CTU.DAISY Notebook (2020)

Contents

1 Back Tracking

1.1 Combination

```cpp
#include <bits/stdc++.h>
#define fto(i, x, y) for (int i = (x); i <= (y); ++i)

using namespace std;

int n, k, a[100];

void _try(int i)
{
    if (i > k)
    {
        fto(j, 1, k) cout << a[j] << " ";
        cout << endl;
        return;
    }
    fto(j, a[i - 1] + 1, n - k + i)
    {
        a[i] = j;
        _try(i + 1);
    }
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin.precision(10);
    cout << fixed;
    cin >> n >> k;
    _try(1);
    return 0;
}
```

1.2 Generate binary string

```cpp
void generate(int i)
{
    if (i > n)
    {
        for (int k = 1; k <= n; ++k)
            cout << a[k];
        cout << endl;
        return;
    }
    for (int j = 0; j <= 1; ++j)
    {
        a[i] = j;
        generate(i + 1);
    }
}
```

1.3 Permutation

```cpp
int main()
{

    scanf("%d", &n);

    fto(i, 0, n - 1) a.pb(i + 1);

    do
    {
        fto(i, 0, n - 1) printf("%d", a[i]);
        printf("\n");
    } while (next_permutation(a.begin(), a.end()));

    return 0;
}
```

2 Data Structures

2.1 BIT find Min Max in Segment

```cpp
#include <bits/stdc++.h>
#define maxN 60005
#define oo 1000000007

using namespace std;

int n, q;
int a[maxN];
pair<int, int> f[maxN];

void update(int x, int val)
{
    while (x <= n)
    {
        f[x].first = max(f[x].first, val);
        f[x].second = min(f[x].second, val);
        x += x & (-x);
    }
}

int get(int l, int r)
{
    int mx = 0, mn = oo;
    while (r >= l)
    {
        if (r - (r & (-r)) >= l)
        {
            mx = max(mx, f[r].first);
            mn = min(mn, f[r].second);
            r -= r & (-r);
        }
        else
        {
            mx = max(mx, a[r]);
            mn = min(mn, a[r]);
            --r;
        }
    }
    return mx - mn;
}

int main()
{
    cin >> n >> q;
    for (int i = 1; i <= n; ++i)
        f[i].second = oo;
    for (int i = 1; i <= n; ++i)
    {
        cin >> a[i];
        update(i, a[i]);
    }

    while (q--)
    {
        int l, r;
        cin >> l >> r;
        cout << get(l, r) << endl;
    }
}
```

2.2 BIT calculate Sum in Segment

```cpp
#include <bits/stdc++.h>
#define maxN 300005

using namespace std;

int n, q;
int f[maxN], ans = 0, a[maxN];

void update(int x, int inc)
{
    while (x <= n)
    {
        f[x] += inc;
        x += x & (-x);
    }
}

int get(int x)
{
    int ans = 0;
    while (x > 0)
    {
        ans += f[x];
        x -= x & (-x);
    }
    return ans;
}

int main()
{
    cin >> n;
    for (int i = 1; i <= n; ++i)
    {
        cin >> a[i];
        update(i, a[i]);
    }
    cin >> q;
    while (q--)
    {
        int t, i, l, r, k;
        cin >> t;
        if (t == 1)
        {
            cin >> i >> k;
            update(i, k);
        }
        else
        {
            cin >> l >> r;
            cout << get(r) - get(l - 1) << endl;
        }
    }
}
```

