# Algorithm Templates

Nephrenn

2024 年 10 月 28 日

# 目录

# 1 数据结构

## 1.1 树状数组

```
template <typename T>
struct Fenwick {
// 1-indexed
private:
    int n;
    std::vector<T> tr;
public:
    Fenwick (int _n = 0) {
        init(_n);
    }
    void init (int _n) {
        n = _n;
        tr.resize(n + 1);
        tr.assign(n + 1, T(0));
    }
    void add (int p, const T &v) {
        for (int i = p; i <= n; i += i & -i) {
            tr[i] += v;
        }
    }
    T sum (int p) {
        T res (0);
        for (int i = p; i >= 1; i -= i & -i) {
            res += tr[i];
        }
        return res;
    }
};
```

## 1.2　并查集

```cpp
class DSU {
// 1-indexed
private:
    std::vector<int> f, sz;
public:
    DSU () {}
    DSU (int _n) {
        init(_n);
    }

    void init (int _n) {
        f.resize(_n + 1);
        std::iota(f.begin(), f.end(), 0);
        sz.assign(_n + 1, 1);
        sz[0] = 0;
    }

    int find (int x) {
        while (x != f[x]) {
            x = f[x] = f[f[x]];
        }
        return x;
    }

    bool same (int x, int y) {
        return find(x) == find(y);
    }

    bool merge (int x, int y) {
        // merge y to x
        x = find(x), y = find(y);
        if (x == y) return false;
        sz[x] += sz[y];
        f[y] = x;
        return true;
    }

    int size (int x) {
        return sz[find(x)];
    }
};
```

## 1.3 FHQ Treap

```
// 洛谷 P3369 【模板】 普通平衡树
struct FHQTreap
{
        struct FHQTreap_node
        {
                int ls, rs, key, val, sz;
        }tree[MAXN];
        int tot = 0;
        int root = 0, t1, t2, t3;

        int new_node(int v)
        {
                tree[++tot] = {0, 0, (int)rnd(), v, 1};
                return tot;
        }

        void push_up(int u)
        {
                tree[u].sz = tree[tree[u].ls].sz +
                ↪   tree[tree[u].rs].sz + 1;
        }
```

```
void split_by_val(int u, int v, int &x, int &y)
{
        if(!u)
        {
                x = y = 0;
                return;
        }

        if(tree[u].val > v)
        {
                y = u;
                split_by_val(tree[u].ls, v, x,
                ↪   tree[u].ls);
        }
        else
        {
                x = u;
                split_by_val(tree[u].rs, v,
                ↪   tree[u].rs, y);
        }

        push_up(u);
}

int merge(int x, int y)
{
        if(!x || !y) return x + y;

        if(tree[x].key > tree[y].key)
        {
                tree[x].rs = merge(tree[x].rs, y);
                push_up(x);
                return x;
```

```
		}
		else
		{
			tree[y].ls = merge(x, tree[y].ls);
			push_up(y);
			return y;
		}
	}

	void insert(int v)
	{
		split_by_val(root, v, t1, t2);
		root = merge(merge(t1, new_node(v)), t2);
	}

	void erase(int v)
	{
		split_by_val(root, v, t1, t2);
		split_by_val(t1, v - 1, t1, t3);
		t3 = merge(tree[t3].ls, tree[t3].rs);
		root = merge(merge(t1, t3), t2);
	}

	int query_rnk(int v)
	{
		split_by_val(root, v - 1, t1, t2);
		int res = tree[t1].sz + 1;
		root = merge(t1, t2);
		return res;
	}

	int query_kth(int k)
	{
        int u = root;
        while(u)
        {
                int t = tree[tree[u].ls].sz + 1;
                if(t == k) break;
                else if(k < t) u = tree[u].ls;
                else
                {
                        k -= t;
                        u = tree[u].rs;
                }
        }
        return tree[u].val;
}

int query_pre(int u, int v)
{
        if(!u) return -INF;
        if(tree[u].val < v)
        {
                int res = query_pre(tree[u].rs, v);
                return res == -INF ? tree[u].val :
                ↪  res;
        }
        else
        {
                return query_pre(tree[u].ls, v);
        }
}

int query_nxt(int u, int v)
{
        if(!u) return INF;
```

```
                if(tree[u].val > v)
                {
                        int res = query_nxt(tree[u].ls, v);
                        return res == INF ? tree[u].val : res;
                }
                else
                {
                        return query_nxt(tree[u].rs, v);
                }
        }
}Treap;
```

## 1.4 ST 表

```
const int MAXN = 1e5 + 10, N = 1e5;;
int lg[MAXN];
void init () {
    lg[1] = 0;
    for (int i = 2; i <= N; ++i) lg[i] = lg[i / 2] + 1;
}

struct ST {
    // 1-indexed
    int n;
    std::vector<std::vector<int>> st;

    ST (int _n, std::vector<int> a) : n(_n) {
        st.resize(n + 1);
        st.assign(n + 1, std::vector<int>(lg[n] + 1, 0));

        build(a);
    }

    void build (std::vector<int> a) {
        for (int i = 1; i <= n; ++i) st[i][0] = a[i];
        for (int j = 1; j <= lg[n]; ++j) {
            for (int i = 1; i + (1 << j) - 1 <= n; ++i) {
                st[i][j] = std::max(st[i][j - 1], st[i + (1 <<
                ↪  (j - 1))][j - 1]);
            }
        }
    }

    int query (int l, int r) {
        int s = lg[r - l + 1];
        return std::max(st[l][s], st[r - (1 << s) + 1][s]);
    }
};
```

