区间覆盖最值问题

题目

给定数列范围$[1,n]$,权值为a_i.有m次操作,每次操作能将给定区间$[l_i,r_i]$的权值变成v_i.

求m次操作后数列总权值的最大(最小)值

做法

法一：

显然是个区间覆盖问题.所以我们可以通过差分 + 最大(最小)堆，然后统计答案的时候从左到右扫一遍.

法二：

由于要求极值,那不难知道,对于权值v_i较大的区间,较小的v_i是不可能覆盖掉它的,否则答案不是最值.

或者我们将操作的权值从大到小(从小到大)排序一遍,由于每次操作的区间权值v_i都是当前最大值,

所以后面的操作无法覆盖它,所以我们可以缩点一般,把这个区间合成一个点

因此我们可以通过并查集来维护这个缩点.

每次操作都暴力遍历操作区间$[l_i,r_i]$,将当前区间合并.

Code

法一:

```cpp
#include <bits/stdc++.h>
#define int long long
#define endl '\n'
#define LL __int128
using namespace std;
int qpow(int a, int b, int p) {int ret = 1; for(a %= p; b; b >>= 1, a = a * a % p) if(b & 1) ret = ret * a % p; return ret; }
int qpow(int a,int b) {int ret = 1; for(; b; b >>= 1, a *= a) if(b & 1) ret *= a; return ret; }
int gcd(int x,int y) {return y ? gcd(y, x % y) : x; }
pair<int,int> exgcd(int a,int b) { if(!b) return {1, 0}; pair<int,int> ret = exgcd(b, a % b); return {ret.second, ret.first - a / b * ret.second }; }
int lcm(int x,int y){ return x / gcd(x, y) * y; }
const int N = 5 + 1e5;
map<char, int> add[N], del[N];

signed main() {
    int n, m; cin >> n >> m;
    string s; cin >> s; s = ' ' + s;
    while (m--) {
        int l, r; char c; cin >> l >> r >> c;
        add[l][c]++; del[r + 1][c]++;
    }
    map<char, int> mp;
    for (int i = 1; i <= n; i++) {
        for (auto &v: add[i]) {
            mp[v.first] += v.second;
```

```
        }
        for (auto &v: del[i]) {
            mp[v.first] -= v.second;
            if (!mp[v.first]) mp.erase(v.first);
        }
        if (mp.size()) {
            auto u = mp.end(); u--;
            if ((*u).second) s[i] = max(s[i], (*u).first);
        }
    }
    int ans = 0;
    for (int i = 1; i <= n; i++) ans += s[i];
    cout << ans << endl;
}
```

法二:

```
#include <bits/stdc++.h>
#define int long long
#define endl '\n'
#define LL __int128
using namespace std;
int qpow(int a, int b, int p) {int ret = 1; for(a %= p; b; b >>= 1, a = a * a % p) if(b & 1) ret = ret * a % p; return ret; }
int qpow(int a,int b) {int ret = 1; for(; b; b >>= 1, a *= a) if(b & 1) ret *= a; return ret; }
int gcd(int x,int y) {return y ? gcd(y, x % y) : x; }
pair<int,int> exgcd(int a,int b) { if(!b) return {1, 0}; pair<int,int> ret = exgcd(b, a % b); return {ret.second, ret.first - a / b * ret.second }; }
int lcm(int x,int y){ return x / gcd(x, y) * y; }
struct T {
    int l, r; char c;
    friend bool operator < (T x, T y) {
        return x.c < y.c;
    }
};
const int N = 5 + 1e7;
int fa[N];
int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }
signed main() {
    ios :: sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int n, m; cin >> n >> m;
    string s; cin >> s; s = ' ' + s;
    priority_queue<T> q;
    for (int i = 0; i < m; i++) {
        int l, r; char c; cin >> l >> r >> c;
        q.push({l, r, c});
    }
    for (int i = 1; i <= n + 1; i++) fa[i] = i;
    int ans = 0;
    while (q.size()) {
        auto u = q.top(); q.pop();
        for (int i = u.l; i <= u.r; i = fa[i + 1]) {
            fa[i] = find(i + 1);
            if (s[i] < u.c) s[i] = u.c;
        }
    }
```

```
    for (int i = 1; i <= n; i++) ans += s[i];
    cout << ans << endl;
}
```

```
#include <bits/stdc++.h>
#define int long long
#define endl '\n'
#define LL __int128
using namespace std;
int qpow(int a, int b, int p) {int ret = 1; for(a %= p; b; b >>= 1, a = a * a % p) if(b & 1) ret = ret * a % p; return ret; }
int qpow(int a,int b) {int ret = 1; for(; b; b >>= 1, a *= a) if(b & 1) ret *= a; return ret; }
int gcd(int x,int y) {return y ? gcd(y, x % y) : x; }
pair<int,int> exgcd(int a,int b) { if(!b) return {1, 0}; pair<int,int> ret = exgcd(b, a % b); return {ret.second, ret.first - a / b * ret.second }; }
int lcm(int x,int y){ return x / gcd(x, y) * y; }
struct T {
    int l, r; char c;
    friend bool operator < (T x, T y) {
        return x.c < y.c;
    }
};
const int N = 5 + 1e7;
int fa[N];
int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }
signed main() {
    ios :: sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int n, m; cin >> n >> m;
    string s; cin >> s; s = ' ' + s;
    priority_queue<T> q;
    for (int i = 0; i < m; i++) {
        int l, r; char c; cin >> l >> r >> c;
        q.push({l, r, c});
    }
    for (int i = 1; i <= n; i++) fa[i] = i;
    int ans = 0;
    int pre = 0;
    while (q.size()) {
        auto u = q.top(); q.pop();
        //cout << u.l << "," << u.r << "," << u.c << endl;
        stack<pair<int, char>> st;
        for (int i = u.l, pre = 0; i <= u.r; pre = i, i++) {
            if (find(i) == i) {
                st.push({i, u.c});
            } else {
                while (st.size()) {
                    if (!pre) pre = st.top().first;
                    fa[st.top().first] = find(pre);
                    s[st.top().first] = max(s[st.top().first], st.top().second);
                    st.pop();
                }
                i = fa[i];
            }
        }
        while (st.size()) {
```

```
            fa[st.top().first] = find(u.r);
            s[st.top().first] = max(s[st.top().first], st.top().second);
            st.pop();
        }
    }
    for (int i = 1; i <= n; i++) ans += s[i];
    cout << ans << endl;
}
```