{% note info %} **摘要** Title: 线段树——加法、最大值 Tag: 线段树 Memory Limit: 64 MB Time Limit: 1000 ms {% endnote %}

Powered by:**NEFU AB-IN**

@TOC

线段树

- 加法

```python
'''
Author: NEFU AB-IN
Date: 2022-02-08 16:02:19
FilePath: \ACM\Acwing\segment-tree.py
LastEditTime: 2022-02-09 16:04:48
'''

ls = lambda p: p << 1
rs = lambda p: p << 1 | 1


class Node(object):
    def __init__(self, l, r) -> None:
        self.l = l
        self.r = r
        self.len = 0
        self.tag = 0


N = int(1010)
tr = [Node(0, 0) for _ in range(N << 3)]


def pushup(p):
    tr[p].len = tr[ls(p)].len + tr[rs(p)].len


def build(p, l, r):
    tr[p] = Node(l, r)
    if l == r:
        return
    mid = l + r >> 1
    build(ls(p), l, mid)
    build(rs(p), mid + 1, r)
    pushup(p)


def addtag(p, d):
    tr[p].tag += d
    tr[p].len += d * (tr[p].r - tr[p].l + 1)


def pushdown(p):
    if tr[p].tag:
        addtag(ls(p), tr[p].tag)
        addtag(rs(p), tr[p].tag)
        tr[p].tag = 0


#l, r 是固定的，二分的永远是tr[p].l和tr[p].r
def update(p, l, r, d):
    if l <= tr[p].l and tr[p].r <= r:
        addtag(p, d)
        return
    pushdown(p)
    mid = tr[p].l + tr[p].r >> 1
    if l <= mid:
        update(ls(p), l, r, d)
    if mid < r:
        update(rs(p), l, r, d)
    pushup(p)


def query(p, l, r):
    res = 0
    if l <= tr[p].l and tr[p].r <= r:
        return tr[p].len
    pushdown(p)
    mid = tr[p].l + tr[p].r >> 1
    if l <= mid:
        res += query(ls(p), l, r)
    if r > mid:
        res += query(rs(p), l, r)
    return res
```

- 最大值

```
'''
Author: NEFU AB-IN
Date: 2022-02-09 15:23:50
FilePath: \ACM\LanQiao\max.py
LastEditTime: 2022-02-09 16:10:36
'''
ls = lambda x: x << 1
rs = lambda x: x << 1 | 1

INF = int(2e9)
```

```python
class Node(object):
    def __init__(self, l, r):
        self.l = l
        self.r = r
        self.x = -INF


N = int(2e5 + 10)
tr = [Node(0, 0) for _ in range(N << 2)]


def pushup(p):
    tr[p].x = max(tr[ls(p)].x, tr[rs(p)].x)


def build(p, l, r):
    tr[p] = Node(l, r)
    if l == r:
        return
    mid = l + r >> 1
    build(ls(p), l, mid)
    build(rs(p), mid + 1, r)


def update(p, k, v): #单点修改
    if tr[p].l == k and tr[p].r == k:
        tr[p].x = v
        return
    mid = tr[p].l + tr[p].r >> 1
    if k <= mid:
        update(ls(p), k, v)
    if k > mid:
        update(rs(p), k, v)
    pushup(p)


def query(p, l, r):
    res = -INF
    if l <= tr[p].l and tr[p].r <= r:
        return tr[p].x
    mid = tr[p].l + tr[p].r >> 1
    if l <= mid:
        res = max(res, query(ls(p), l, r))
    if r > mid:
        res = max(res, query(rs(p), l, r))
    return res
```