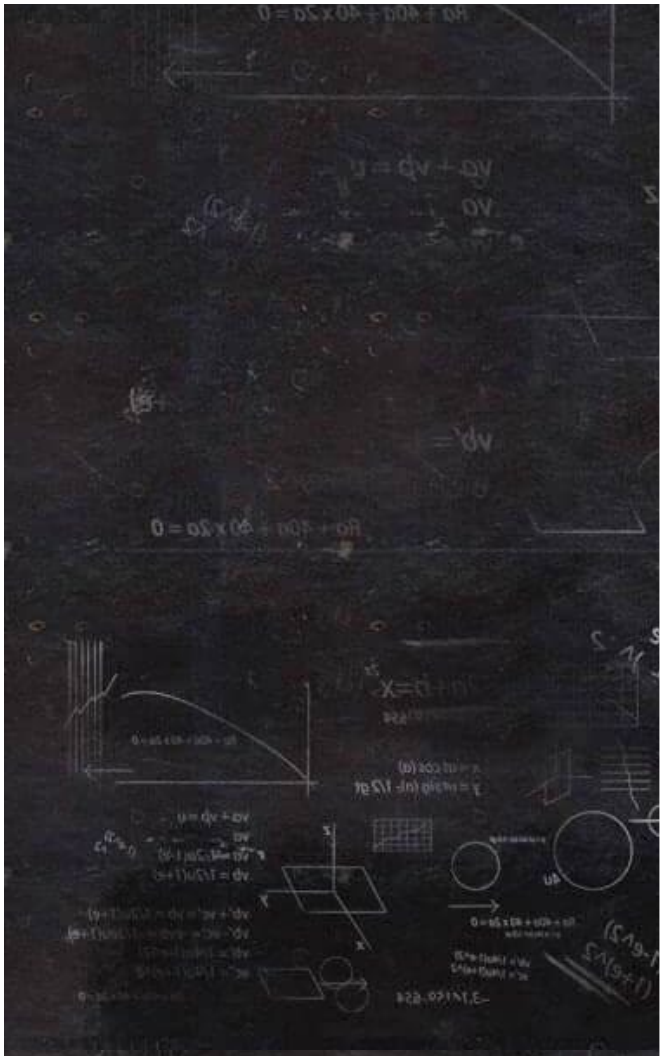
```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
using namespace std;
#define int long long
string big[11][10], sma[11][10];
string dengyu[10], inf[40];
char g[15][2010];
int col = 0;
void init()
{
    big[0][0]=".......",sma[0][0]="......";
    big[0][1]=".......",sma[0][1]=".00000";
    big[0][2]=".0000000",sma[0][2]=".0...0";
    big[0][3]=".0.....0",sma[0][3]=".0...0";
    big[0][4]=".0.....0",sma[0][4]=".0...0";
    big[0][5]=".0.....0",sma[0][5]=".00000";
    big[0][6]=".0.....0",sma[0][6]="......";
    big[0][7]=".0.....0",sma[0][7]="......";
    big[0][8]=".0000000",sma[0][8]="......";
    big[0][9]=".......",sma[0][9]="......";

    big[1][0]=".......",sma[1][0]="......";
    big[1][1]=".......",sma[1][1]=".....1";
    big[1][2]=".......1",sma[1][2]=".....1";
```



