memset0's Notebook

方知暮然回首之时
那人却已不在灯火阑珊处

LOJ6515 「雅礼集训 2018 Day10」贪玩蓝月

2019-01-03 | 题解

考虑离线做法，每个物品存在的时间一定是一段连续的区间。建一棵线段树按时间分治，乱搞一下。

考虑在线做法：有一个部分分是只会在一端进行插入删除操作，那么我们开个栈，插入元素就加进去，删除就弹出。复杂度 $O(nm)$ ；既然我们需要在两端操作，那么我们只要维护两个这样的栈即可，查询时把两边的 dp 数组合并。暴力合并是 $O(m^2)$ 的，会 TLE ，所以我们把一个数组放在线段树上，枚举另一个数组的每一个数，做一下区间查询即可。同时有个问题即有可能在前端删除从后端插入的元素，这种时候我们暴力重构两个栈，各放一半的元素，就能保证复杂度。最后总时间复杂度 $O(n\log n\times m\log m)$ ，可以通过此题。

由于笔者在做题时把合并的 $O(m^2)$ 算成了 $O(m)$ 所以去写了在线做法，还好想出了线段树才捡回一条命 233。

代码：

关于我
友情链接
文章聚合

memset0's Notebook

方知蓦然回首之时
那人却已不在灯火阑珊处

关于我
友情链接
文章聚合


```cpp
// ================================
//   author: memset0
//   date: 2019.01.03 10:25:41
//   website: https://memset0.cn/
// ================================
#include <bits/stdc++.h>
#define ll long long
namespace ringo {
template <class T> inline void read(T &x) {
    x = 0; register char c = getchar(); register bool f = 0;
    while (!isdigit(c)) f ^= c == '-', c = getchar();
    while (isdigit(c)) x = x * 10 + c - '0', c = getchar();
    if (f) x = -x;
}
template <class T> inline void print(T x) {
    if (x < 0) putchar('-'), x = -x;
    if (x > 9) print(x / 10);
    putchar('0' + x % 10);
}
template <class T> inline void print(T x, char c) { print(x), putchar(c); }

const int N = 5e4 + 10, M = 500;
int n, mod, w, v, l, r, id, c_1, c_2;
ll ans, now[M];

std::deque <std::pair <int, int> > deque;
std::vector <std::pair <int, int> > l_queue, r_queue;

inline void maxd(ll &a, ll b) { if (b > a) a = b; }

struct stack {
    int top;
    ll dp[N][M];
    std::pair <int, int> index[N];
    inline stack() { memset(dp[0], -1, sizeof(dp[0])); dp[0][0] = 0; }
    inline int size() { return top; }
```

```
    inline ll* last() { return dp[top]; }
    inline std::pair <int, int> pop() { return index[top--]; }
    void push(int w, int v) {
        index[++top] = std::make_pair(w, v);
        auto f = dp[top], g = dp[top - 1];
        for (int i = 0; i < mod; i++) f[i] = g[i];
        for (int i = 0; i < mod; i++)
            if (~g[i])
                maxd(f[(i + w) % mod], g[i] + v);
    }
} L, R;

void merge(ll *f, ll *g, ll *ans) {
    for (int i = 0; i < mod; i++) ans[i] = std::max(f[i], g[i]);
    for (int i = 0; i < mod; i++)
        for (int j = 0; j < mod; j++)
            if (~f[i] && ~g[j])
                maxd(ans[(i + j) % mod], f[i] + g[j]);
}

void maintain(int flag) {
    if (L.size() && R.size()) return;
    if (!L.size() && !R.size()) return;
    while (L.size()) deque.push_back(L.pop());
    while (R.size()) deque.push_front(R.pop());
    int limit = deque.size() >> 1;
    if (flag == 0 && limit == 0) ++limit;
    if (flag == 1 && deque.size() - limit == 0) --limit;
    while (limit--) l_queue.push_back(deque.front()), deque.pop_front();
    while (deque.size()) r_queue.push_back(deque.back()), deque.pop_back();
    std::reverse(l_queue.begin(), l_queue.end());
    std::reverse(r_queue.begin(), r_queue.end());
    for (auto i : l_queue) L.push(i.first, i.second);
    for (auto i : r_queue) R.push(i.first, i.second);
    l_queue.clear(), r_queue.clear(), deque.clear();
}
```

```
struct node {
    int l, r, mid;
    ll max;
} p[M << 2];

void build(int u, int l, int r) {
    p[u].l = l, p[u].r = r, p[u].mid = (l + r) >> 1;
    if (l == r) return;
    build(u << 1, l, p[u].mid);
    build(u << 1 | 1, p[u].mid + 1, r);
}

void modify(int u, int l, int r, ll *a) {
    if (p[u].l == p[u].r) { p[u].max = a[l]; return; }
    modify(u << 1, l, p[u].mid, a);
    modify(u << 1 | 1, p[u].mid + 1, r, a);
    p[u].max = std::max(p[u << 1].max, p[u << 1 | 1].max);
}

ll query(int u, int l, int r) {
    if (p[u].l == l && p[u].r == r) return p[u].max;
    if (r <= p[u].mid) return query(u << 1, l, r);
    if (l > p[u].mid) return query(u << 1 | 1, l, r);
    return std::max(query(u << 1, l, p[u].mid), query(u << 1 | 1, p[u].mid +
}

ll merge(int l, int r, ll *f, ll *g) {
    modify(1, 0, mod - 1, g);
    ll ans = -1, another;
    for (int i = l; i <= r; i++) ans = std::max(ans, std::max(f[i], g[i]));
    for (int i = 0, _l, _r; i < mod; i++)
        if (~f[i]) {
            _l = (l - i + mod) % mod;
            _r = (r - i + mod) % mod;
            another = -1;
            if (_l <= _r) another = query(1, _l, _r);
            else another = std::max(query(1, 0, _r), query(1, _l, mod - 1));
```

```
            if (~another) ans = std::max(ans, another + f[i]);
        }
    return ans;
}

void main() {
    read(id), read(n), read(mod);
    build(1, 0, mod - 1);
    for (int i = 1; i <= n; i++) {
        while (c_1 = getchar(), !isupper(c_1));
        while (c_2 = getchar(), !isupper(c_2));
        if (c_1 == 'I' && c_2 == 'F') {
            read(w), read(v), w %= mod;
            L.push(w, v);
        } else if (c_1 == 'I' && c_2 == 'G') {
            read(w), read(v), w %= mod;
            R.push(w, v);
        } else if (c_1 == 'D' && c_2 == 'F') {
            if (!L.size()) maintain(0);
            L.pop();
        } else if (c_1 == 'D' && c_2 == 'G') {
            if (!R.size()) maintain(1);
            R.pop();
        } else {
            read(l), read(r);
            print(merge(l, r, L.last(), R.last()), '\n');
        }
    }
}

} signed main() { return ringo::main(), 0; }
```

巧妙的思路 线段树分治

memset0's Notebook

方知暮然回首之时
那人却已不在灯火阑珊处

用户名

邮箱

网址（选填）

可以在这里写评论哦 ~

提交评论

LOJ6514 「雅礼集训 2018 Day10」文明
上一篇 «

None
» 下一篇

关于我
友情链接
文章聚合

memset0's Notebook

方知暮然回首之时
那人却已不在灯火阑珊处

关于我
友情链接
文章聚合