Team notebook

March 10, 2019

Contents

1 Estructuras

1.1 Segment Tree

```
struct node {
    int mn, l, r;
    node *left, *right;
    node(int l, int r, int* A) : l(l), r(r) {
        if(l == r)
            mn = A[l];
        else {
            int mi = (l + r)/2;
            left = new node(l, mi, A);
            right = new node(r + 1, mi, A);
            mn = min(left->mn, right->mn);
        }
    }
    void upd(int p, int v) {
        if(r < p || p < l)
            return;
        if(l == r) {
            mn = v;
            return;
        }
        left->upd(p, v), right->upd(p, v);
        mn = min(left->mn, right->mn);
    }
    int qry(int rl, int rr) {
        if(rr < l || r < rl)
            return NE;
        if(rl <= l && r <= rr)
            return mn;
        return min(left->qry(rl, rr),
            right->qry(rl, rr));
    }
};
```

1.2 Wavelet Tree

```
struct wnode {
    wnode *left, *right;
    int lo, hi;
    vector<int> c;

    wnode(int lo, int hi, int* st, int* en) :
        lo(lo), hi(hi) {
        if(hi == lo || st == en)
            return;
        int mi = (lo + hi)/2;
        auto f = [mi](int x) { return x <= mi;
            };
        c.push_back(0);
        for(auto it = st; it != en; ++it)
            c.push_back(c.back() + f(*it));
        auto it = stable_partition(st, en, f);
        left = new wnode(lo, mi, st, it);
        right = new wnode(mi + 1, hi, it, en);
    }

    int kth(int L, int R, int k) {
        if(lo == hi)
            return lo;
        int der = c[R], izq = c[L - 1], tol =
            der - izq;
        if(tol >= k)
            return left->kth(izq + 1, der, k);
        return right->kth(L - izq, R - der, k
            - tol);
    }
};
```

2 Flujo

2.1 Dinic

```
const int MAXN = 5000;
const int INF = 1e9;

int dist[MAXN], ptr[MAXN], src, dst;
```

```
struct Edge {
    int to, rev, f, cap;
        Edge(int to, int rev, int f, int cap)
            : to(to), rev(rev), f(f), cap(cap);
};

vector<Edge> G[MAXN];


void addEdge(int u, int v, int cap) {
    G[u].push_back(Edge(v, G[v].size(), 0,
        cap));
    G[v].push_back(Edge(u, G[u].size() - 1, 0,
        0));
}

bool bfs() {
    queue<int> q({src});
    memset(dist, -1, sizeof dist);
    dist[src] = 0;
    while(!q.empty() && dist[dst] == -1) {
        int u = q.front();
        q.pop();
        for(auto e : G[u]) {
            int v = e.to;
            if(dist[v] == -1 && e.f < e.cap) {
                dist[v] = dist[u] + 1;
                q.push(v);
            }
        }
    }
    return dist[dst] != -1;
}

int dfs(int u, int f)
    if(u == dst || !f)
        return f;
    for(int &i = ptr[u]; i < G[u].size(); i++)
        {
        Edge &e = G[u][i];
        int v = e.to;
        if(dist[v] != dist[u] + 1)
            continue;
        if(int df = dfs(v, min(f, e.cap -
            e.f))) {
            e.f += df;
            G[v][e.rev].f -= df;
            return df;
        }
    }
    return 0;
}

long long dinic() {
    long long mf = 0;
    while(bfs()) {
        memset(ptr, 0, sizeof ptr);
        while(long long pushed = dfs(src, INF))
            mf += pushed;
    }
    return mf;
}
```

3 Mate

3.1 CRT

```
template<typename T> void euclid(T a, T b, T
    &x, T &y) {
    if(!b) {x = 1, y = 0; return;}
    euclid(b, a % b, x, y);
    T x1 = y, y1 = x - (a/b)*y;
    x = x1, y = y1;
}

template<typename T> T crt(T x1, T m1, T x2,
    T m2) {
    T d = __gcd(m1, m2), r, s;
    if(x1 % d != x2 % d)
        return -1;
    m1 /= d, m2 /= d;
    T mod = d*m1*m2, a1 = ((m1*r)%mod*x2)%mod,
        a2 = ((m2*s)%mod*x1)%mod;
    T y = (a1 + a2)%mod;
    if(y < 0)
        y += mod;
    return y;
}
```