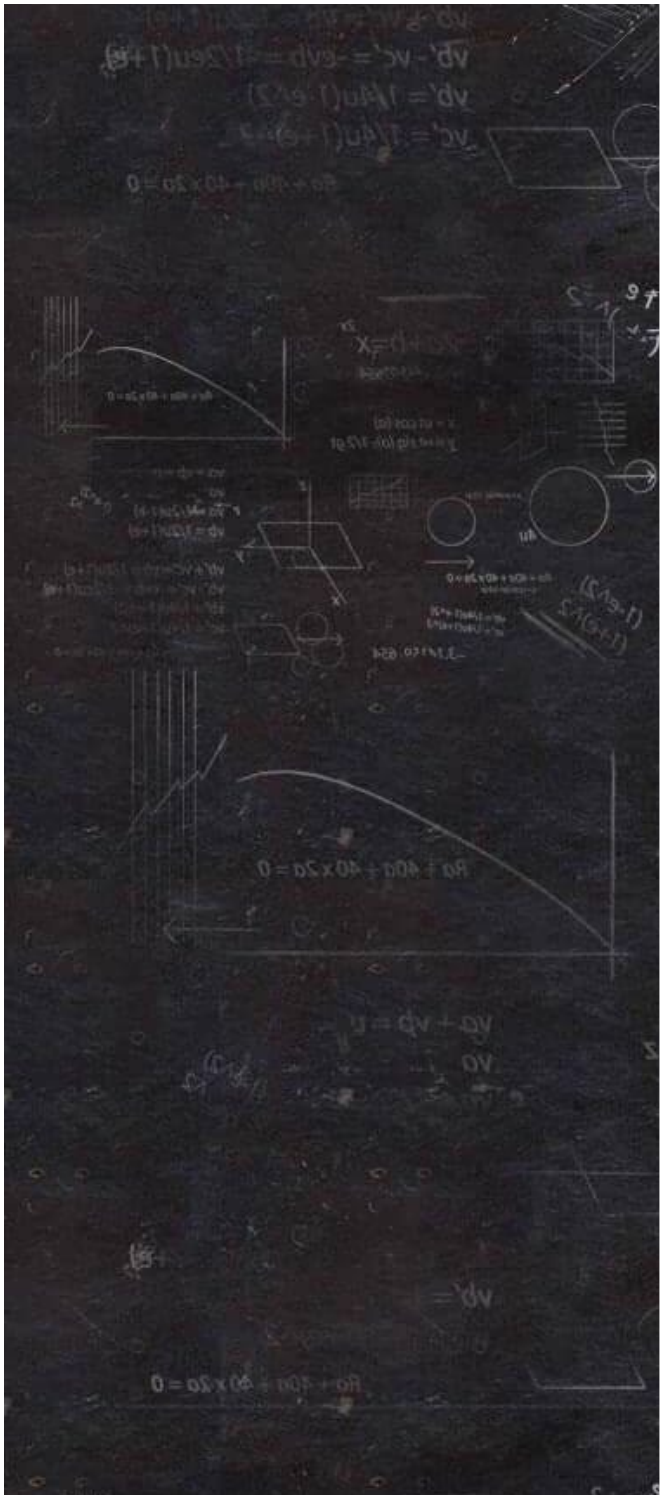
```
27      big[1][3]=".......1",sma[1][3]=".....1";
28      big[1][4]=".......1",sma[1][4]=".....1";
29      big[1][5]=".......1",sma[1][5]=".....1";
30      big[1][6]=".......1",sma[1][6]="......";
31      big[1][7]=".......1",sma[1][7]="......";
32      big[1][8]=".......1",sma[1][8]="......";
33      big[1][9]="........",sma[1][9]="......";
34
35      big[2][0]="........",sma[2][0]="......";
36      big[2][1]="........",sma[2][1]=".22222";
37      big[2][2]=".2222222",sma[2][2]=".....2";
38      big[2][3]=".......2",sma[2][3]=".22222";
39      big[2][4]=".......2",sma[2][4]=".2....";
40      big[2][5]=".2222222",sma[2][5]=".22222";
41      big[2][6]=".2......",sma[2][6]="......";
42      big[2][7]=".2......",sma[2][7]="......";
43      big[2][8]=".2222222",sma[2][8]="......";
44      big[2][9]="........",sma[2][9]="......";
45
46      big[3][0]="........",sma[3][0]="......";
47      big[3][1]="........",sma[3][1]=".33333";
48      big[3][2]=".3333333",sma[3][2]=".....3";
49      big[3][3]=".......3",sma[3][3]=".33333";
50      big[3][4]=".......3",sma[3][4]=".....3";
51      big[3][5]=".3333333",sma[3][5]=".33333";
52      big[3][6]=".......3",sma[3][6]="......";
53      big[3][7]=".......3",sma[3][7]="......";
54      big[3][8]=".3333333",sma[3][8]="......";
55      big[3][9]="........",sma[3][9]="......";
56
57      big[4][0]="........",sma[4][0]="......";
58      big[4][1]="........",sma[4][1]=".4...4";
59      big[4][2]=".4.....4",sma[4][2]=".4...4";
60      big[4][3]=".4.....4",sma[4][3]=".44444";
61      big[4][4]=".4.....4",sma[4][4]=".....4";
62      big[4][5]=".4444444",sma[4][5]=".....4";
63      big[4][6]=".......4",sma[4][6]="......";
64      big[4][7]=".......4",sma[4][7]="......";
65      big[4][8]=".......4",sma[4][8]="......";
```



