Drzewa przedziałowe

- **złożoność obliczeniowa:** O(logn)

DRZEWO PUNKT - PRZEDZIAŁ

PROBLEM

dana jest liczba n i q oraz q zapytań. zapytania są wykonywane na n elementowym ciągu liczb, którego wszystkie elementy są na początku równe 0. możliwe zapytania:

- in **i x** → zapytanie dodające do i-tego elementu ciągu liczbę x
- query **i j** → zapytanie znajdujące sumę elementów z przedziału <i, j>

IMPLEMENTACJA

```
#include <bits/stdc++.h>
using namespace std;

const int M = 1 << 21; //przesunięcie bitowe o 21 zer
const int leaf = 1 << 20; //przesunięcie bitowe do pierwszego liścia - tab[0]
int tree[M + 7]{}; //drzewo przedziałowe

int query(int i, int j){
  i += leaf;
    j += leaf;
    int ans = tree[i];
    if(i != j) ans += tree[j];
    while(i/2 != j/2){
        if(i % 2 == 0) ans += tree[i + 1];
        if(j % 2 == 1) ans += tree[j - 1];
        i /= 2;
        j /= 2;
    }
    return ans;
}

void in(int a, int x){
    a += leaf;
    while(a){
        tree[a] += x;
        a /= 2;
    }
}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
```

```
    int n, q, i, j;
    cin >> n >> q;
    for(int i = 0; i < q; i++){
        char a;
        cin >> a;
        if(a == 'q') query(i, j);
        else in(i, j);
    }
}
```

synami wierzchołka tree[v] są tree[2 * v] i tree[2 * v + 1]

DRZEWO PRZEDZIAŁ - PUNKT

PROBLEM

dana jest liczba n i q oraz q zapytań. zapytania są wykonywane na n elementowym ciągu liczb, którego wszystkie elementy są na początku równe 0. możliwe zapytania:

- in **i j x** → zapytanie dodające do każdego elementu <i, j> wartość x
- query **i** → zapytanie znajdujące wartość i-tego elementu ciągu

IMPLEMENTACJA

```
#include <bits/stdc++.h>
using namespace std;

const int M = 1 << 21; //przesunięcie bitowe o 21 zer
const int leaf = 1 << 20; //przesunięcie bitowe do pierwszego liścia - tab[0]
int tree[M + 7]{}; //drzewo przedziałowe

void in(int i, int j, int x){
    i += leaf;
    j += leaf;
    tree[i] += x;
    if(i != j) tree[j] += x;
    while(i/2 != j/2){
        if(i % 2 == 0) tree[i + 1] += x;
        if(j % 2 == 1) tree[j - 1] += x;
        i /= 2;
        j /= 2;
    }
}

int query(int i){
    i += leaf;
    int w = 0;
    while(i){
        w += tab[a];
        a /= 2;
    }
```

```
    return w;
}

int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int n, q, i, j;
    cin >> n >> q;
    for(int i = 0; i < q; i++){
        char a;
        cin >> a;
        if(a == 'q') query(i);
        else in(i, j, x);
    }
}
```

synami wierzchołka tree[v] są tree[2 * v] i tree[2 * v + 1]

DRZEWO PRZEDZIAŁ - PRZEDZIAŁ

PROBLEM

dana jest liczba n i q oraz q zapytań. zapytania są wykonywane na n elementowym ciągu liczb, którego wszystkie elementy są na początku równe 0. możliwe zapytania:

- in **i j x** → zapytanie dodające do elementów <i, j> wartość x
- query **i j** → zapytanie znajdujące sumę elementów z przedziału <i, j>

IMPLEMENTACJA

```
#include<bits/stdc++.h>

using namespace std;

const int M = 1 << 20;
long long suma[M<<1];   //tablica przechowująca sumę wartości wszystkich liści poddrzewa,
            //którego korzeniem jest komórka tablicy o danym ideksie
long long obciazenie[M<<1]; //tu przechowujemy obciążenia

void insert(int a, int b, int wartosc){
    int l = a + M, r = b + M;
    int dlugosc = 1;
    obciazenie[l] += wartosc;
    suma[l] += wartosc;
    //jeśli przedział nie jest jednoelementowy
    if(l!=r){
        obciazenie[r] += wartosc;
        suma[r] += wartosc;
    }
    while(l > 0){
```

```
        if(l < r -  1){ //jeśli wierzchołki nie mają wspólnego ojca
            if(l%2 == 0){
                suma[l+1] += wartosc*dlugosc;
                obciazenie[l+1] += wartosc;
            }
            if(r%2 == 1){
                suma[r - 1] += wartosc*dlugosc;
                obciazenie[r - 1] += wartosc;
            }
        }

        if(r < M){ //jeśli to nie są liście
          suma[l] = suma[l*2] + suma[l*2 + 1] + obciazenie[l]*dlugosc;
            suma[r] = suma[r*2] + suma[r*2 + 1] + obciazenie[r]*dlugosc;
        }

        r /= 2;
        l /= 2;
        dlugosc *= 2;
    }
}

long long query(int a, int b){
    int l = a + M, r = b + M,
    dl_r,   //długość przedziału od prawego krańca
    dl_l; //długość przedziału od lewego krańca
    int dlugosc = 1;
    dl_l = 1;
    if(l!=r) dl_r = 1;
    else dl_r = 0;

    long long wynik = 0;

    while(l > 0){ //dopóki nie dotrzemy do korzenia
        wynik += obciazenie[l]*dl_l + obciazenie[r]*dl_r;
        if(l < r -  1){ //jeśli wierzchołki nie mają wspólnego ojca
            if(l%2 == 0){ //lewy syn na lewej ścieżce
                wynik+= suma[l+1];
                dl_l+= dlugosc;
            }
            if(r%2 == 1){ //prawy syn na prawej ścieżce
                wynik+= suma[r-1];
                dl_r+= dlugosc;
            }
        }
        r /= 2;
        l /= 2;
        dlugosc<<=1;
    }
    return wynik;
}

int main(){
  ios_base::sync_with_stdio(0);
    int n, option, q, a, b, wartosc;

    cin >> q;
    while(q--){
```

```
        cin >> option; //1 insert, 0 query
        if(option){
            cin >> a >> b >> wartosc;
            insert(a, b, wartosc);
        }
        else{
            cin >> a >> b;
            cout << query(a, b) << endl;
        }
    }
}
```

synami wierzchołka tree[v] są tree[2 * v] i tree[2 * v + 1]

na początku wartości drzewa przedziałowego powinny zostać ustawione na wartość neutralną