Segment Tree (range sum/max/min with update)

```
太复杂，一般不考，range sum 能用 Binary Indexed Tree 代替，range max/min 没办法
```

```
""" Segment Tree (low, high, minimum, delta) """
lchild = lambda p: 2*p + 1
rchild = lambda p: 2*p + 2
parent = lambda p: (p-1) // 2

class SegmentTree(object):
    def __init__(self, array):
        n = len(array)
        self.array = array
        self.low = [0] * (4 * n)
        self.high = [0] * (4 * n)
        self.minimum = [0] * (4 * n)
        self.delta   = [0] * (4 * n)
        self.buildTree(p=0, i=0, j=n-1)

    def buildTree(self, p, i, j):
        low, high, minimum, array = self.low, self.high, self.minimum, self.array
        low[p], high[p] = i, j
        if i == j:
            minimum[p] = array[i]
        else: # i < j
            mid = i + (j-i)//2
            self.buildTree(lchild(p), i, mid)
            self.buildTree(rchild(p), mid+1, j)
            minimum[p] = min(minimum[lchild(p)], minimum[rchild(p)])

    def prop(self, p):
        delta = self.delta
        delta[lchild(p)] += delta[p]
        delta[rchild(p)] += delta[p]
        delta[p] = 0

    def update(self, p):
        minimum, delta = self.minimum, self.delta
        lc, rc = lchild(p), rchild(p)
        minimum[p] = min(minimum[lc] + delta[lc], minimum[rc] + delta[rc])
    def range_increment(self, i, j, val, p=0):
        lo, hi = self.low[p], self.high[p]
        if j < lo or hi < i:          # no overlap
            return
        elif i <= lo and hi <= j:     # total overlap
            self.delta[p] += val
        else:                         # partial overlap
            self.prop(p)
            self.range_increment(i, j, val, lchild(p))
            self.range_increment(i, j, val, rchild(p))
            self.update(p)

    def range_minimum(self, i, j, p=0):
        lo, hi = self.low[p], self.high[p]
        if j < lo or hi < i:          # no overlap
            return math.inf
        elif i <= lo and hi <= j:     # total overlap
            return self.minimum[p] + self.delta[p]
        else:                         # partial overlap
            self.prop(p)
            ret = min(self.range_minimum(i, j, lchild(p)), self.range_minimum(i, j, rchild(p)))
            self.update(p)
            return ret
```

Range Module

My Calendar III

Rectangle Area II

Falling Squares