字符串

KMP 自动机

```
ll s[MAXN], a[MAXN], b[MAXN];
vector<int> del[MAXN];
int pre[MAXN]; // border
int nxt[MAXN];
void solve(){
    int n;
    cin >> n;
    ll last = 0;
    ll sum = 0;
    rep(i, 1, n){
        cin >> s[i] >> a[i] >> b[i];
        s[i] = (s[i] + last) % n;
        if(i == 1){
            pre[i] = 0;
            nxt[i] = 0;
            last = a[i] * b[i];
            sum += b[i];
            cout << last << endl;
            continue;
        }
        if(pre[i - 1] == 0){
            if(s[i] == s[1]){
                pre[i] = 1;
//              del[i].push_back(1);
                nxt[i] = 0;
            }
            else{
                pre[i] = 0;
                nxt[i] = 0;
            }
        }
        else{
            if(s[pre[i - 1] + 1] == s[i]){
                pre[i] = pre[i - 1] + 1;
                if(del[pre[i]].size()){
                    nxt[i] = pre[i];
                }
                else{
                    nxt[i] = nxt[pre[i]];
                }
            }
            else{
                int p = pre[i - 1];
                while(p && s[p + 1] != s[i]){
                    del[i].push_back(p);
                    p = pre[p];
                }
                if(s[p + 1] == s[i]){
                    pre[i] = p + 1;
                    nxt[i] = p + 1;
                }
                else{
                    pre[i] = 0;
                    nxt[i] = 0;
                }
            }
        }
        if(s[i] == s[1]){
            sum += b[i];
        }
        int p = i;
        while(p){
            for(auto pos : del[p]){
                sum -= b[i - 1 - pos + 1];
            }
            p = nxt[p];
        }
        last += a[i] * sum;
        cout << last << endl;
    }
}
```

马拉车（Manacher)

```
std::vector<int> manacher(std::string s) {
    std::string t = "#";
    for (auto c : s) {
        t += c;
        t += '#';
    }
    int n = t.size();
    std::vector<int> r(n);
    for (int i = 0, j = 0; i < n; i++) {
        if (2 * j - i >= 0 && j + r[j] > i) {
            r[i] = std::min(r[2 * j - i], j + r[j] - i);
        }
        while (i - r[i] >= 0 && i + r[i] < n && t[i - r[i]] == t[i + r[i]]) {
```

```cpp
            r[i] += 1;
        }
        if (i + r[i] > j + r[j]) {
            j = i;
        }
    }
    return r;
}
```

z 函数

```cpp
std::vector<int> Z(std::string s) {
    int n = s.size();
    std::vector<int> z(n + 1);
    z[0] = n;
    for (int i = 1, j = 1; i < n; i++) {
        z[i] = std::max(0, std::min(j + z[j] - i, z[i - j]));
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) {
            z[i]++;
        }
        if (i + z[i] > j + z[j]) {
            j = i;
        }
    }
    return z;
}
```

