Standard Code Library

111

zu

July 28, 2022

Contents

一切的开始

宏定义

- 需要 C++11

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
#ifdef zerol
#define dbg(x...) do { cout << "\033[32;1m" << #x << " -> "; err(x); } while (0)
void err() { cout << "\033[39;0m" << endl; }
template<template<typename...> class T, typename t, typename... A>
void err(T<t> a, A... x) { for (auto v: a) cout << v << ' '; err(x...); }
template<typename T, typename... A>
void err(T a, A... x) { cout << a << ' '; err(x...); }
#else
#define dbg(...)
#endif
// -----------------------------------------------------------------------------
```

数据结构

ST 表

- 二维

```cpp
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

树状数组

```cpp
int t[N], n;

inline int lowbit(int x) { return x & (-x); }

void add(int x, int k) {
    while (x <= n) {
        t[x] += k;
        x += lowbit(x);
    }
}

int qry(int x) {
    int ret = 0;
    while (x) {
        ret = max(ret,t[x]);
        x -= lowbit(x);
    }
    return ret;
}
```

线段树

```
struct node {
    ll sum;
    ll plz, mlz;
    int l, r;
} tree[N * 4];

void build(int i, int l, int r) {
    tree[i].l = l;
    tree[i].r = r;
    tree[i].plz = 0;
    tree[i].mlz = 1;
    if (l == r) {
        cin >> tree[i].sum;
        tree[i].sum = tree[i].sum % p;
        return;
    }
    int mid = (l + r) >> 1;
    build(i * 2, l, mid);
    build(i * 2 + 1, mid + 1, r);
    tree[i].sum = (tree[i * 2].sum + tree[i * 2 + 1].sum) % p;
}

inline void push_down(ll i) {
    ll k1 = tree[i].mlz, k2 = tree[i].plz;
    tree[i << 1].sum = (tree[i << 1].sum * k1 + k2 * (tree[i << 1].r - tree[i << 1].l + 1)) % p;
    tree[i << 1 | 1].sum = (tree[i << 1 | 1].sum * k1 + k2 * (tree[i << 1 | 1].r - tree[i << 1 | 1].l + 1)) % p;
    tree[i << 1].mlz = (tree[i << 1].mlz * k1) % p;
    tree[i << 1 | 1].mlz = (tree[i << 1 | 1].mlz * k1) % p;
    tree[i << 1].plz = (tree[i << 1].plz * k1 + k2) % p;
    tree[i << 1 | 1].plz = (tree[i << 1 | 1].plz * k1 + k2) % p;
    tree[i].plz = 0;
    tree[i].mlz = 1;
}

inline void add(int i, int l, int r, ll k) {
    if (tree[i].l >= l && tree[i].r <= r) {
        tree[i].sum = (tree[i].sum + k * (tree[i].r - tree[i].l + 1)) % p;
        tree[i].plz += k;
        return;
    }
    push_down(i);
    if (tree[i * 2].r >= l) add(i * 2, l, r, k);
    if (tree[i * 2 + 1].l <= r) add(i * 2 + 1, l, r, k);
    tree[i].sum = tree[i * 2].sum + tree[i * 2 + 1].sum;
}

inline void mul(int i, int l, int r, ll k) {
    if (tree[i].l >= l && tree[i].r <= r) {
        tree[i].mlz = tree[i].mlz * k % p;
        tree[i].plz = tree[i].plz * k % p;
        tree[i].sum = tree[i].sum * k % p;
        return;
    }
    push_down(i);
    if (tree[i * 2].r >= l) mul(i * 2, l, r, k);
    if (tree[i * 2 + 1].l <= r) mul(i * 2 + 1, l, r, k);
    tree[i].sum = tree[i * 2].sum + tree[i * 2 + 1].sum;
}

ll search(int i, int l, int r) {
    if (tree[i].l >= l && tree[i].r <= r) {
        return tree[i].sum % p;
    }
    if (tree[i].r < l || tree[i].l > r) return 0;
    push_down(i);
    ll s = 0;
    if (tree[i * 2].r >= l) s = (s + search(i * 2, l, r)) % p;
    if (tree[i * 2 + 1].l <= r) s = (s + search(i * 2 + 1, l, r)) % p;
    return s % p;
}
```

