cf506E.Mr. Kitayuta's Gift 解题报告

题目大意

你有一个由小写字母组成字符串S，现在你可以对字符串S插入恰好n个字符（必须也是小写字母），使得最终的字符串是回文串。请问你可以生成多少个这样的回文串，输出答案模10007的结果。

数据范围

$|S| \le 200$

$n \le 10^9$

解题过程

问题就是，求长度为$n+|S|$的所有能够以S作为子序列的回文串的个数。记$K=26$，我们容易得到一个复杂度为$O(|S|^2 n)$的dp算法。

设$f_{l,r,x}$表示长度为x的回文串中能够匹配S的第l位到第r位的串的数量（认为$l>r$时，对回文串没有限制，但规定$l \le r+2$）。记S_i表示S的第i位字符。

边界条件:当$l \le r$时，$f_{l,r,0}=0$，否则$f_{l,r,0}=1$。当$l \ge r$时，$f_{l,r,1}=K$，否则,$f_{l,r,1}=0$。

转移：设$x \ge 2$。若$l>r$，$f_{i,j,x}=Kf_{i,j,x-1}$，否则若$S_l=S_r$，$f_{i,j,x}=f_{i+1,j-1,x-1}+(K-1)f_{i,j,x-1}$，否则$f_{i,j,x}=f_{i+1,j,x-1}+f_{i,j-1,x-1}+(K-2)f_{i+1,j-1,x-1}$。

如果直接使用矩阵快速幂，那么复杂度为$O(|S|^6 \log n)$，无法通过本题。因此我们需要深入挖掘dp的性质。

我们把这样的dp理解成一个图上路径计数问题，起点为$[1,|S|]$，终点为$[i,i]$或$[i+1,i]$或$[i+1,i-1]$。且路径步数为$[\frac{n+|S|}{2}]$。对$1 \le l \le r \le |S|$，若$S_l=S_r$，$[l,r]$向自身连$K-1$条自环，且向$[l+1,r-1]$连一条边。否则$[l,r]$向自身连$K-2$条自环，且向$[l+1,r],[l,r-1]$各连一条边。若$l>r$则$[l,r]$向自身连K条自环。

这样的路径有什么性质呢？如果我们不考虑路径走了自环，而固定去掉自环之后的路径，那么设路径经过的点上（除了终点以外）有n_1个点有$K-1$个自环，有n_2个点有$K-2$个自环。那么$2n_1+n_2=|S|-1$或$|S|$或$|S|+1$，这是因为每走过有$K-1$个自环的点，区间长度减了1，否则区间长度减了2。

如果我们固定了去掉自环以后的路径，再设该路径有n_0个点有K个自环，有n_1个点有$K-1$个自环，有n_2个点有$K-2$个自环。那么原路径的可能情况数为：

$\frac{x^{n_0+n_1+n_2-1}}{(1-Kx)^{n_0}(1-(K-1)x)^{n_1}(1-(K-2)x)^{n_2}}$ 的$x^{\lceil\frac{n+|S|}{2}\rceil}$项系数。这是因为$n_0$个点走$i$个自环的情况数为$K^i$，然后可以写成生成函数形式，再将它们相乘。

注意到(n_0,n_1,n_2)的总数只有$O(|S|)$种，所以先用$O(|S|^3)$的dp算出所有的(n_0,n_1,n_2)对于多少种去掉自环的路径，再使用多项式快速幂 + 暴力取模计算上式的$x^{\lceil\frac{n+|S|}{2}\rceil}$系数，可以得到$O(|S|^3\log n)$的复杂度，应该可以通过此题。

当然，有一个可以进行优化的地方，就是我们可以将下面的多项式先通分后计算$x^{\lceil\frac{n+|S|}{2}\rceil}$系数，这样你如果用把问题重新转化成常系数线性递推，并使用矩阵快速幂计算，仍然得到$O(|S|^3\log n)$的复杂度。但是使用多项式快速幂 + 取模这样的高级技巧，可以得到$O(|S|^3+\log n)$的复杂度，可以稳稳地通过此题。