后缀数组

```cpp
struct SA {
    int n;
    std::vector<int> sa, rk, lc;

    SA(std::string s) {
        n = s.size();
        sa.resize(n);
        lc.resize(n - 1);
        rk.resize(n);
        std::iota(sa.begin(), sa.end(), 0);
        std::sort(sa.begin(), sa.end(),
            [&](int a, int b) {
                return s[a] < s[b];
            });
        rk[sa[0]] = 0;
        for (int i = 1; i < n; i++) {
            rk[sa[i]] = rk[sa[i - 1]] + (s[sa[i]] != s[sa[i - 1]]);
        }
        int k = 1;
        std::vector<int> tmp, cnt(n);
        tmp.reserve(n);
        while (rk[sa[n - 1]] < n - 1) {
            tmp.clear();
            for (int i = 0; i < k; i++) {
                tmp.push_back(n - k + i);
            }
            for (auto i : sa) {
                if (i >= k) {
                    tmp.push_back(i - k);
                }
            }
            std::fill(cnt.begin(), cnt.end(), 0);
            for (int i = 0; i < n; i++) {
                cnt[rk[i]]++;
            }
            for (int i = 1; i < n; i++) {
                cnt[i] += cnt[i - 1];
            }
            for (int i = n - 1; i >= 0; i--) {
                sa[--cnt[rk[tmp[i]]]] = tmp[i];
            }
            std::swap(rk, tmp);
            rk[sa[0]] = 0;
            for (int i = 1; i < n; i++) {
                rk[sa[i]] = rk[sa[i - 1]] + (tmp[sa[i - 1]] < tmp[sa[i]] || sa[i - 1] + k == n || tmp[sa[i - 1] + k] < tmp[sa[i] + k]);
            }
            k *= 2;
        }
        for (int i = 0, j = 0; i < n; i++) {
            if (rk[i] == 0) {
                j = 0;
            } else {
                for (j -= j > 0; i + j < n && sa[rk[i] - 1] + j < n && s[i + j] == s[sa[rk[i] - 1] + j]; ) {
                    j++;
                }
                lc[rk[i] - 1] = j;
            }
        }
    }
};

void solve() {
    constexpr int K = 21;
    std::vector st(K, std::vector<int>(l - 1));
    st[0] = lc;
    for (int j = 0; j < K - 1; j++) {
        for (int i = 0; i + (2 << j) <= l - 1; i++) {
            st[j + 1][i] = std::min(st[j][i], st[j][i + (1 << j)]);
        }
    }

    auto rmq = [&](int l, int r) {
        int k = std::__lg(r - l);
        return std::min(st[k][l], st[k][r - (1 << k)]);
    };

    auto lcp = [&](int i, int j) {
        if (i == j || i == n || j == n) {
            return std::min(n - i, n - j);
        }
        int a = rk[i];
        int b = rk[j];
        if (a > b) {
            std::swap(a, b);
        }
        return std::min({n - i, n - j, rmq(a, b)});
    };

    auto lcs = [&](int i, int j) {
        if (i == j || i == 0 || j == 0) {
            return std::min(i, j);
        }
        int a = rk[n + n - i];
        int b = rk[n + n - j];
        if (a > b) {
            std::swap(a, b);
        }
        return std::min({i, j, rmq(a, b)});
    };
}
```

后缀自动机

```
struct SAM {
    static constexpr int ALPHABET_SIZE = 26;
    struct Node {
        int len;
        int link;
        std::array<int, ALPHABET_SIZE> next;
        Node() : len{}, link{}, next{} {}
    };
    std::vector<Node> t;
    SAM() {
        init();
    }
    void init() {
        t.assign(2, Node());
        t[0].next.fill(1);
        t[0].len = -1;
    }
    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }
    int extend(int p, int c) {
        if (t[p].next[c]) {
            int q = t[p].next[c];
            if (t[q].len == t[p].len + 1) {
                return q;
            }
            int r = newNode();
            t[r].len = t[p].len + 1;
            t[r].link = t[q].link;
            t[r].next = t[q].next;
            t[q].link = r;
            while (t[p].next[c] == q) {
                t[p].next[c] = r;
                p = t[p].link;
            }
            return r;
        }
        int cur = newNode();
        t[cur].len = t[p].len + 1;
        while (!t[p].next[c]) {
            t[p].next[c] = cur;
            p = t[p].link;
        }
        t[cur].link = extend(p, c);
        return cur;
    }
    int extend(int p, char c, char offset = 'a') {
        return extend(p, c - offset);
    }

    int next(int p, int x) {
        return t[p].next[x];
    }

    int next(int p, char c, char offset = 'a') {
        return next(p, c - 'a');
    }

    int link(int p) {
        return t[p].link;
    }

    int len(int p) {
        return t[p].len;
    }

    int size() {
        return t.size();
    }
};
```