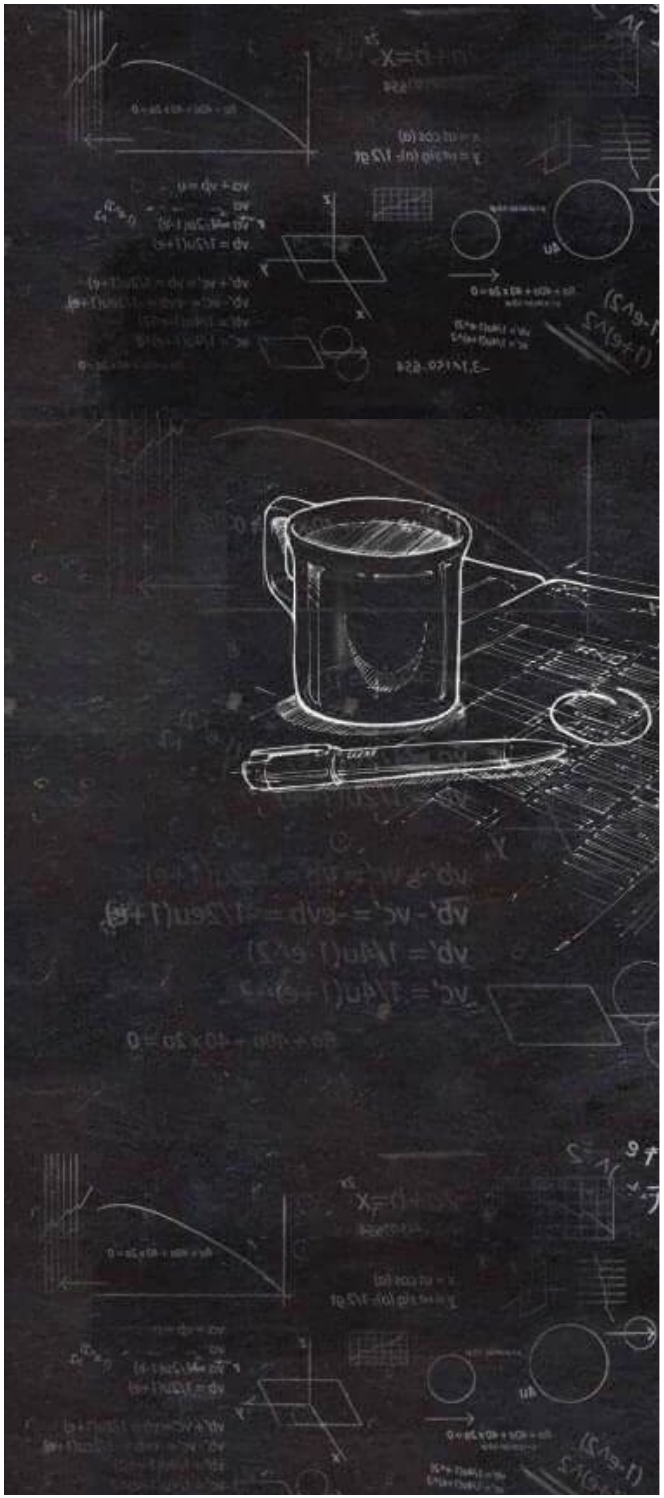
```
66      big[4][9]=".......",sma[4][9]="......";
67
68      big[5][0]=".......",sma[5][0]="......";
69      big[5][1]=".......",sma[5][1]=".55555";
70      big[5][2]=".5555555",sma[5][2]=".5....";
71      big[5][3]=".5......",sma[5][3]=".55555";
72      big[5][4]=".5......",sma[5][4]=".....5";
73      big[5][5]=".5555555",sma[5][5]=".55555";
74      big[5][6]=".......5",sma[5][6]="......";
75      big[5][7]=".......5",sma[5][7]="......";
76      big[5][8]=".5555555",sma[5][8]="......";
77      big[5][9]=".......",sma[5][9]="......";
78
79      big[6][0]=".......",sma[6][0]="......";
80      big[6][1]=".......",sma[6][1]=".66666";
81      big[6][2]=".6666666",sma[6][2]=".6....";
82      big[6][3]=".6......",sma[6][3]=".66666";
83      big[6][4]=".6......",sma[6][4]=".6...6";
84      big[6][5]=".6666666",sma[6][5]=".66666";
85      big[6][6]=".6.....6",sma[6][6]="......";
86      big[6][7]=".6.....6",sma[6][7]="......";
87      big[6][8]=".6666666",sma[6][8]="......";
88      big[6][9]=".......",sma[6][9]="......";
89
90      big[7][0]=".......",sma[7][0]="......";
91      big[7][1]=".......",sma[7][1]=".77777";
92      big[7][2]=".7777777",sma[7][2]=".....7";
93      big[7][3]=".......7",sma[7][3]=".....7";
94      big[7][4]=".......7",sma[7][4]=".....7";
95      big[7][5]=".......7",sma[7][5]=".....7";
96      big[7][6]=".......7",sma[7][6]="......";
97      big[7][7]=".......7",sma[7][7]="......";
98      big[7][8]=".......7",sma[7][8]="......";
99      big[7][9]=".......",sma[7][9]="......";
100
101
102     big[8][0]=".......",sma[8][0]="......";
103     big[8][1]=".......",sma[8][1]=".88888";
104     big[8][2]=".8888888",sma[8][2]=".8...8";
```