主席树

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5 + 7;
struct node{
    int sum,ls,rs;
}tree[N << 5];
int tot = 0;
int a[N];
int v[N];
int rt[N];
int getid(int x,int n) {
    return lower_bound(v + 1,v + n + 1,x) - v;
}
int build(int l,int r) {
    int root = ++tot;
    tree[root].sum = 0;
    if (l >= r) return root;
    int mid = l + r >> 1;
    tree[root].ls = build(l, mid);
    tree[root].rs = build(mid + 1, r);
    return root;
}
int update(int x,int l, int r,int root) {
    int d = ++tot;
    tree[d].ls = tree[root].ls;
    tree[d].rs = tree[root].rs;
    tree[d].sum = tree[root].sum + 1;
    if (l >= r) return d;
    int mid = l + r >> 1;
    if (x <= mid) tree[d].ls = update(x, l, mid, tree[d].ls);
    else tree[d].rs = update(x, mid + 1, r, tree[d].rs);
    return d;
}
int query(int u, int v, int l, int r, int k) {
    if (l >= r) return l;
    int mid = l + r >> 1;
    int x = tree[tree[v].ls].sum - tree[tree[u].ls].sum;
    if (x >= k) return query(tree[u].ls,tree[v].ls,l,mid,k);
    else query(tree[u].rs, tree[v].rs, mid + 1, r, k - x);
}

void solve() {
    tot = 0;
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        v[i] = a[i];
    }
    sort(v + 1,v + n + 1);
    int len = unique(v + 1, v + n + 1) - v - 1;
    rt[0] = build(1,len);
    for (int i = 1; i <= n; ++i) {
        rt[i] = update(getid(a[i],len),1,len,rt[i - 1]);
    }
    for (int i = 1; i <= m; ++i) {
        int l, r, k;
        cin >> l >> r >> k;
        cout << v[query(rt[l - 1],rt[r],1,len,k)] << '\n';
```

~~~
    }
}

# 数学

## 类欧几里得

* $m = \lfloor \frac{an+b}{c} \rfloor$.
* $f(a,b,c,n)=\sum_{i=0}^n\lfloor\frac{ai+b}{c}\rfloor$: 当 $a \ge c$ or $b \ge c$ 时，$f(a,b,c,n)=(\frac{a}{c})n(n+1)/2+(\frac{b}{c})(n+1)+f(a \bmod c,b \bmod c,c,n)$；否则 $f(a,b,c,n)=nm-f(c,c-b-1,a,m-1)$。
* $g(a,b,c,n)=\sum_{i=0}^n i \lfloor\frac{ai+b}{c}\rfloor$: 当 $a \ge c$ or $b \ge c$ 时，$g(a,b,c,n)=(\frac{a}{c})n(n+1)(2n+1)/6+(\frac{b}{c})n(n+1)/2+g(a \bmod c,b \bmod c,c,n)$；否则 $g(a,b,c,n)=\frac{1}{2} (n(n+1)m-f(c,c-b-1,a,m-1)-h(c,c-b-1,a,m-1))$。
* $h(a,b,c,n)=\sum_{i=0}^n\lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \ge c$ or $b \ge c$ 时，$h(a,b,c,n)=(\frac{a}{c})^2 n(n+1)(2n+1)/6 +(\frac{b}{c})^2 (n+1)+(\frac{a}{c})(\frac{b}{c})n(n+1)+h(a \bmod c, b \bmod c,c,n)+2(\frac{a}{c})g(a \bmod c,b \bmod c,c,n)+2(\frac{b}{c})f(a \bmod c,b \bmod c,c,n)$；否则 $h(a,b,c,n)=nm(m+1)-2g(c,c-b-1,a,m-1)-2f(c,c-b-1,a,m-1)-f(a,b,c,n)$。

