

# Segment Tree

hzwer,  $n + e$

PKU, THU

2017 年 12 月 28 日



# Introduction

- 给你一段长度为  $N$  的序列  $A[]$ , 求:
  - ① 单点修改, 单点查询;
  - ② 单点修改, 区间查询;
  - ③ 区间修改, 单点查询;
  - ④ 区间修改, 区间查询;
- $N \leq 100000$
- 修改操作包括但不限于  $+ C$ ,  $* C$ , 强制  $= C$ , .....
- 查询操作包括但不限于  $\max$ ,  $\min$ ,  $\text{sum}$ , .....

## ② Application

- 线段树将一个区间划分成一些单元区间, 每个单元区间对应线段树中的一个叶结点.
- 根节点对应区间为  $[1, N]$
- 对于线段树中的每一个非叶子节点  $[a, b]$ , 它的左儿子表示的区间为  $[a, \frac{a+b}{2}]$ , 右儿子表示的区间为  $[\frac{a+b}{2} + 1, b]$ . 因此线段树是平衡二叉树, 最后的叶子节点数目为  $N$ .
- 为提高运行效率,  $\frac{a+b}{2}$  常常写成  $a + b \gg 1$
- 堆式存贮: 父亲节点编号为  $i$ , 则左儿子编号为  $i * 2$ , 右儿子编号为  $i * 2 + 1$
- 为提高运行效率, 常常写成  $i \ll 1$  和  $i \ll 1 | 1$

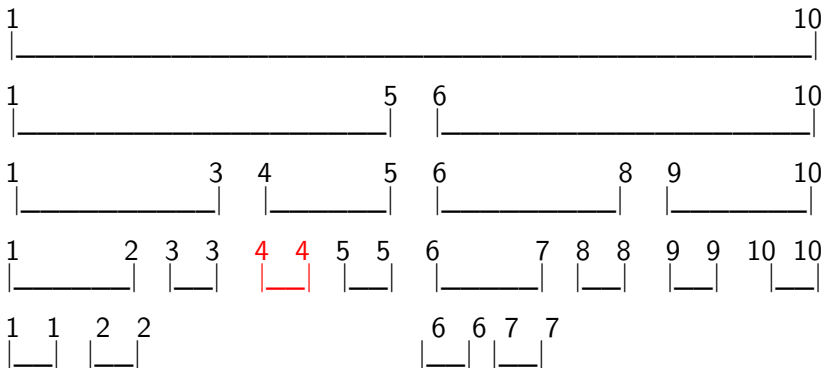
## 举个栗子

- 易知树高为  $\log N$

6 / 57

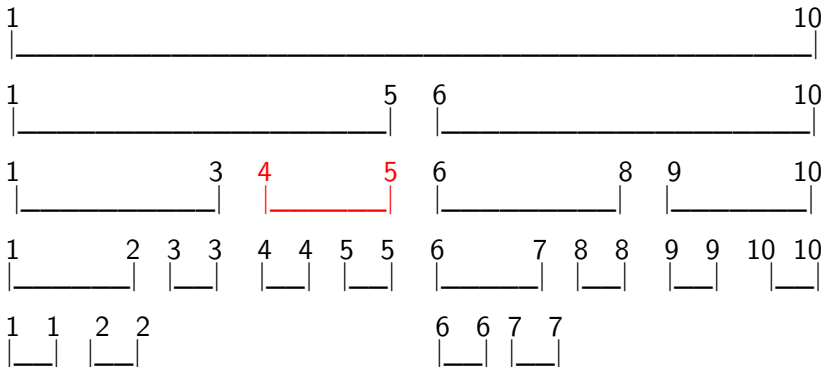
长啥样?  
修改  
查询  
Lazy tag  
代码

修改 A[4]



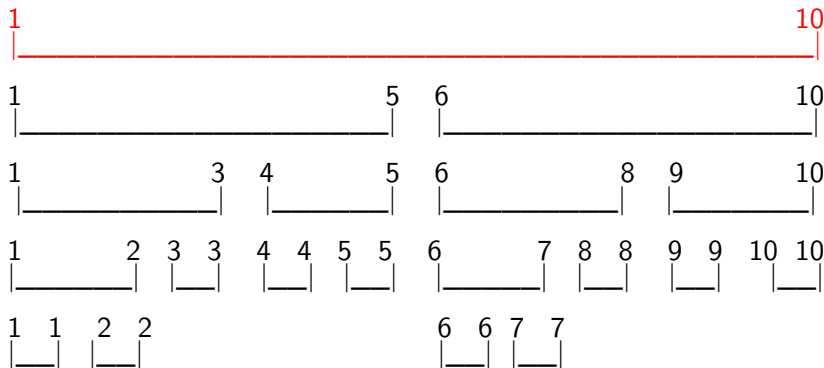


修改 A[4]





修改 A[4]



- 易知单点修改复杂度为  $\log N$
- 单点查询类似

```
void Update(int o, int l, int r) { // A[x] = y
    if (l == r) { sum[o] = y; return; }
    int mid = l + r >> 1;
    if (x <= mid) Update(o << 1, l, mid);
    else Update(o << 1 | 1, mid + 1, r);
    sum[o] = sum[o << 1] + sum[o << 1 | 1];
}
```

## ① Introduction

长啥样?