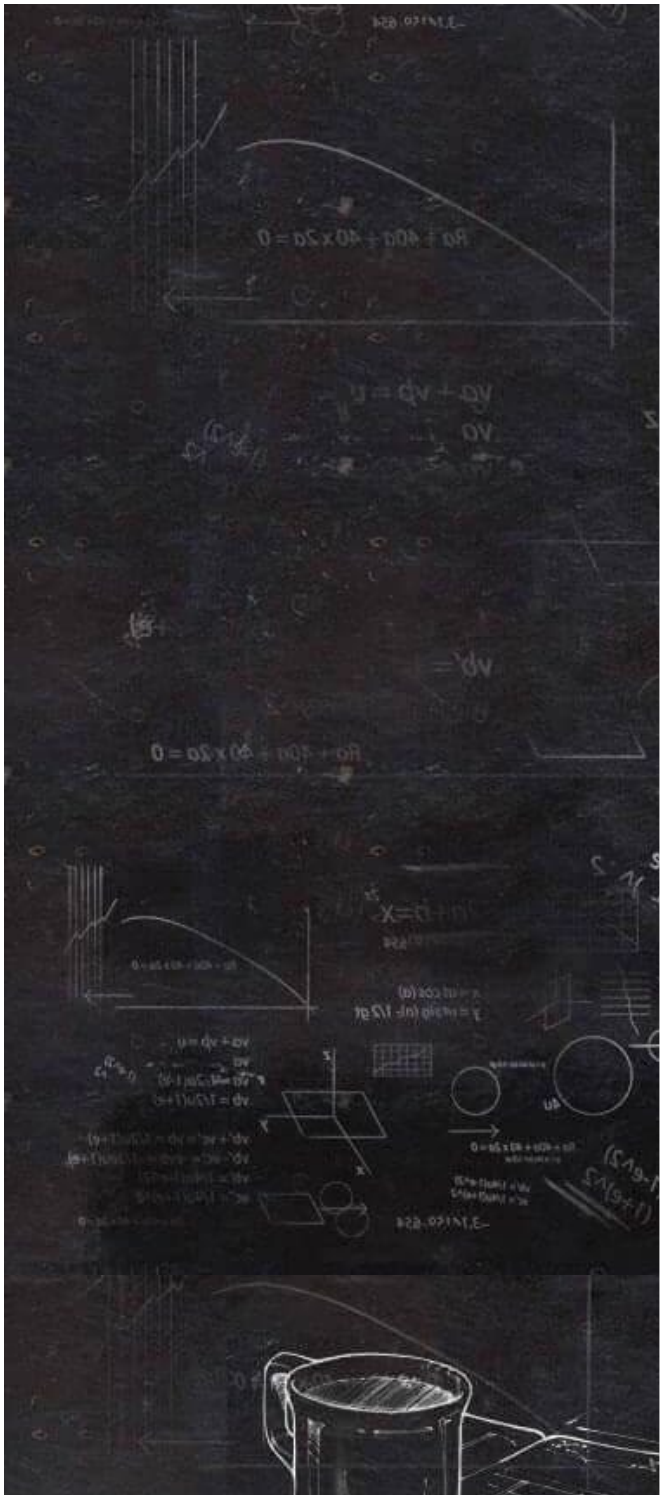
```
105     big[8][3]=".8.....8",sma[8][3]=".88888";
106     big[8][4]=".8.....8",sma[8][4]=".8...8";
107     big[8][5]=".8888888",sma[8][5]=".88888";
108     big[8][6]=".8.....8",sma[8][6]="......";
109     big[8][7]=".8.....8",sma[8][7]="......";
110     big[8][8]=".8888888",sma[8][8]="......";
111     big[8][9]=".......",sma[8][9]="......";
112
113     big[9][0]=".......",sma[9][0]="......";
114     big[9][1]=".......",sma[9][1]=".99999";
115     big[9][2]=".9999999",sma[9][2]=".9...9";
116     big[9][3]=".9.....9",sma[9][3]=".99999";
117     big[9][4]=".9.....9",sma[9][4]=".....9";
118     big[9][5]=".9999999",sma[9][5]=".99999";
119     big[9][6]=".......9",sma[9][6]="......";
120     big[9][7]=".......9",sma[9][7]="......";
121     big[9][8]=".9999999",sma[9][8]="......";
122     big[9][9]=".......",sma[9][9]="......";
123
124     dengyu[0]=".......";
125     dengyu[1]=".......";
126     dengyu[2]=".......";
127     dengyu[3]=".......";
128     dengyu[4]=".=======";
129     dengyu[5]=".......";
130     dengyu[6]=".=======";
131     dengyu[7]=".......";
132     dengyu[8]=".......";
133     dengyu[9]=".......";
134
135     inf[0]="......................";
136     inf[1]="......................";
137     inf[2]=".IIIIIII.N.....N.FFFFFFF";
138     inf[3]="....I....NN....N.F......";
139     inf[4]="....I....N.N...N.F......";
140     inf[5]="....I....N..N..N.FFFFFFF";
141     inf[6]="....I....N...N.N.F......";
142     inf[7]="....I....N....NN.F......";
143     inf[8]=".IIIIIII.N.....N.F......";
```