# 图论

## LCA

+ 倍增

```cpp
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
~~~

计算几何

二维几何：点与向量

```cpp
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

```

```
P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// --------------------------------------------
```

字符串

AC 自动机

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 7;
int z[N][26];
int fail[N];
int res[N];
int cnt = 0;
int re = 0;i
nt mp[N];
void insert(string s) {
    int r = 0;
    re++;
    for (int i = 0; i < s.size(); ++i) {
        if (!z[r][s[i] - 'a']) {
            z[r][s[i] - 'a'] = ++cnt;
        }
        r = z[r][s[i] - 'a'];
    }
    mp[re] = r;
}
void bfs() {
    queue <int> q;
    for (int i = 0; i < 26; ++i) {
        if (z[0][i]) {
            fail[z[0][i]] = 0;
            q.push(z[0][i]);
        }
    }
    while (!q.empty()) {
        int now = q.front();
        q.pop();
        for (int i = 0; i < 26; ++i) {
            if (z[now][i]) {
                fail[z[now][i]] = z[fail[now]][i];
                q.push(z[now][i]);
            }
            else z[now][i] = z[fail[now]][i];
        }
    }
}
void quert(string s) {
    int now = 0;
    int ans = 0;
    for (int i = 0; i < s.size(); ++i) {
        now = z[now][s[i]-'a'];
        for (int j = now;j ; j = fail[j]) {
            res[j]++;;
        }
    }
}int n;
string ss[155];
void solve() {
    memset(z,0,sizeof z);
    memset(res,0,sizeof res);
    memset(fail,0,sizeof fail);
    cnt = 0;
    re = 0;
    for (int i = 1; i <= n; ++i) {
        cin >> ss[i];
        insert(ss[i]);
    }
    bfs();
    string s;
    cin >> s;
    quert(s);
    int tmp = 0;
    int ans;
    for (int i = 1; i <= n; ++i) {
        if (res[mp[i]] > tmp) {
            tmp = res[mp[i]];
            ans = i;
        }
    }
    cout << res[mp[ans]] << '\n';
    for (int i = 1; i <= n; ++i) {
        if (res[mp[i]] == tmp) {
            cout << ss[i] << '\n';
        }
    }

}
int main() {
    ios::sync_with_stdio(0);
    while (cin >> n && n) {
        solve();
    }
}
```

KMP

```
void get(string s) {
    int j = 0, k = -1;
    next[0] = -1;
    while (j < s.length()) {
        if (k == -1 || s[j] == s[k]) {
            j++, k++;
            if (s[j] != s[k]) {
                next[j] = k;
            }
            else next[j] = next[k];
        }
        else k = next[k];
    }
}
```

SA（dc3）

```
//大小开 3 倍
/*
    suffix[i]: 以 i 为起始位置的后缀
    sa[i]: 排名第 i 的后缀的起始位置
    rk[i]: 表示 suffix[i] 的排名
    height[i]: suffix(sa[i-1]) 和 suffix(sa[i]) 的最长公共前缀
        · h[i] = height[rak[i]], h[i] >= h[i-1]-1
        · suffix[i] 和 suffix[j] 之间的最长公共前缀 = min(height[rak[i]+1]...height[rak[j]])
*/