# 2 数学

## 2.1 组合数

```
constexpr int N = 1e5;
constexpr i64 P = 1e9 + 7;
typedef ModInt<P> Z;
Z fac[N + 10], ifac[N + 10];
void init () {
    fac[0] = 1;
    for (int i = 1; i <= N; ++i) {
        fac[i] = fac[i - 1] * Z(i);
    }
    ifac[N] = ModInt<P>::inv(fac[N]);
    for (int i = N - 1; i >= 0; --i) {
        ifac[i] = ifac[i + 1] * Z(i + 1);
    }
}
Z binom (int n, int m) {
    return fac[n] * ifac[m] * ifac[n - m];
}
```

### 2.2 卢卡斯定理

```
int lucas (int n, int m) {
    return m ? binom(n % P, m % P) * lucas(n / P, m / P) % P :
     ↪  1;
}
```

### 2.3 线性筛素数

```
const int N = 1e5;
bool not_prime[N + 10];
int prime[N + 10], tot = 0;
for (int i = 2; i <= N; ++i) {
    if (not_prime[i] == 0) prime[++cnt] = i;
    for (int j = 1; j <= cnt && i * prime[j] <= N; ++j) {
        not_prime[i * prime[j]] = 1;
        if (i % prime[j] == 0) break;
    }
}
```

### 2.4 线性求逆元

```
const int N = 1e5, P = 998244353;
int inv[N + 10];
inv[1] = 1;
for (int i = 2; i <= n; ++i) {
    inv[i] = (P - P / i) * inv[P % i] % P;
}
```

# 3 字符串

## 3.1 前缀函数

```cpp
// string: 0-indexed
// f: 1-indexed

std::vector<int> kmp(std::string s) {
    int n = s.size();
    std::vector<int> f(n + 1);
    for (int i = 1, j = 0; i < n; ++i) {
        while (j && s[i] != s[j]) j = f[j];
        j += (s[i] == s[j]);
        f[i + 1] = j;
    }
    return f;
}

// 洛谷 P3375 【模板】KMP
auto z = kmp(s2);
for (int i = 0, j = 0; i < s1.size(); ++i) {
    while (j && s2[j] != s1[i]) j = z[j];
    j += (s2[j] == s1[i]);
    if (j == s2.size()) {
        std::cout << i + 2 - s2.size() << "\n";
    }
}

for (int i = 1; i <= s2.size(); ++i) {
    std::cout << z[i] << " ";
}
std::cout << "\n";
```

# 4 杂项

## 4.1 取模类

```
using i64 = long long;
template <i64 mod>
class ModInt {
private:
    i64 x;
public:
    ModInt (const i64 _x = 0) : x(_x % mod) {}
    friend std::ostream &operator << (std::ostream &os, const
    ↪   ModInt &z) {
        os << z.x;
        return os;
    }
    friend std::istream &operator >> (std::istream &is, ModInt
    ↪   &z) {
        i64 v;
        is >> v;
        z = ModInt(v);
        return is;
    }
    explicit operator i64() {
        return x;
    }
    ModInt &operator += (const ModInt &rhs) {
        x = (x + rhs.x) % mod;
        return *this;
    }
    friend ModInt operator + (const ModInt &lhs, const ModInt
    ↪   &rhs) {
        ModInt res = lhs;
        res += rhs;
        return res;
    }
    ModInt &operator -= (const ModInt &rhs) {
        x = (x - rhs.x) % mod;
        x = (x < 0 ? x + mod : x);
        return *this;
    }
    friend ModInt operator - (const ModInt &lhs, const ModInt
    ↪   &rhs) {
        ModInt res = lhs;
        res -= rhs;
        return res;
    }
    ModInt &operator *= (const ModInt &rhs) {
        x = (x * rhs.x) % mod;
        return *this;
    }
    friend ModInt operator * (const ModInt &lhs, const ModInt
    ↪   &rhs) {
        ModInt res = lhs;
        res *= rhs;
        return res;
    }
    static ModInt quick_pow (ModInt a, i64 b) {
        ModInt res(1);
        for ( ; b; a *= a, b >>= 1) {
            if (b & 1) res *= a;
        }
        return res;
    }
    static ModInt inv (ModInt a) {
        // guarantee that mod is prime
        return quick_pow(a, mod - 2);
    }
    ModInt &operator /= (const ModInt &rhs) {
        return *this *= inv(rhs);
    }
    friend ModInt operator / (const ModInt &lhs, const ModInt
    ↪   &rhs) {
        ModInt res = lhs;
        res /= rhs;
        return res;
    }
};
constexpr i64 P = 1e9 + 7;
typedef ModInt<P> Z;
```