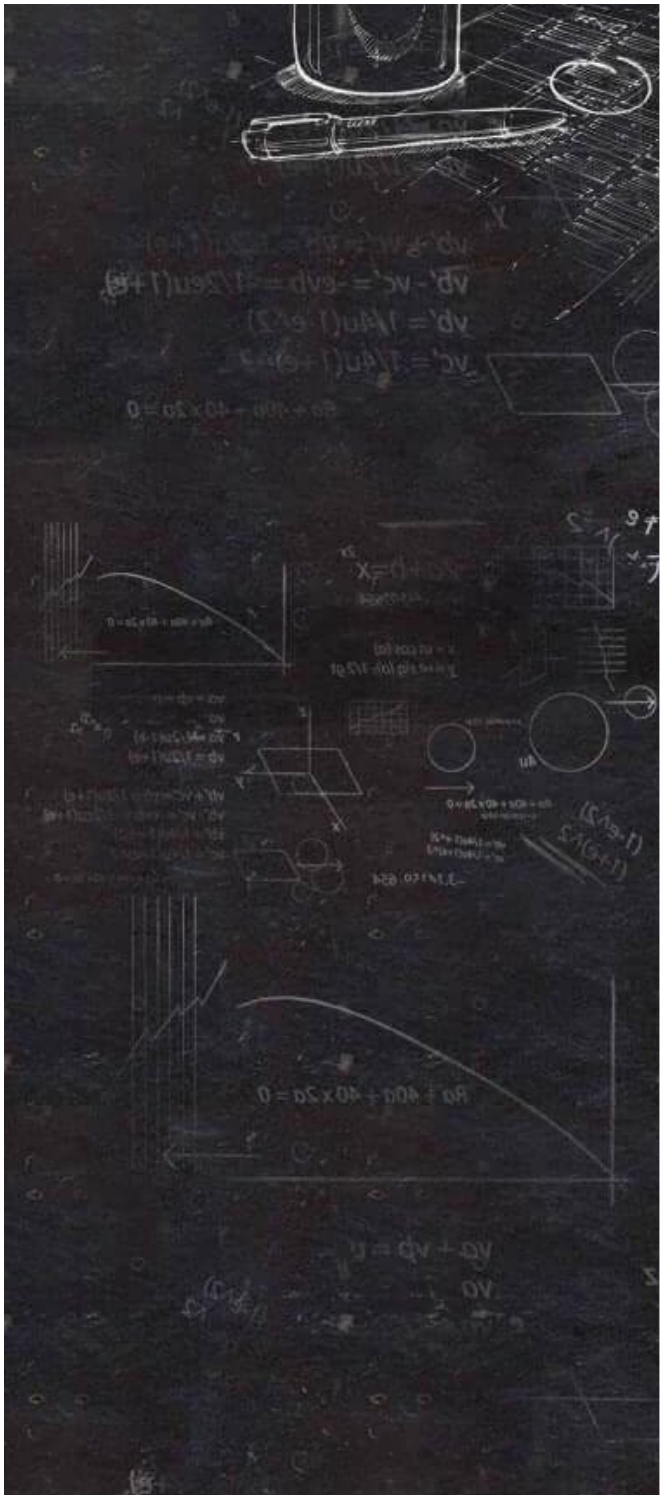
```
144     inf[9]=".....................";
145
146 }
147
148 // 判断是否超过1e18
149 int check(int x, int y)
150 {
151     if(x == 1) return 1;
152
153     __int128 res = 1;
154     for(int i = 1; i <= y; i ++)
155     {
156         res *= x;
157         if(res > 1e18) return 0;
158     }
159     return res;
160 }
161
162 // 把数字提取到vector数组中
163 void tiqu(int x, vector<int>& vec)
164 {
165     while(x)
166     {
167         vec.push_back(x % 10);
168         x /= 10;
169     }
170     reverse(vec.begin(),vec.end());
171 }
172
173 // 添加大数
174 void add_big(vector<int>& vec)
175 {
176     for(auto x : vec)
177     {
178         for(int i = 0; i < 10; i ++)
179             for(int j = 0; j < 8;j ++)
180                 g[i][col + j] = big[x][i][j];
181         col += 8;
182     }
```



```
183 }
184
185 // 添加小数
186 void add_small(vector<int>& vec)
187 {
188     for(auto x : vec)
189     {
190         for(int i = 0; i < 10; i ++)
191             for(int j = 0; j < 6;j ++)
192                 g[i][col + j] = sma[x][i][j];
193         col += 6;
194     }
195 }
196
197 // 添加等于号
198 void add_dengyu()
199 {
200     for(int i = 0; i < 10; i ++)
201         for(int j = 0; j < 8;j ++)
202         g[i][col + j] = dengyu[i][j];
203     col += 8;
204 }
205
206 // 添加INF
207 void add_inf()
208 {
209     for(int i = 0; i < 10; i ++)
210         for(int j = 0; j < 24;j ++)
211         g[i][col + j] = inf[i][j];
212     col += 24;
213 }
214
215 // 最后一列"."
216 void add_col()
217 {
218     for(int i = 0; i < 10; i ++)
219         g[i][col] = '.';
220     col += 1;
221 }
```



```
222
223 // 打印结果
224 void print()
225 {
226     for(int i = 0; i < 10; i ++)
227     {
228         for(int j = 0; j < col; j ++)
229             cout << g[i][j];
230         cout << endl;
231     }
232 }
233
234 void solve()
235 {
236     col = 0;
237     int x, y;
238     scanf("%lld^{%lld}", &x, &y);
239
240     vector<int> vecx;
241     tiqu(x, vecx);
242     add_big(vecx);
243
244     vector<int> vecy;
245     tiqu(y, vecy);
246     add_small(vecy);
247
248     add_dengyu();
249
250     int res = check(x, y);
251     if(res == 0){
252         add_inf();
253     }
254     else{
255         vector<int> vecres;
256         tiqu(res, vecres);
257         add_big(vecres);
258     }
259     add_col();
260     print();
```



```
}

signed main()
{
    init();
    int T; cin >> T;
    while(T --) solve();
}