参考代码

```
#include <bits/stdc++.h>
#define debug(x) cerr << #x << " " << (x) << endl
using namespace std;

const int S = 205;
const long long K = 26, mod = 10007ll;

long long qpow (long long a, long long k) {
    long long res = 1ll;
    while (k) {
        if (k & 1) res = res * a % mod;
        a = a * a % mod, k >>= 1;
    }
    return res;
}
long long get_inv (long long x) {
    if (x == 1ll) return 1ll;
    return qpow(x, mod - 2);
}

struct poly {
    int deg;
    vector<long long> coef;
    poly () {
        deg = 0;
        coef.push_back(0ll);
```

```
    }

    poly (long long c) {
        deg = 0;
        coef.push_back(c);
    }

    long long get_coef (int k) const {
        return k <= deg ? coef[k] : 0ll;
    }

    poly& operator = (poly rhs) {
        deg = rhs.deg, coef = rhs.coef;
        return *this;
    }

    poly operator + (poly rhs) const {
        poly res;
        res.deg = max(deg, rhs.deg), res.coef.resize(res.deg + 1);
        for (int i = 0; i <= res.deg; i++) res.coef[i] = (get_coef(i) + rhs.get_coef(i)) %
mod;
        while (res.coef.size() > 1 && !res.coef.back()) res.deg--, res.coef.pop_back();
        return res;
    }

    poly operator - (poly rhs) const {
        poly res;
        res.deg = max(deg, rhs.deg), res.coef.resize(res.deg + 1);
        for (int i = 0; i <= res.deg; i++) res.coef[i] = (get_coef(i) + mod -
rhs.get_coef(i)) % mod;
        while (res.coef.size() > 1 && !res.coef.back()) res.deg--, res.coef.pop_back();
        return res;
    }

    poly operator += (poly rhs) {
        return *this = *this + rhs;
    }

    poly operator -= (poly rhs) {
        return *this = *this - rhs;
    }

    poly operator * (poly rhs) const {
        poly res;
        res.deg = deg + rhs.deg, res.coef.resize(res.deg + 1);
        for (int i = 0; i <= deg; i++) {
            for (int j = 0; j <= rhs.deg; j++) res.coef[i + j] = (res.coef[i + j] + coef[i]
* rhs.coef[j]) % mod;
        }
        return res;
    }

    poly operator % (poly rhs) const {
```

```
        poly res = *this;
        for (int i = deg; i >= rhs.deg; i--) {
            long long k = res.coef[i] * get_inv(rhs.coef[rhs.deg]) % mod;
            for (int j = 0; j <= rhs.deg; j++) {
                res.coef[i - rhs.deg + j] = (res.coef[i - rhs.deg + j] + mod * mod -
rhs.coef[j] * k) % mod;
            }
        }
        while (res.coef.size() > 1 && !res.coef.back()) res.deg--, res.coef.pop_back();
        return res;
    }

} f0, f1, f2, F, G, pw[S << 1], pw1[S << 1], pw2[S << 1], id;

int s, n;
char str[S];
long long f[S][S][S], tmp0[S], tmp1[S], tmp2[S];

long long calc (poly f, poly g, int k) {
    vector<long long> vec(g.deg);
    for (int i = 0; i < g.deg; i++) {
        vec[i] = f.get_coef(i);
        for (int j = 1; j <= i; j++) vec[i] = (vec[i] + mod * mod - g.coef[j] * vec[i - j])
% mod;
    }

    poly res(1ll), a = id, g_;
    g_.deg = g.deg, g_.coef.resize(g.deg + 1);
    for (int i = 0; i <= g.deg; i++) g_.coef[i] = g.coef[g.deg - i];
    while (k) {
        if (k & 1) res = res * a % g_;
        a = a * a % g_, k >>= 1;
    }


    long long ans = 0ll;
    for (int i = 0; i < g.deg; i++) ans = (ans + vec[i] * res.get_coef(i)) % mod;
    return ans;
}
int main () {
    scanf("%s", str + 1), s = strlen(str + 1);
    scanf("%d", &n);

    for (int i = 0; i <= s + 1; i++) {
        for (int j = 0; j <= s + 1; j++) {
            for (int k = 0; k <= s; k++) f[i][j][k] = 0ll;
        }
    }
    for (int i = 1; i <= s; i++) {
        for (int j = s; j >= i; j--) {
            for (int k = 0; k <= s; k++) {
                if (i == 1 && j == s && k == 0) f[i][j][k] = 1ll;
```

```
                if (str[i] == str[j]) f[i + 1][j - 1][k + 1] = (f[i + 1][j - 1][k + 1] +
f[i][j][k]) % mod;
                else {
                    f[i + 1][j][k] = (f[i + 1][j][k] + f[i][j][k]) % mod;
                    f[i][j - 1][k] = (f[i][j - 1][k] + f[i][j][k]) % mod;
                }
            }
        }
    }

    for (int i = 0; i <= s; i++) tmp0[i] = 0ll, tmp1[i] = tmp2[i] = 0ll;
    for (int i = 0; i <= s + 1; i++) {
        for (int j = 0; j <= s + 1; j++) {
            for (int k = 0; k <= s; k++) {
                if (i < j || !f[i][j][k]) continue;
                if (i - j == 1) tmp1[k] = (tmp1[k] + f[i][j][k]) % mod;
                if (i - j == 2) tmp2[k] = (tmp2[k] + f[i][j][k]) % mod;
                if (i == j && (n + s & 1)) tmp0[k] = (tmp0[k] + f[i][j][k]) % mod;
            }
        }
    }

    id.deg = 1, id.coef.resize(2);
    id.coef[1] = 1ll;

    f0.deg = f1.deg = f2.deg = 1;
    f0.coef.resize(2), f1.coef.resize(2), f2.coef.resize(2);

    f0.coef[0] = f1.coef[0] = f2.coef[0] = 1ll;
    f0.coef[1] = (mod - K) % mod, f1.coef[1] = (mod - K + 1) % mod, f2.coef[1] = (mod - K +
2) % mod;
    pw[0] = pw1[0] = pw2[0] = poly(1ll);

    for (int i = 1; i <= 2 * s; i++) {
        pw[i] = pw[i - 1] * id % mod;
        pw1[i] = pw1[i - 1] * f1 % mod;
        pw2[i] = pw2[i - 1] * f2 % mod;
    }

    G = f0 * pw1[s] * pw2[s + 1];
    for (int i = 0; i <= s; i++) {
        if (i < s) F += poly(tmp0[i]) * pw[s - 1 - i] * f0 * pw1[s - 1 - i] * pw2[i + 1 <<
1];
        if (n + s & 1) F += poly(tmp1[i] * K % mod) * pw[s - i] * pw1[s - i] * pw2[i << 1 |
1];
        else F += poly(tmp1[i]) * pw[s - i] * pw1[s - i] * pw2[i << 1 | 1];
        if (n + s & 1) F += poly(tmp2[i] * K % mod) * pw[s + 1 - i] * pw1[s - i] * pw2[i <<
1];
        else F += poly(tmp2[i]) * pw[s + 1 - i] * pw1[s - i] * pw2[i << 1];
    }

    printf("%lld\n", calc(F, G, n + s >> 1));
    return 0;
```

```
}
```

