

Contents

1	Setting	1
1.1	PS	1
2	Math	1
2.1	Basic Arithmetics	1
3	String	2
3.1	KMP	2
3.2	Manacher	3
3.3	Suffix Array	3
3.4	Suffix Array	3

1 Setting

1.1 PS

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

using namespace std;

#define for1(s, e) for(int i = s; i < e; i++)
#define for1j(s, e) for(int j = s; j < e; j++)
#define forEach(k) for(auto i : k)
#define forEachj(k) for(auto j : k)
#define sz(vct) vct.size()
#define all(vct) vct.begin(), vct.end()
#define sortv(vct) sort(vct.begin(), vct.end())
#define uniq(vct) sort(all(vct));vct.erase(unique(all(vct)), vct.end())
#define fi first
#define se second
#define INF (1ll << 6011)

typedef unsigned long long ull;
typedef long long ll;
typedef ll lllint;
typedef unsigned int uint;
typedef unsigned long long int ull;
typedef ull ullint;

typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
typedef pair<double, double> pdd;
typedef pair<double, int> pdi;
typedef pair<string, string> pss;

typedef vector<int> iv1;
typedef vector<iv1> iv2;
typedef vector<ll> llv1;
typedef vector<llv1> llv2;
```

```
typedef vector<pii> piiv1;
typedef vector<piiv1> piiv2;
typedef vector<pll> pll1;
typedef vector<pll1> pll2;
typedef vector<pdd> pdd1;
typedef vector<pdd1> pdd2;

const double EPS = 1e-8;
const double PI = acos(-1);

template<typename T>
T sq(T x) { return x * x; }

int sign(ll x) { return x < 0 ? -1 : x > 0 ? 1 : 0; }
int sign(int x) { return x < 0 ? -1 : x > 0 ? 1 : 0; }
int sign(double x) { return abs(x) < EPS ? 0 : x < 0 ? -1 : 1; }

void solve() {

}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(NULL);cout.tie(NULL);
    int tc = 1; // cin >> tc;
    while(tc--) solve();
}
```

2 Math

2.1 Basic Arithmetics

```
typedef long long ll;
typedef unsigned long long ull;

// calculate lg2(a)
inline int lg2(ll a) {
    return 63 - __builtin_clzll(a);
}

// calculate the number of 1-bits
inline int bitcount(ll a) {
    return __builtin_popcountll(a);
}

// calculate ceil(a/b)
// |a|, |b| <= (2^63)-1 (does not dover -2^63)
ll ceildiv(ll a, ll b) {
    if (b < 0) return ceildiv(-a, -b);
    if (a < 0) return (-a) / b;
    return ((ull)a + (ull)b - 1ull) / b;
}
```

```
// calculate floor(a/b)
// |a|, |b| <= (2^63)-1 (does not cover -2^63)
ll floordiv(ll a, ll b) {
    if (b < 0) return floordiv(-a, -b);
    if (a >= 0) return a / b;
    return -(ll)((ull)(-a) + b - 1) / b;
}

// calculate a*b % m
// x86-64 only
ll large_mod_mul(ll a, ll b, ll m) {
    return ll((__int128)a*(__int128)b%m);
}

// calculate a*b % m
// |m| < 2^62, x86 available
// O(logb)
ll large_mod_mul(ll a, ll b, ll m) {
    a %= m; b %= m; ll r = 0, v = a;
    while (b) {
        if (b&1) r = (r + v) % m;
        b >>= 1;
        v = (v << 1) % m;
    }
    return r;
}

// calculate n^k % m
ll modpow(ll n, ll k, ll m) {
    ll ret = 1;
    n %= m;
    while (k) {
        if (k & 1) ret = large_mod_mul(ret, n, m);
        n = large_mod_mul(n, n, m);
        k /= 2;
    }
    return ret;
}

// calculate gcd(a, b)
ll gcd(ll a, ll b) {
    return b == 0 ? a : gcd(b, a % b);
}

// find a pair (c, d) s.t. ac + bd = gcd(a, b)
pair<ll, ll> extended_gcd(ll a, ll b) {
    if (b == 0) return { 1, 0 };
    auto t = extended_gcd(b, a % b);
    return { t.second, t.first - t.second * (a / b) };
}

// find x in [0, m) s.t. ax === gcd(a, m) (mod m)
ll modinverse(ll a, ll m) {
    return (extended_gcd(a, m).first % m + m) % m;
}
```

```
// calculate modular inverse for 1 ~ n
void calc_range_modinv(int n, int mod, int ret[]) {
    ret[1] = 1;
    for (int i = 2; i <= n; ++i)
        ret[i] = (ll)(mod - mod/i) * ret[mod%i] % mod;
}
```

3 String

3.1 KMP

```
struct KMP {
    /*
     * s 문자열에서 문자열을 o 찾습니다. 매칭이 시작되는 인덱스 목록을 반환합니다
     *
     * Time: O(n + m)
     */
    vector<int> result;
    int MX;
    string s, o;
    int n, m; // n : s.Length(), m : o.Length();
    vector<int> fail;