```

H - Travel Begins — 贪心+推导

题意是 给一个数n，任意构造k个实数（可为0），他们的和为n，实数小数部分小于0.5的舍掉，大于等于0.5进位，使其k个数都为整数，求这样做后最小之和 和 最大之和
如果$k > 2 * n$可确保 将每个数分为$n/k < 1/2$ 最小为0， 或者每个数分为0.5,其余为0，最大值为$2 * n$

如果$k <= 2 * n$ 直接贪心即可
最小的时候一定是尽可能让前k-1个数为0.499999999…，这样前k-1个数就会全部把小数舍掉
最大的时候一定是尽可能让前k-1个数为0.5， 这样前k-1个数全都变成1
然后让n减去前k-1个数为第k个数，第k个数小数部分大于0.5就进1，最终求得结果

最小的时候
第k个数为 $n - (0.5 - eps) * (k - 1)$ 其中eps趋近于0
前k-1个数之和为0，只需要判断第k个数是否进位即可
第k个数结果推导

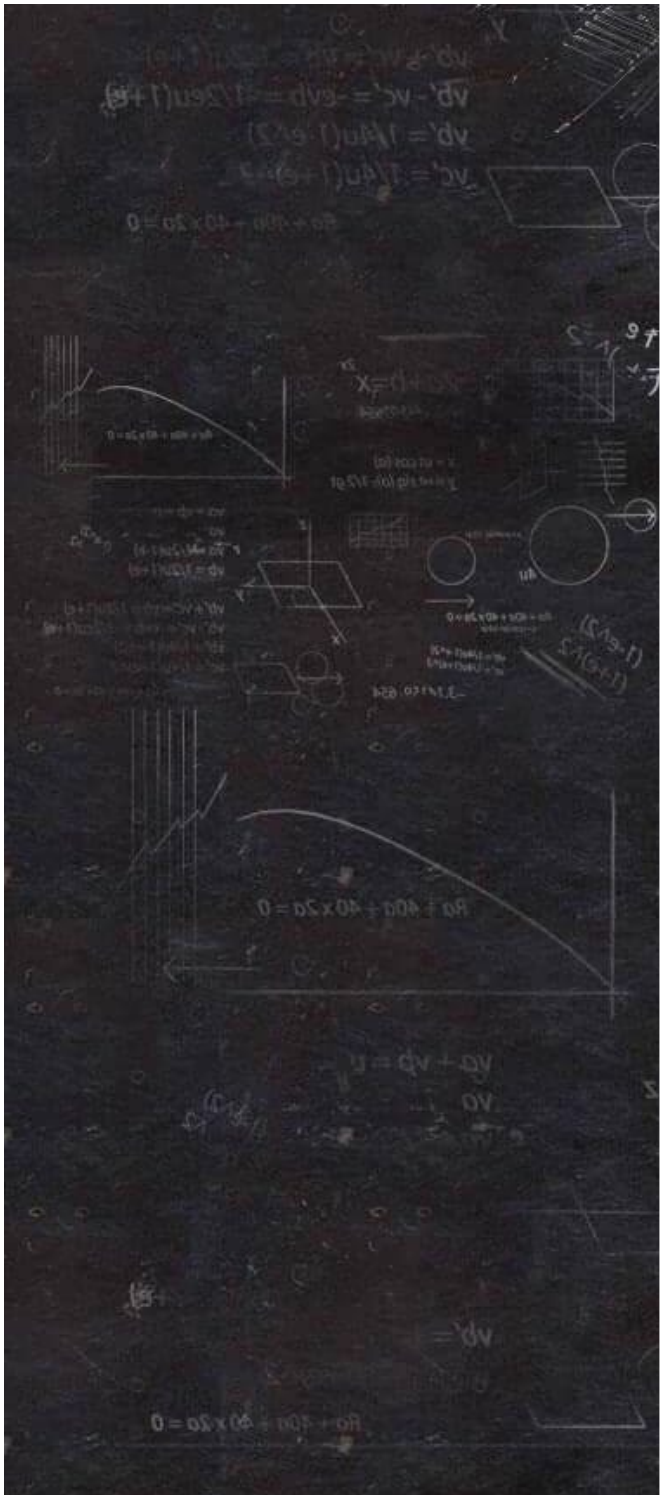
```
res = n - (0.5 - eps) * (k - 1)      其中eps->0
= n - (k - 1) / 2 + eps * (k - 1)    其中eps->0
= n - (k - 1) / 2 + eps              其中eps->0

if k - 1 为奇数
    res = n - (k - 1) / 2
其中 (k - 1) / 2 向下取整，少减了0.5，
但是即使减去后res - 1但小数会剩余 0.5 + eps会进位又 + 1，
所以减不减0.5无所谓

if k - 1 为偶数
    res = n - (k - 1) / 2;
故最小值为 n - k / 2;
```