cf639F. Bear and Chemistry 解题报告

题目大意

你有一张无向图$G=(V,E)$，每次询问，会把无向图加上一些新的边，再给你一个顶点的集合S，问你是否对于任意$\{x,y\}\subseteq S$，都存在一条从x到y再回到x的不经过重复边的路径（询问是独立的，且强制在线）。

其中$|V|=n,|E|=m$，询问次数为q。第i次询问所添加的边数为m_i，且给定的点集S的大小是n_i。

数据范围

$1\leq n,q\leq 300000, 0\leq m\leq 300000$

$\sum_{i=1}^{q} n_i \leq 300000$

$\sum_{i=1}^{q} m_i \leq 300000$

参考解答

我们考虑加边后的新图的每个边双连通分量，那么如果x,y不在一个边双，那么从x到y必定经过一条割边（删去这条边能使x到y不连通），从y到x也一定经过这条割边，与不经过重复边的条件矛盾。而如果它在一个边双内，那么从x到y存在两条边不相交的路径，所以条件成立。问题可以转换为询问S是否在同一边双内。

如果我们每次对于加边后的新图求一遍边双，那么会获得超时。不过为了简化问题，可以先将原先给的图G缩成边双（使用tarjan算法），然后建成边双树。这样，就可以把本来在同一边双的点看成一个点，只需做G是树的情况。

