Олимпиадное программирование

Занятие 14. ДО. Массовые операции

Труфанов Павел Николаевич

Онлайн-школа  Фоксфорд

Foxford.ru 2019-2020

Несколько задач с прошлого занятия

- Количество максимумов на отрезке
- Максимальная сумма двух чисел на отрезке
- Максимальный возрастающий подотрезок на отрезке.
- Максимальная сумма подотрезка на отрезке

Массовые операции

Теперь мы хотим делать запросы на отрезке, а также делать массовые изменения. Мы разделим это на две части: массовое прибавление на отрезке и массовое присвоение на отрезке. Идея в данной задаче – так называемые "отложенные операции"

Операция проталкивания

Давайте для начала решим задачу о массовом прибавлении. Давайте в каждой вершине хранить то, сколько на отрезке этой вершины нужно прибавить. Но если я внезапно в процессе некоторой операции решу пойти в детей некоторой вершины, в которой есть отложенная операция, то я не хочу помнить о том, что сверху нас еще ждет некоторое изменение. Поэтому я введу операцию "проталкивания".

Код

```
c - array with postponed changes

void push(int v) {
    c[2 * v + 1] += c[v];
    c[2 * v + 2] += c[v];
    c[v] = 0;
}
```

Операция проталкивания

Итого за $O(1)$ времени мы избавились от необходимости помнить о изменениях сверху. К тому же, такая операция не меняет никакие конечные значения в дереве. Таким образом, всегда, когда я иду в детей некоторой вершины v, я сделаю push(v).

Что я храню?

Мы уже поняли, что мы храним в массиве c. Но что мы храним в массиве t, массиве самого дерева отрезков? Давайте в t[v] хранить функцию от ее подотрезка, плюс учитывать все проталкивания в детях, ведь они тоже влияют на функцию. Но мы не будем в t[v] учитывать c[v]. Так как t[2 * v + 1] и t[2 * v + 2] учитывают все проталкивания в их детях, то в t[v] осталось учесть только c[2 * v + 1] и c[2 * v + 2]

Пересчет

Всегда, когда я делаю операцию push, структура дерева меняется. Поэтому после операций push я всегда буду запускать функцию update(v), которая обновляет значение в вершине. Но надо делать это только после того, как мы запустились от детей.

Пересчет

```
void update(int v,int l,int r,int m) {
    //summ
    t[v] = t[2*v+1]+t[2*v+2]+
           c[2*v+1]*(m-l)+
           c[2*v+2]+(r-m);
    //minimum
    t[v] = min(t[2*v+1]+c[2*v+1],
               t[2*v+2]+c[2*v+2]);
}
```

Функция массового изменения

Она будет выглядеть почти так же, как выглядела функция запроса на отрезке. Идея будет та же самая.

Код

```
void change(int v,int l,int r,
            int askl,int askr,int val)
    if (l >= askr || r <= askl)
        return;
    if (l >= askl && r <= askr)
        c[v] += val;
        return;
    push(v);
    int m = (l + r) / 2;
    change(2*v+1,l,m,askl,askr,val);
    change(2*v+2,m,r,askl,askr,val);
    update(v, l, r, m);
```

Функция запроса

Пусть ask для вершины v возвращает значения для отрезка v, который учитывает значение с. Не забудем сделать update, если мы сделали push.

Код

```
int ask(int v,int l,int r,
        int askl,int askr)
    if (l >= askr || r <= askl)
        return NEUTRAL;
    if (l >= askl && r <= askr)
        return t[v] + c[v] * (r - l);
    push(v);
    int m = (l + r) / 2;
    int x = ask(2*v+1,l,m,askl,askr);
    int y = ask(2*v+2,m,r,askl,askr);
    update(v);
    return x + y;
```

Массовое присвоение

Здесь понадобится переписать код функции push, update и формулу в ask. Для начала заметим, что раньше в функции push мы в c[v] присваивали 0, что значило, что в вершине v больше нет изменения. Для присвоений так не получится, поэтому заведем массив типа bool, который показывает было ли изменение в вершине v.

До встречи!

FOXFORD.RU

Онлайн-школа Фоксфорд

Фоксфорд