#define F(x) ((x) / 3 + ((x) % 3 == 1 ? 0 : tb))
#define G(x) ((x) < tb ? (x) * 3 + 1 : ((x) - tb) * 3 + 2)

int wa[N], wb[N], wss[N], wv[N], sa[N * 3];
int rk[N], height[N], r[N], lcp[N][30];


int c0(int *r, int a, int b) {
    return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
}

int c12(int k, int *r, int a, int b) {
    if (k == 2)
        return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b + 1);
    return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];
}

void Rsort(int *r, int *a, int *b, int n, int m) {
    for (int i = 0; i < n; i++) wv[i] = r[a[i]];
    for (int i = 0; i < m; i++) wss[i] = 0;
    for (int i = 0; i < n; i++) wss[wv[i]]++;
    for (int i = 1; i < m; i++) wss[i] += wss[i - 1];
    for (int i = n - 1; i >= 0; i--) b[--wss[wv[i]]] = a[i];
}

void dc3(int *r, int *sa, int n, int m) {
    int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
    r[n] = r[n + 1] = 0;
    for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
    Rsort(r + 2, wa, wb, tbc, m);
    Rsort(r + 1, wb, wa, tbc, m);
    Rsort(r, wa, wb, tbc, m);
    for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
        rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
    if (p < tbc) dc3(rn, san, tbc, p);
    else for (i = 0; i < tbc; i++) san[rn[i]] = i;
    for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i] * 3;
    if (n % 3 == 1) wb[ta++] = n - 1;
    Rsort(r, wb, wa, ta, m);
    for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
    for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
        sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
    for (; i < ta; p++) sa[p] = wa[i++];
    for (; j < tbc; p++) sa[p] = wb[j++];
}

void calHeight(int *r, int *sa, int n) {
    int i, j, k = 0;
    for (i = 1; i <= n; i++) rk[sa[i]] = i;
    for (i = 0; i < n; height[rk[i++]] = k)
        for (k ? k-- : 0, j = sa[rk[i] - 1]; r[i + k] == r[j + k]; k++);

    // for (int i = 1; i <= n; ++i) {
    //     dbg(i, height[i]);
    // }
    // 用 getLcp 要去掉下面的注释
    // for (int i = 1; i <= n; ++i) lcp[i][0] = height[i];
    // for (int l = 1; (1 << l) <= n; l++) {
    //     for (int i = 1; i + (1 << l) - 1 <= n; ++i) {
    //         lcp[i][l] = min(lcp[i][l - 1], lcp[i + (1 << (l - 1))][l - 1]);
    //     }
    // }
}

int getLcp(int i, int j, int n) {
    if (i == j) return n - i;
    int l = rk[i], r = rk[j];
    if (l > r) swap(l, r);
```

```cpp
    l++;
    int k = __lg(r - l + 1);
    return min(lcp[l][k], lcp[r - (1 << k) + 1][k]);
}

char s[N];

void solve() {
    int n = 0;
    cin >> s;
    for (int i = 0; s[i]; ++i) {
        r[n++] = s[i];
    }
    r[n] = 0;
    dc3(r, sa, n + 1, 256);
    calHeight(r, sa, n);
    for (int i = 1; i <= n; ++i) {
        cout << sa[i] + 1 << " \n" [i == n];
    }
    for (int i = 1; i <= n; ++i) {
        cout << height[i] << " \n"[i == n];
    }
    // 不同子串个数
    /*
    ll ans = 1ll * n * (n + 1) / 2;
    for (int i = 1; i <= n; ++i) {
        ans -= height[i];
    }
    // 两个串的最长公共子串
    int n = 0;
    scanf("%s",s);
    scanf("%s",t);
    int l = strlen(s);
    s[l] = '!';
    int tag = l;
    for (int i = 0; t[i] ; ++i) {
        s[++l] += t[i];
    }
    for (int i = 0; s[i]; ++i) {
        r[n++] = s[i];
    }
    r[n] = 0;
    dc3(r, sa, n + 1, 256);
    calHeight(r, sa, n);
    int ans = 0;
    for (int i = 1; i <= n; ++i) {
        int x1 = sa[i - 1],x2 = sa[i];
        if ((x1 < tag && x2 > tag) || (x1 > tag && x2 < tag)) {
            ans = max(ans,height[i]);
        }
    }
    printf("%lld",ans);
    //不同公共子串的个数
    ll ans = 0;
    int tmp = 0;
    for (int i = 1; i <= n; ++i) {
        int x1 = sa[i - 1], x2 = sa[i];
        if ((x1 < tag && x2 > tag) || (x1 > tag && x2 < tag)) {
            ans += height[i];
            if (tmp > 0) ans -= min(getLcp(sa[i], sa[tmp], n), getLcp(sa[i - 1], sa[tmp - 1], n));
            tmp = i;
        }
    }
    printf("%lld", ans);
    */
}
```

SAM