2.3 IT with one query's type

```cpp
// Update [l, r] to k
#include <bits/stdc++.h>
#define maxN 500005
#define oo 1000000007

using namespace std;

int n, q;
int a[maxN];

struct Node
{
    int val;
    int lazy;
    Node()
    {
        val = 0, lazy = -1;
    }
    Node(int v, int l)
    {
        val = v, lazy = l;
    }
};

Node node[4 * maxN];

void initIT(int k, int l, int r)
{
    if (l == r)
    {
        node[k] = Node(a[l], -1);
        return;
    }
    int m = (l + r) / 2;
    initIT(2 * k, l, m);
    initIT(2 * k + 1, m + 1, r);
    node[k].val = node[2 * k].val + node[2 * k + 1].val;
    node[k].lazy = -1;
}

void down(int k, int l, int r)
{
    //Lazy Update || Lazy Propagation
    int v = node[k].lazy;
    if (v != -1)
    {
        int m = (l + r) / 2;
        // Increasing each value in range [a, b] by x.
        node[2 * k].val += (m - l + 1) * v;
        node[2 * k + 1].val += (r - m) * v;

        // Setting each value in range [a, b] to x.
        // node[2 * k].val = (m - l + 1) * v;
        // node[2 * k + 1].val = (r - m) * v;
        node[2 * k].lazy = node[2 * k + 1].lazy = v;

        // Just increase each value in range [a, b] by x and find Min, Max
        // Keep value not SUM
        // node[2 * k].val += v;
        // node[2 * k + 1].val += v;
        // node[2 * k].lazy += v;
        // node[2 * k + 1].lazy += v;

        // Like above but set each value in range [a, b] to x.
        // node[2 * k].val = v;
        // node[2 * k + 1].val = v;
        // node[2 * k].lazy += v;
        // node[2 * k + 1].lazy += v;
        node[k].lazy = -1;
    }
}

void updateIT(int k, int l, int r, int u, int v, int val)
{
    // (u v) l, r (u v)
    if (v < l || r < u)
        return;
    // u (l, r) v
    if (u <= l && r <= v)
    {
        // Increasing each value in range [a, b] by x.
        node[k].val += (r - l + 1) * val;

        // Setting each value in range [a, b] to x.
        // node[k].val = (r - l + 1) * val;
        node[k].lazy = val;

        // Just increase each value in range [a, b] by x and find Min, Max
        // Keep value not SUM
        // node[k].val += val;
        // node[k].lazy += val;

        // Like above but set each value in range [a, b] to x.
        // node[k].val = val;
        // node[k].lazy += val;
        return;
    }

    int m = (l + r) / 2;
    down(k, l, r);
    updateIT(2 * k, l, m, u, v, val);
    updateIT(2 * k + 1, m + 1, r, u, v, val);

    node[k].val = node[2 * k].val + node[2 * k + 1].val;
}

int queryIT(int k, int l, int r, int u, int v)
{

    // (u v) l, r (u v)
    if (v < l || r < u)
        return 0;
    // u (l, r) v
    if (u <= l && r <= v)
        return node[k].val;

    int m = (l + r) / 2;
    down(k, l, r);
```

```
    return queryIT(2 * k, l, m, u, v) + queryIT(2 * k + 1, m + 1, r, u, v);
}

int main()
{
    cin >> n >> q;
    for (int i = 1; i <= n; ++i)
        cin >> a[i];
    initIT(1, 1, n);
    while (q--)
    {
        int t, u, v, val;
        cin >> t >> u >> v;
        if (t == 0)
        {
            cin >> val;
            updateIT(1, 1, n, u, v, val);
        }
        else
        {
            cout << queryIT(1, 1, n, u, v) << endl;
        }
    }
}
```

2.4 IT with 2 queries' type

```
/*
Your task is to maintain an array of n values and efficiently process the following types of queries:
    1. Increase each value in range [a, b] by x.
    2. Set each value in range [a, b] to x.
    3. Calculate the sum of values in range [a, b].
*/
#include <bits/stdc++.h>
#define ll long long
#define maxN 200005

using namespace std;

int n, q;
int a[maxN];

struct node
{
    ll sum, inc;
    bool set;
};

node st[4 * maxN];

void initIT(int k, int l, int r)
{
    if (l == r)
    {
        st[k] = {a[l], 0, 0};
        return;
    }
    int m = (l + r) / 2;
    initIT(2 * k, l, m);
    initIT(2 * k + 1, m + 1, r);
    st[k].sum = st[2 * k].sum + st[2 * k + 1].sum;
}

void app(int k, ll x, bool is_set, int u, int v)
{
    if (is_set)
    {
        st[k].sum = 0;
        st[k].inc = 0;
        st[k].set = 1;
    }
    st[k].sum += x * (v - u + 1);
    st[k].inc += x;
}

void push_down(int k, int l, int m, int r)
{
    // Lazy Update
    app(2 * k, st[k].inc, st[k].set, l, m);
    app(2 * k + 1, st[k].inc, st[k].set, m + 1, r);

    st[k].set = false;
    st[k].inc = 0;
}
void update(int k, int l, int r, int u, int v, ll x, bool is_set)
{
    if (v < l || r < u)
        return;
    if (u <= l && r <= v)
    {
        app(k, x, is_set, l, r);
        return;
    }
    int m = (l + r) / 2;
    push_down(k, l, m, r);
    update(2 * k, l, m, u, v, x, is_set);
    update(2 * k + 1, m + 1, r, u, v, x, is_set);
    st[k].sum = st[2 * k].sum + st[2 * k + 1].sum;
}

ll query(int k, int l, int r, int u, int v)
{
    // (u v) l r
    if (v < l || r < u)
        return 0LL;

    if (u <= l && v >= r)
        return st[k].sum;

    int m = (l + r) / 2;
    push_down(k, l, m, r);

    return query(2 * k, l, m, u, v) + query(2 * k + 1, m + 1, r, u, v);
}

int main()
{
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; ++i)
        scanf("%d", &a[i]);

    initIT(1, 1, n);
    while (q--)
    {
        int t, a, b;
        scanf("%d%d%d", &t, &a, &b);
        if (t != 3)
        {
            int x;
            scanf("%d", &x);
            bool set_val = (t == 2);
            update(1, 1, n, a, b, x, set_val);
        }
        else
            printf("%lld\n", query(1, 1, n, a, b));
    }
    return 0;
}
```