基本解法

Java代码

```
class Solution {
    class SegTree { // 定义线段树，采用数组存储
        int[] value, nums;
        int n;// 表示nums数组的长度
        SegTree(int n, int[] nums) {
            value = new int[4 * n];
            this.n = n;
            this.nums = nums;
            build(1, 0, n - 1);//注意bulid pos的初值是1 后序的坐标也要跟着变
        }

        public void build(int pos, int left, int right) {// pos表示
nums[left,right]最小值坐标在value中存储的位置
            if (right == left) {value[pos] = left;return;}// 注意存储的是坐标
            build(2 * pos, left, (left + right) / 2);// 建立左子树
            build(2 * pos + 1, (right + left) / 2 + 1, right);// 建立右子树
            value[pos] = nums[value[2 * pos]] > nums[value[2 * pos + 1]] ?
value[2 * pos + 1] : value[2 * pos];
        }

        // pos 为nums[left,right]最小值的坐标在value中存储的位置
        int query(int pos, int qleft, int qright, int left, int right) {// 递归
的找到给定区间的最小值
            if (left > right) return -1;
            if (qright == right && qleft == left) return value[pos];// 递归出口
            int mid = (qright + qleft) / 2;
            int leftIndex = query(2 * pos, qleft, mid, left, Math.min(mid,
right));//左侧区间最小值的坐标
            int rightIndex = query(2 * pos + 1, mid + 1, qright, Math.max(left,
mid + 1), right);
            if (leftIndex == -1) return rightIndex;
            if (rightIndex == -1) return leftIndex;
            return nums[leftIndex] > nums[rightIndex] ? rightIndex : leftIndex;
        }

        int rec(int od, int l, int r) {// 分别对左右子区间递归累加直至区间最小值 本题
定制函数
            if (l > r) return 0;
            int m = query(1, 0, n - 1, l, r);
            return nums[m] - od + rec(nums[m], l, m - 1) + rec(nums[m], m + 1,
r);
```

```
        }
    }

    public int minNumberOperations(int[] t) {
        int n = t.length;
        SegTree st = new SegTree(n, t);
        return st.rec(0, 0, n - 1);
    }
}
```

##

最优解法

Java代码

```
 public int minNumberOperations(int[] A) {
        int res = 0, pre = 0;
        for (int a: A) {
            res += Math.max(a - pre, 0);
            pre = a;
        }
        return res;
    }
```

C++代码

```
class Solution {
public:
    int minNumberOperations(vector<int>& target) {
        const int n = target.size();
        int ans = target[0];

        for (int i = 0; i < n - 1; i++)
            ans += max(target[i + 1] - target[i], 0);

        return ans;
    }
};
```

Python代码

```python
def minNumberOperations(self, A):
        res = pre = 0
        for a in A:
            res += max(a - pre, 0)
            pre = a
        return res
```