```cpp
struct SAM {
    struct state {
```

```
    int len, link;
    map<char, int> next;
};

state st[N * 2];
int sz, last;
int cnt[N * 2];
int siz[N * 2];
int a[N * 2];

void init() {
    st[0].len = 0;
    st[0].link = -1;
    sz = 1;
    last = 0;
}

void extend(char c) {
    int cur = sz++;
    st[cur].len = st[last].len + 1;
    int p = last;
    while (p != -1 && !st[p].next.count(c)) {
        st[p].next[c] = cur;
        p = st[p].link;
    }
    if (p == -1) {
        st[cur].link = 0;
    } else {
        int q = st[p].next[c];
        if (st[p].len + 1 == st[q].len) {
            st[cur].link = q;
        } else {
            int clone = sz++;
            st[clone].len = st[p].len + 1;
            st[clone].next = st[q].next;
            st[clone].link = st[q].link;
            while (p != -1 && st[p].next[c] == q) {
                st[p].next[c] = clone;
                p = st[p].link;
            }
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
    siz[cur]++;
}

void run() { // 求子串出现次数
    ll ans = 0;
    for (int i = 1; i <= sz; ++i) cnt[st[i].len]++;
    for (int i = 1; i <= sz; ++i) cnt[i] += cnt[i - 1];
    for (int i = 1; i <= sz; ++i) a[cnt[st[i].len]--] = i;
    for (int i = sz; i; --i) {
        int p = a[i];
        siz[st[p].link] += siz[p];
        if (siz[p] > 1) ans = max(ans, 1LL * siz[p] * st[p].len);
    }
    cout << ans;
}
/* 不同子串个数
void run1() {
    for (int i = 1; i <= sz; ++i) cnt[st[i].len]++;
    for (int i = 1; i <= sz; ++i) cnt[i] += cnt[i - 1];
    for (int i = 1; i <= sz; ++i) a[cnt[st[i].len]--] = i;
    for (int i = sz; i >= 0; --i) {
        int p = a[i];
        siz[p] = 1;
        for (auto j : st[p].next) {
            siz[p] += siz[j.second];
        }
    }
```

```cpp
        cout << siz[0] - 1;
    }
    void run2(int x) {
        dp[x] = 1;
        for (auto i : st[x].next) {
            if (!dp[i.second]) run2(i.second);
            dp[x] += dp[i.second];
        }
    }
    dp[0] - 1;
    void run3() {
        ll ans = 0;
        for (int i = 1; i <= sz; ++i) {
            ans += st[i].len - st[st[i].link].len;
        }
        cout << ans;
    }
    */

    string lcs(string t) { //最长公共子串
        int v = 0, l = 0, mx = 0, mx_end = 0;
        for (int i = 1; i <= t.size(); ++i) {
            while (v && !st[v].next.count(t[i - 1])) {
                v = st[v].link;
                l = st[v].len;
            }
            if (st[v].next.count(t[i - 1])) {
                v = st[v].next[t[i - 1]];
                l++;
            }
            if (l > mx) {
                mx = l;
                mx_end = i;
            }
        }
        return t.substr(mx_end - mx + 1, mx);
    }
} sam;


void solve() {
    string s;
    cin >> s;
    sam.init();
    for (char i: s) {
        sam.extend(i);
    }
}
```

杂项

STL

- copy

```cpp
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```