CSDN @_WAWA鱼_

最大的时候
第k个数为 $n-0.5*(k-1)$
前k-1个数之和为k-1
推导

```
因为前k-1个数都是0.5，进位需要加上k-1
res = k - 1 + n - (k - 1) / 2;
    = n + (k - 1) / 2;

因为 遇0.5就进位 故答案可直接写为
    res = n + k / 2;
if k - 1 为奇数
    小数出现0.5会多进 1，  k / 2符合
if k - 1 为偶数
    k / 2向下取整，也符合
故最大值为 n + k / 2;
```

CSDN @_WAWA鱼_



```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
using namespace std;
void solve()
{
    int n, k; cin >> n >> k;
    if(n * 2 < k) cout << "0 " << n * 2 << '\n';
    else cout << n - (k - 1) / 2 << ' ' << n + k / 2 << '\n';
}

signed main()
{
    int T; cin >> T;
    while(T --) solve();
}

```

K - 排列与质数 — 构造

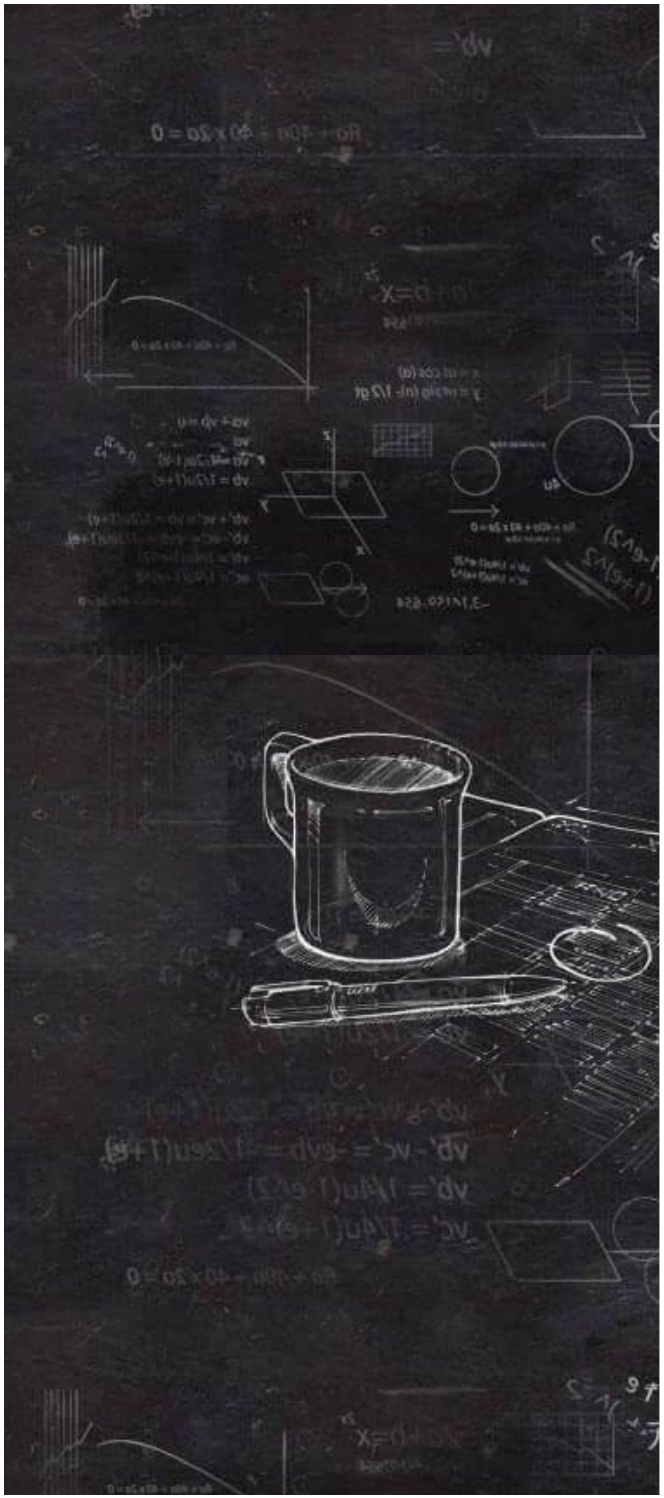
构造题没什么可说的，脑筋急转弯，看官方题解吧

对于 $n \leq 10$，可以暴力枚举排列求解；

对于 $n > 10$ 的奇数，先将数按照 $1, 3, 5, \ldots, n-2, n, n-3, n-5, \ldots, 8, 6, 4$ 排列；

对于 $n > 10$ 的偶数，先将数按照 $1, 3, 5, \ldots, n-3, n, n-2, n-4, \ldots, 8, 6, 4$ 排列；

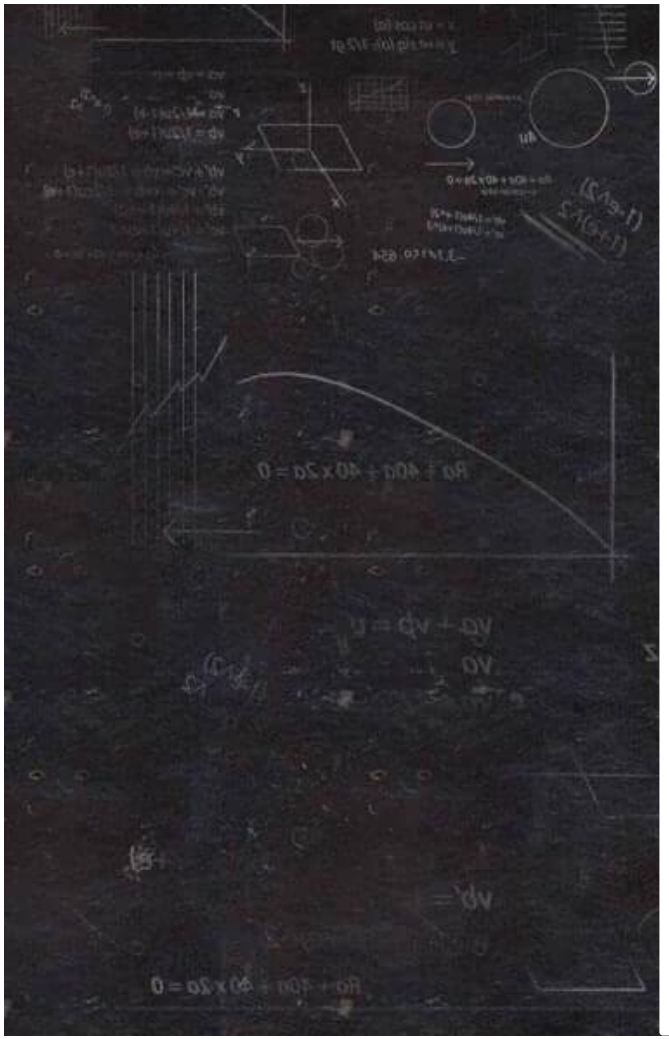
即先将奇数升序排列，再将偶数降序排列。



可以发现，现在除了 2 和 $n-1$ 以外，所有数均已出现，且满足题目的限制。那么我们只需要将这两个数插进合适的位置即可。容易发现一定有解，因为可以将 2 插在 5 和 7 之间，将 $n-1$ 插在 $n-4$ 和 $n-6$ 之间。

复杂度取决于判断质数的速度，$\mathcal{O}(n\sqrt{n})$ 已经足以通过此题。


```cpp
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
using namespace std;

void solve() {
    int n; cin >> n;