    KMP(string s, string o) : s(s), o(o) {
        n = s.length();
        m = o.length();
        MX = max(n, m) + 1;
        fail.resize(MX, 0);

        run();
    }

    void run() {
        for(int i = 1, j = 0; i < m; i++){
            while(j > 0 && o[i] != o[j]) j = fail[j-1];
            if(o[i] == o[j]) fail[i] = ++j;
        }
        for(int i = 0, j = 0; i < n; i++) {
            while(j > 0 && s[i] != o[j]) {
                j = fail[j - 1];
            }
            if(s[i] == o[j]) {
                if(j == m - 1) {
                    // matching OK;
                    result.push_back(i - m + 1);
                    j = fail[j];
                }
                else {
                    j++;
                }
            }
        }
    }
}
```

```
};
```

3.2 Manacher

```
// Use space to insert space between each character
// To get even length palindromes!
// O(|str|)
```

```
vector<int> manacher(string &s) {
    int n = s.size(), R = -1, p = -1;
    vector<int> A(n);
    for (int i = 0; i < n; i++) {
        if (i <= R) A[i] = min(A[2 * p - i], R - i);
        while (i - A[i] - 1 >= 0 && i + A[i] + 1 < n && s[i - A[i] - 1] == s[i + A[i]
            ] + 1))
            A[i]++;
        if (i + A[i] > R)
            R = i + A[i], p = i;
    }
    return A;
}

string space(string &s) {
    string t;
    for (char c : s) t += c, t += ' ';
    t.pop_back();
    return t;
}

int maxpalin(vector<int> &M, int i) {
    if (i % 2) return (M[i] + 1) / 2 * 2;
    return M[i] / 2 * 2 + 1;
}
```

3.3 Suffix Array

```
typedef char T;
```

```
// calculates suffix array.
// O(n*logn)
```

```
vector<int> suffix_array(const vector<T> &in) {
    int n = (int)in.size(), c = 0;
    vector<int> temp(n), pos2bckt(n), bckt(n), bpos(n), out(n);
    for (int i = 0; i < n; i++) out[i] = i;
    sort(out.begin(), out.end(), [&](int a, int b) { return in[a] < in[b]; });
    for (int i = 0; i < n; i++) {
        bckt[i] = c;
        if (i + 1 == n || in[out[i]] != in[out[i + 1]]) c++;
    }
    for (int h = 1; h < n && c < n; h <= 1) {
        for (int i = 0; i < n; i++) pos2bckt[out[i]] = bckt[i];
        for (int i = n - 1; i >= 0; i--) bpos[bckt[i]] = i;
        for (int i = 0; i < n; i++)
            if (out[i] >= n - h) temp[bpos[bckt[i]]++] = out[i];
```

```
        for (int i = 0; i < n; i++)
            if (out[i] >= h) temp[bpos[pos2bckt[out[i] - h]]++] = out[i] - h;
        c = 0;
        for (int i = 0; i + 1 < n; i++) {
            int a = (bckt[i] != bckt[i + 1]) || (temp[i] >= n - h)
                || (pos2bckt[temp[i + 1] + h] != pos2bckt[temp[i] + h]);
            bckt[i] = c;
            c += a;
        }
        bckt[n - 1] = c++;
        temp.swap(out);
    }
    return out;
}

// calculates lcp array. it needs suffix array & original sequence.
// O(n)
vector<int> lcp(const vector<T> &in, const vector<int> &sa) {
    int n = (int)in.size();
    if (n == 0) return vector<int>();
    vector<int> rank(n), height(n - 1);
    for (int i = 0; i < n; i++) rank[sa[i]] = i;
    for (int i = 0, h = 0; i < n; i++) {
        if (rank[i] == 0) continue;
        int j = sa[rank[i] - 1];
        while (i + h < n && j + h < n && in[i + h] == in[j + h]) h++;
        height[rank[i] - 1] = h;
        if (h > 0) h--;
    }
    return height;
}
```

3.4 2nd Suffix Array

```
struct SuffixComparator {
    const vector<int> &group;
    int t;

    SuffixComparator(const vector<int> &_group, int _t) : group(_group), t(_t) {}
    bool operator()(int a, int b) {
        if (group[a] != group[b]) return group[a] < group[b];
        return group[a + t] < group[b + t];
    }
};

vector<int> getSuffixArr(const string &s) {
    int n = s.size();
    int t = 1;

    vector<int> group(n + 1);

    for (int i = 0; i < n; i++) group[i] = s[i];
    group[n] = -1;

    vector<int> perm(n);
```

```
for (int i = 0; i < n; i++) perm[i] = i;

while (t < n) {
    SuffixComparator compare(group, t);
    sort(perm.begin(), perm.end(), compare);
    t *= 2;
    if (t >= n)
        break;

    vector<int> new_group(n + 1);
    new_group[n] = -1;
    new_group[perm[0]] = 0;
    for (int i = 1; i < n; i++)
        if (compare(perm[i - 1], perm[i]))
            new_group[perm[i]] = new_group[perm[i - 1]] + 1;
        else
            new_group[perm[i]] = new_group[perm[i - 1]];
    group = new_group;
}
return perm;
}

int getHeight(const string &s, vector<int> &pos) {
    // 최장중복부분문자열의길이
    const int n = pos.size();
    vector<int> rank(n);
    for (int i = 0; i < n; i++)
        rank[pos[i]] = i;
    int h = 0, ret = 0;
    for (int i = 0; i < n; i++) {
        if (rank[i] > 0) {
            int j = pos[rank[i] - 1];
            while (s[i + h] == s[j + h])
                h++;
            ret = max(ret, h);
            if (h > 0)
                h--;
        }
    }
    return ret;
}
```