我们把询问中加的边所在的点和集合中给的点，作为关键点。然后，我们可以对关键点建虚树，再对这些虚树的边和新加的边作为一张新图G'。把G'缩成边双，然后判断S中所有集合是否在同一边双即可。

下面解释为什么建虚树能够保留原有的边双的信息。那是因为在树上如果有一个点度数为1且不是关键点，那么可以把它删去。如果有一个点度数为2且不是关键点，设u连接了v,w，将u删去，连接上v,w，边双的信息不变。这样不断地操作下去，可以得到关键点的虚树。

总复杂度为$O(m+n\log n+\sum_{i=1}^{q} n_i+\sum_{i=1}^{q} m_i)$（若使用Euler序+RMQ求LCA）。

参考代码

```
#include <bits/stdc++.h>
#define debug(x) cerr << #x << " " << (x) << endl
using namespace std;

const int N = 300005, M = 300005, LOGN = 21;

template <class T>
void read (T &x) {
    int sgn = 1;
    char ch;
    x = 0;
    for (ch = getchar(); (ch < '0' || ch > '9') && ch != '-'; ch = getchar()) ;
    if (ch == '-') ch = getchar(), sgn = -1;
    for (; '0' <= ch && ch <= '9'; ch = getchar()) x = x * 10 + ch - '0';
    x *= sgn;
}

template <class T>
void write (T x) {
    if (x < 0) putchar('-'), write(-x);
    else if (x < 10) putchar(x +'0');
    else write(x / 10), putchar(x % 10 + '0');
}

struct edge {
    int to, nxt;
} ;
struct graph {
    int V, head[N], cnt;
    edge g[M << 1];
    void init (int v) {
        V = v;
        for (int i = 0; i < v; i++) head[i] = -1;
        cnt = 0;
    }

    void addedge (int u, int v) {
```

```
        edge e = {v, head[u]};
        g[head[u] = cnt++] = e;
    }

    int low[N], dfn[N], bcc[N], tot, t;
    stack<int> stk;

    void dfs (int u, int fa) {
        low[u] = dfn[u] = ++tot;
        stk.push(u);
        for (int i = head[u]; ~i; i = g[i].nxt) {
            int v = g[i].to;
            if (i != fa) {
                if (!dfn[v]) dfs(v, i ^ 1), low[u] = min(low[u], low[v]);
                else low[u] = min(low[u], dfn[v]);
            }
        }

        if (low[u] == dfn[u]) {
            for (; ;) {
                int x = stk.top();
                bcc[x] = t, stk.pop();
                if (x == u) break;
            }
            t++;
        }
    }
    void get_bcc () {
        tot = t = 0;
        for (int i = 0; i < V; i++) low[i] = dfn[i] = 0;
        for (int i = 0; i < V; i++) {
            if (!dfn[i]) dfs(i, -1);
        }
    }
} g1, tree, g2;

int dep[N], euler[N << 1][LOGN], id[N], Log2[N << 1], cnt = 0;
void dfs (int u, int fa) {
    euler[id[u] = ++cnt][0] = u;
    for (int i = tree.head[u]; ~i; i = tree.g[i].nxt) {
        int v = tree.g[i].to;
        if (v != fa) {
            dep[v] = dep[u] + 1;
            dfs(v, u);
            euler[++cnt][0] = u;
        }
    }
}

void rmq_init () {
    for (int i = 1; i <= cnt; i++) {
        for (int j = 1; (1 << j) <= i; j++) {
```

```
            if (dep[euler[i][j - 1]] < dep[euler[i - (1 << (j - 1))][j - 1]]) euler[i][j] = euler[i][j - 1];
            else euler[i][j] = euler[i - (1 << (j - 1))][j - 1];
        }
    }
    Log2[1] = 0;
    for (int i = 2; i <= cnt; i++) Log2[i] = Log2[i >> 1] + 1;
}

int lca (int u, int v) {
    int l = id[u], r = id[v];
    if (l > r) swap(l, r);
    int label = Log2[r - l + 1];
    if (dep[euler[r][label]] < dep[euler[l + (1 << label) - 1][label]]) return euler[r][label];
    return euler[l + (1 << label) - 1][label];
}

int n, m, q, stk[N], R = 0, top = 0;
vector<int> vec, vec_v;
vector<pair<int, int> > vec_e;


bool cmp (int x, int y) {
    return id[x] < id[y];
}
int main () {
    read(n), read(m), read(q);
    g1.init(n);
    for (int i = 0; i < m; i++) {
