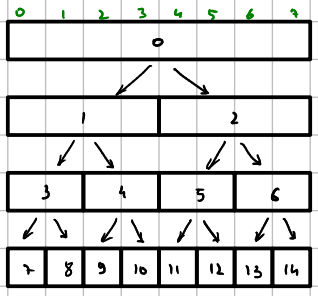
Дерево отрезков

1. update i x
2. get l r



T – стр. с информ. о верш.

v<T>t(4n) – знач. вершин
v<i>a(n) – массив, по кот. строим

вершина:
v – номер
[l;r) – границы
t[v] – значение верш.
v·2+1 – номер левого
v·2+2 – номер правого

Build

```
T build(int v, int l, int r) {
    if (r - l == 1)
        return t[v] = a[l];
    int m = (l + r) / 2;
    return t[v] = merge(build(v * 2 + 1, l, m),
                        build(v * 2 + 2, m, r));
}
```

1. Посчит. подд.
2. Присв. t[v]
3. Вернуть t[v]

Update

```
T update(int v, int l, int r, int i, int x) {
    if (i < l || i >= r)
        return t[v];
    if (r - l == 1)
        return t[v] = x;
    int m = (l + r) / 2;
    return t[v] = merge(update(v * 2 + 1, l, m, i, x),
                        update(v * 2 + 2, m, r, i, x));
}
```

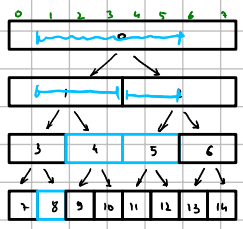
(i – индекс, x – знач.)

1. Обн. значений
2. Обн. t[v]
3. Вернуть t[v] после изм.

get

v [l;m) [ql;qm) – возвр. T пересечения [l;r) и [ql;qr)

(l;m) – отр. верш., [ql;qm) – отр. запр.



Случаи

1. ql ≤ l && r ≤ qr (пересек.) [l;m) ∩ [ql;qm) = [l;m)
   ret. t[v]
2. qr ≤ l (o r ≤ o ql) [l;m) ∩ [ql;qm) = ∅
   ret T(∅)
3. ret merge(get(лев. сын), get(прав. сын))

```
T get(int v, int l, int r, int ql, int qr) {
    if (ql <= l && r <= qr)
        return t[v];
    if (qr <=  l || r <= ql)
        return T({});
    int m = (l + r) / 2;
    return merge(get(v * 2 + 1, l, m, ql, qr),
                 get(v * 2 + 2, m, r, ql, qr));
}
```

```
struct segtree {
    vector<int> t, a;

    int build(int v, int l, int r) {
        if (r - l == 1)
            return t[v] = a[l];
        int m = (l + r) / 2;
        return t[v] = build(v * 2 + 1, l, m)
                    + build(v * 2 + 2, m, r);
    }

    segtree(vector<int> &vec) {
        a = vec;
        t.resize(a.size() * 4);
        build(0, 0, a.size());
    }

    int update(int v, int l, int r, int i, int x) {
        if (i < l || i >= r)
            return t[v];
        if (r - l == 1)
            return t[v] = x;
        int m = (l + r) / 2;
        return t[v] = update(v * 2 + 1, l, m, i, x)
                    + update(v * 2 + 2, m, r, i, x);
    }

    int get(int v, int l, int r, int ql, int qr) {
        if (ql <= l && r <= qr)
            return t[v];
        if (qr <= l || r <= ql)
            return 0;
        int m  = (l + r) / 2;
        return get(v * 2 + 1, l, m, ql, qr)
             + get(v * 2 + 2, m, r, ql, qr);
    }
};
```

k-й ноль

t[v] – кол-во 0 в v

get_cnt(ql, qr) – кол-во 0 на [ql, qr)

k-й 0 на l r ⟷ k + get_cnt(0, l) - й 0 на всем массиве

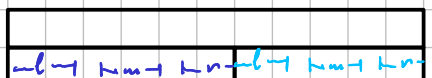
1. На макс. ≥ k + get_cnt(0, l) 0
2. k + get_cnt(0, l) - й 0 ∈ [ql, qr]

int get_k(v, k): – номер 0 на верш. v, инд. k-го 0

```
if (лист)
    ret l
if (t[v·2+1] >= k)
    ret get(лев.с, k)
else
    ret get(прав.с, k - t[v·2+1]);
```

Макс. кол-во подряд идущих

```
struct T {
    int l, m, r;   // макс. кол-во подряд; на суфф.; на преф.
    bool f;        // вся верш. – нули
};
```



```
merge:
if f && f:
    ret {l+m, m+m, m+m, true};
if f && !f:
    ret {l+l, max(m+l, m), r, false};
if !f && !f:
    ret {l, max(m, m, r+l), r, false};
```