Дерево Фенвика
Introduction to Fenwick tree

Дерево Фе́нвика — структура данных, требующая O(n) памяти и позволяющая эффективно (за O(log n)) выполнять следующие операции:

Memory: O(n) , Time: O(log n)

Operations

- изменять значение любого элемента в массиве,
- To change the value of any element in the array
- выполнять некоторую ассоциативную, коммутативную, и обратимую операцию на отрезке.
- To calculate some associative and commutative functions on the interval

There is an array A[0..N-1].

Let’s construct an array T[0..N-1]

$$T_i = \sum_{k=F(i)}^{i} A_k$$

where $F(i)$ is some function, we will discuss it below.

Теперь мы можем реализовать следующие функции:

$$T_i = \sum_{k=F(i)}^{i} A_k$$

```
//Sum [0; R]
int sum(int r)
{
    int result = 0;
    while (r >= 0) {
        result += t[r];
        r = f(r) - 1;
    }
    return result;
}
```

```
//Update i on value delta
void update(int i, int delta)
{
    for all j, where F(j) <= i <= j
    {
        t[j] += delta;
    }
}
```

Что за функция F - ?

- Let $F(x) = x \,\&\, (x + 1)$, where «&» is bitwise AND.

Что за функция F - ?

- Let $F(x) = x \,\&\, (x+1)$, where «&» is bitwise AND.

How can we find for given i all the j that F(j) <= i <= j

Что за функция F - ?

- Let $F(x) = x \,\&\, (x+1)$, where «&» is bitwise AND.

How can we find for given i all the j that F(j) <= i <= j

Let's use $H(x) = x \mid (x+1)$, where «|» is bitwise OR.

```
int sum (int r)
{
    int result = 0;
    for (; r >= 0; r = (r & (r+1)) - 1)
        result += t[r];
    return result;
}
```

```
int sum (int l, int r)
{
    return sum (r) - sum (l-1);
}
```

```
void change (int i, int delta)
{
    for (; i < n; i = (i | (i+1)))
        t[i] += delta;
}
```

Initialization in O(N log N)

```
void init (vector<int> a)
{
    t.resize((int) a.size(), 0);
    for (int i = 0; i < (int)a.size(); i++)
        change(i, a[i]);
}
```

Initialization in O(N)

```
void init(vector<int> a)
{
    int k;
    for(int i=0; i < (int)a.size(); i++)
    {
        t[i]+=a[i];
        if((k = i | (i + 1)) < (int)a.size())
            L[k]+=t[i];
    }
}
```

Advantages

- It allows to calculate the value of some associative, commutative operations on the interval [L; R] and to change the value of any element in O (log N)
- Memory in O(N)
- The easy implementation
- It can be easy transformed into 2D and 3D arrays

2D Fenwick Tree

```
int sum (int x, int y)
{
    int result = 0;
    for (int i = x; i >= 0; i = (i & (i+1)) - 1)
        for (int j = y; j >= 0; j = (j & (j+1)) - 1)
            result += t[i][j];
    return result;
}
```

```
void change (int x, int y, int delta)
{
    for (int i = x; i < n; i = (i | (i+1)))
        for (int j = y; j < m; j = (j | (j+1)))
            t[i][j] += delta;
}
```

Disadvantages

- The using operation in Fenwick Tree must be reversible, so it can't work with "maximum" and "minimum" operations well on intervals.

…but!

- При некоторой модификации, мы всё же сможем работать с минимумом (с максимумом) на отрезке, но с ограничениями.
- Так же дерево можно модифицировать для изменения элементов на отрезке.
- We can modify it so it can work even with max and min (with some limitation).
- The tree can be modified to work with changing elements on the interval.

Полезные ссылки

- Общая информация по дереву
  a) E-maxx
  b) Конспекты ИТМО
  c) Хабрахабр
- Дерево и максимум
- Дерево с модификацией на отрезке