        int u, v;
        read(u), read(v);
        u--, v--;
        g1.addedge(u, v), g1.addedge(v, u);
    }

    g1.get_bcc();
    tree.init(g1.t + 1);
    for (int i = 0; i < n; i++) {
        for (int j = g1.head[i]; ~j; j = g1.g[j].nxt) {
            int u = i, v = g1.g[j].to;
            if (g1.bcc[u] != g1.bcc[v]) tree.addedge(g1.bcc[u], g1.bcc[v]);
        }
    }

    for (int i = 0; i <= g1.t; i++) dep[i] = 0;
    euler[id[g1.t] = ++cnt][0] = g1.t;
    for (int i = 0; i < g1.t; i++) {
        if (!dep[i]) {
            tree.addedge(g1.t, i), tree.addedge(i, g1.t);
            dep[i] = 1, dfs(i, g1.t);
            euler[++cnt][0] = g1.t;
        }
```

```
    }
    rmq_init();

    for (int i = 1; i <= q; i++) {
        vec.clear(), vec_v.clear(), vec_e.clear();
        int V, E;
        read(V), read(E);
        for (int j = 0; j < V; j++) {
            int u;
            read(u);
            u--, u = (u + R) % n;
            vec_v.push_back(g1.bcc[u]), vec.push_back(g1.bcc[u]);
        }
        for (int j = 0; j < E; j++) {
            int u, v;
            read(u), read(v);
            u--, u = (u + R) % n;
            v--, v = (v + R) % n;
            vec_e.push_back(make_pair(g1.bcc[u], g1.bcc[v]));
            vec_v.push_back(g1.bcc[u]), vec_v.push_back(g1.bcc[v]);
        }

        sort(vec_v.begin(), vec_v.end(), cmp);

        stk[top = 0] = g1.t;
        for (int j = 0; j < vec_v.size(); j++) {
            int p = lca(vec_v[j], stk[top]);
            while (top && id[stk[top - 1]] >= id[p]) {
                vec_e.push_back(make_pair(stk[top - 1], stk[top]));
                top--;
            }
            if (p != stk[top]) {
                vec_e.push_back(make_pair(p, stk[top]));
                stk[top] = p;
            }
            if (stk[top] != vec_v[j]) stk[++top] = vec_v[j];
        }

        while (top) {
            vec_e.push_back(make_pair(stk[top - 1], stk[top]));
            top--;
        }

        vec_v.clear();
        for (int j = 0; j < vec_e.size(); j++) {
            vec_v.push_back(vec_e[j].first);
            vec_v.push_back(vec_e[j].second);
        }

        sort(vec_v.begin(), vec_v.end());
        int s = unique(vec_v.begin(), vec_v.end()) - vec_v.begin();
        g2.init(s);
```

```
        for (int j = 0; j < vec_e.size(); j++) {
            if (vec_e[j].first == g1.t) continue;
            if (vec_e[j].second == g1.t) continue;
            int u = lower_bound(vec_v.begin(), vec_v.begin() + s, vec_e[j].first) - 
vec_v.begin();
            int v = lower_bound(vec_v.begin(), vec_v.begin() + s, vec_e[j].second) - 
vec_v.begin();
            g2.addedge(u, v), g2.addedge(v, u);
        }

        g2.get_bcc();
        bool flag = true;
        for (int j = 1; j < vec.size(); j++) {
            int u = lower_bound(vec_v.begin(), vec_v.begin() + s, vec[j]) - vec_v.begin();
            int v = lower_bound(vec_v.begin(), vec_v.begin() + s, vec[0]) - vec_v.begin();
            if (g2.bcc[u] != g2.bcc[v]) flag = false;
        }
        if (flag) puts("YES"), R = (R + i) % n;
        else puts("NO");
    }
    return 0;
}
```

agc039D.Incenters 解题报告

题目大意

给定在笛卡尔坐标系的单位圆上的N个点（圆心为$(0,0)$）。第i个点的坐标为$(cos(\frac{2\pi T_i}{L}),sin(\frac{2\pi T_i}{L}))$。