    if (n <= 4) cout << -1 << "\n";
    else if (n == 5) cout << "4 1 3 5 2\n";
    else if (n == 6) cout << "1 3 5 2 4 6\n";
    else if (n == 7) cout << "1 3 5 7 2 4 6\n";
    else if (n == 8) cout << "1 3 5 7 2 4 6 8\n";
    else if (n == 9) cout << "1 3 5 7 9 2 4 6 8\n";
    else if (n == 10) cout << "1 3 10 5 7 9 2 4 6 8\n";
    else if (n == 11) cout << "1 3 10 5 2 7 9 11 8 6 4\n";
    else
    {
        vector<int> vec;
        if(n & 1) {
            for(int i = 1; i <= n; i += 2)
            {
                vec.push_back(i);
                if(i == 5) vec.push_back(2);
                if(i == n - 6) vec.push_back(n - 1);
            }
```



```
            for(int i = n - 3; i >= 4; i -= 2)
                vec.push_back(i);
        }
        else {
            for(int i = 1; i <= n - 3; i += 2)
            {
                vec.push_back(i);
                if(i == 5) vec.push_back(2);
            }
            for(int i = n; i >= 4; i -= 2)
            {
                vec.push_back(i);
                if(i == n - 4) vec.push_back(n - 1);
            }
        }
        for(auto x : vec) cout << x << " ";
        cout << '\n';
    }

}
signed main()
{
    int T; T = 1;
    while(T --) solve();
}
```