回文自动机

```
struct PAM {
    static constexpr int ALPHABET_SIZE = 26;
    struct Node {
        int len;
        int link;
        int cnt;
        std::array<int, ALPHABET_SIZE> next;
        Node() : len{}, link{}, cnt{}, next{} {}
    };
    std::vector<Node> t;
    int suff;
    std::string s;
    PAM() {
        init();
    }
    void init() {
        t.assign(2, Node());
        t[0].len = -1;
        suff = 1;
        s.clear();
    }
    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }
    bool add(char c) {
        int pos = s.size();
        s += c;
        int let = c - 'a';
        int cur = suff, curlen = 0;
        while (true) {
            curlen = t[cur].len;
            if (pos - 1 - curlen >= 0 && s[pos - 1 - curlen] == s[pos]) {
                break;
            }
            cur = t[cur].link;
        }
        if (t[cur].next[let]) {
            suff = t[cur].next[let];
            return false;
        }
        int num = newNode();
        suff = num;
        t[num].len = t[cur].len + 2;
        t[cur].next[let] = num;
        if (t[num].len == 1) {
            t[num].link = 1;
            t[num].cnt = 1;
            return true;
        }
        while (true) {
            cur = t[cur].link;
            curlen = t[cur].len;
            if (pos - 1 - curlen >= 0 && s[pos - 1 - curlen] == s[pos]) {
                t[num].link = t[cur].next[let];
                break;
            }
        }
        t[num].cnt = 1 + t[t[num].link].cnt;
        return true;
    }
    int next(int p, int x) {
        return t[p].next[x];
    }
    int link(int p) {
        return t[p].link;
    }
    int len(int p) {
        return t[p].len;
    }
    int size() {
        return t.size();
    }
};
```

字符串哈希

```
#include <bits/stdc++.h>

using i64 = long long;

bool isprime(int n) {
    if (n <= 1) {
        return false;
    }
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}

int findPrime(int n) {
    while (!isprime(n)) {
        n++;
    }
    return n;
}

using Hash = std::array<int, 2>;
```

```cpp
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    std::mt19937 rng(std::chrono::steady_clock::now().time_since_epoch().count());

    const int P = findPrime(rng() % 900000000 + 100000000);

    std::string s, x;
    std::cin >> s >> x;

    int n = s.length();
    int m = x.length();

    std::vector<int> h(n + 1), p(n + 1);
    for (int i = 0; i < n; i++) {
        h[i + 1] = (10LL * h[i] + s[i] - '0') % P;
    }
    p[0] = 1;
    for (int i = 0; i < n; i++) {
        p[i + 1] = 10LL * p[i] % P;
    }

    auto get = [&](int l, int r) {
        return (h[r] + 1LL * (P - h[l]) * p[r - l]) % P;
    };

    int px = 0;
    for (auto c : x) {
        px = (10LL * px + c - '0') % P;
    }

    for (int i = 0; i <= n - 2 * (m - 1); i++) {
        if ((get(i, i + m - 1) + get(i + m - 1, i + 2 * m - 2)) % P == px) {
            std::cout << i + 1 << " " << i + m - 1 << "\n";
            std::cout << i + m << " " << i + 2 * m - 2 << "\n";
            return 0;
        }
    }

    std::vector<int> z(m + 1), f(n + 1);
    z[0] = m;

    for (int i = 1, j = -1; i < m; i++) {
        if (j != -1) {
            z[i] = std::max(0, std::min(j + z[j] - i, z[i - j]));
        }
        while (z[i] + i < m && x[z[i]] == x[z[i] + i]) {
            z[i]++;
        }
        if (j == -1 || i + z[i] > j + z[j]) {
            j = i;
        }
    }
    for (int i = 0, j = -1; i < n; i++) {
        if (j != -1) {
            f[i] = std::max(0, std::min(j + f[j] - i, z[i - j]));
        }
        while (f[i] + i < n && f[i] < m && x[f[i]] == s[f[i] + i]) {
            f[i]++;
        }
        if (j == -1 || i + f[i] > j + f[j]) {
            j = i;
        }
    }

    for (int i = 0; i + m <= n; i++) {
        int l = std::min(m, f[i]);

        for (auto j : { m - l, m - l - 1 }) {
            if (j <= 0) {
                continue;
            }
            if (j <= i && (get(i - j, i) + get(i, i + m)) % P == px) {
                std::cout << i - j + 1 << " " << i << "\n";
                std::cout << i + 1 << " " << i + m << "\n";
                return 0;
            }
            if (i + m + j <= n && (get(i, i + m) + get(i + m, i + m + j)) % P ==
px) {
                std::cout << i + 1 << " " << i + m << "\n";
                std::cout << i + m + 1 << " " << i + m + j << "\n";
                return 0;
            }
        }
    }

    return 0;
}
```