三个不同的点将在这N个点中等概率的随机，请求出这三个点构成的三角形的内切圆圆心的x坐标的数学期望和y坐标的数学期望。

数据范围

$3 \le N \le 3000$

$N \le L \le 10^9$

$0 \le T_i \le L-1$

$T_i < T_{i+1}$

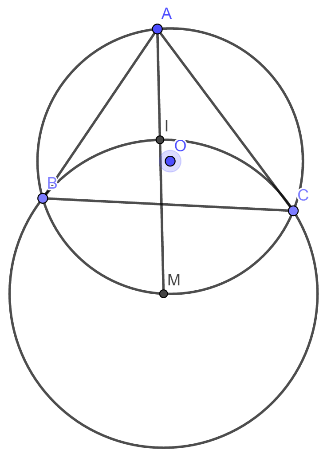
所有的输入的数都是整数。

解题过程

首先，我们考虑一种内心的刻画方法（这种刻画方法在数学竞赛中被称为"鸡爪定理")。

设$\triangle ABC$ 的内心为I，AI与$\triangle\ ABC$的外接圆交于另一点M，则$BM = CM = IM$ 。

证明：由于$\angle BAM = \angle CAM$，故$\angle BCM = \angle CBM$，所以$BM = CM$。又因为
$\angle IBM = \angle CBM + \angle IBC = \frac{1}{2}(\angle CAB + \angle ABC) = \angle MAB + \angle ABI = \angle BIM$，所以$BM = MI$



这样，如果我们固定了B, C两点，以及A在B, C与圆的哪一段弧上，我们就可以得到弧BC的中点M（M与A在BC异侧）。我们不能枚举A点，但是我们将I刻画为:$M + (B - M) \cdot e^{i\angle AMB}$。（这里我们使用了复平面的工具），那么$M$是固定的,$B - M$是固定的（即与$A$无关）。要求所有$I$的坐标之和，只需要知道$e^{i\angle AMB}$的和。虽然这个式子与$A, B$有关，但是与$M, C$均无关（圆周角相等）。

因此，我们先枚举B，接着逆时针顺序枚举C，在枚举的过程中顺便维护

<1> 从B逆时针到C的点的数目

<2> 对于从B逆时针到C经过的点A，维护$e^{i\angle AMB}$的和。

这样，我们就可以以$O(n^2)$的复杂度算出内心的坐标和了。但是，每个内心被算了三次，而且我们最终答案是内心横纵坐标的期望，所以要将答案除以$\frac{n(n-1)(n-2)}{2}$

参考代码

```cpp
#include <bits/stdc++.h>
using namespace std;

const int N = 3005;
const double PI = acos(-1), eps = 1e-10;

int n, L;
double alpha[N], ansx = 0.0, ansy = 0.0;

double midpoint (double x1, double x2) {
    double len = x2 - x1;
    if (len < -eps) len += 2.0 * PI;
    len /= 2.0;
    double mid = x1 + len;
    if (mid >= 2.0 * PI - eps) mid -= 2.0 * PI;
    return mid;
}

int main () {
    scanf("%d%d", &n, &L);
    for (int i = 0; i < n; i++) {
        int x;
        scanf("%d", &x);
        alpha[i] = 2.0 * PI * x / L;
    }

    for (int i = 0; i < n; i++) {
        double nowx = 0.0, nowy = 0.0;
        for (int j = (i + 1) % n, k = 0; j != i; j = (j + 1) % n, k++) {
            double m = midpoint(alpha[j], alpha[i]), arg = alpha[j] - alpha[i];
            double vecx = cos(alpha[i]) - cos(m), vecy = sin(alpha[i]) - sin(m);

            ansx += cos(m) * k, ansy += sin(m) * k;
            ansx += vecx * nowx - vecy * nowy, ansy += vecx * nowy + vecy * nowx;

            if (arg < -eps) arg += 2.0 * PI;
            nowx += cos(arg / 2.0), nowy += sin(arg / 2.0);
        }
    }

    ansx /= 0.5 * n * (n - 1) * (n - 2), ansy /= 0.5 * n * (n - 1) * (n - 2);
    printf("%.10lf %.10lf\n", ansx, ansy);
    return 0;
}
```