修改

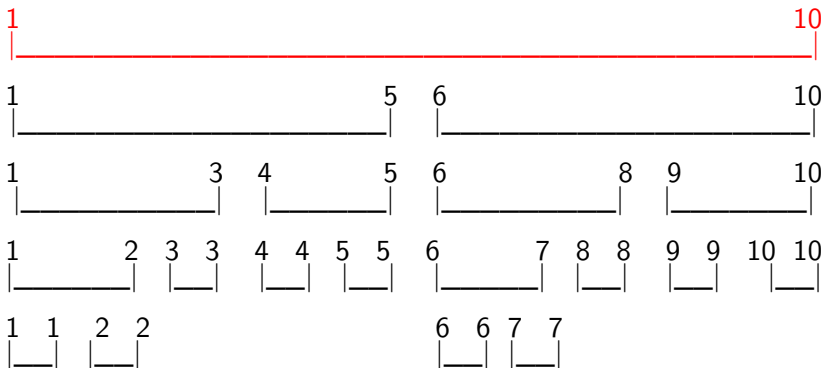
查询

Lazy tag

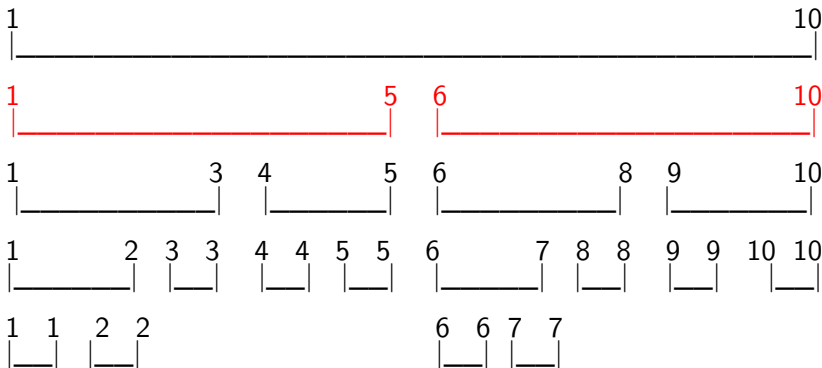
代码

## ② Application

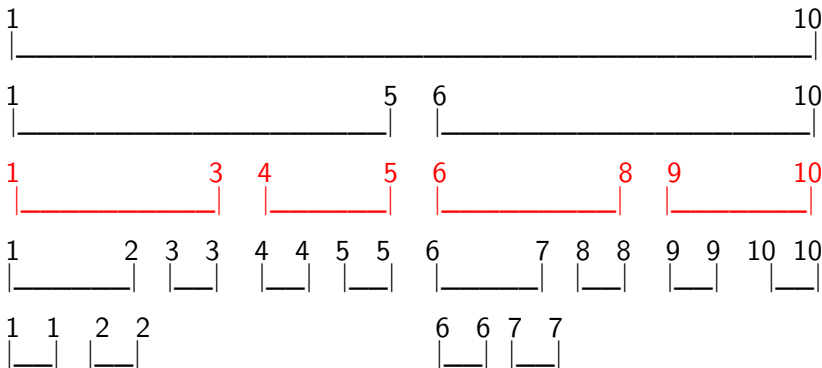
查询  $A[2..9]$



查询  $A[2..9]$

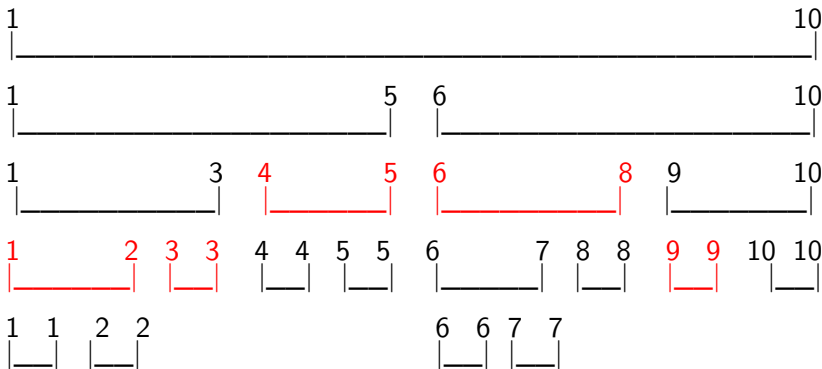


查询  $A[2..9]$

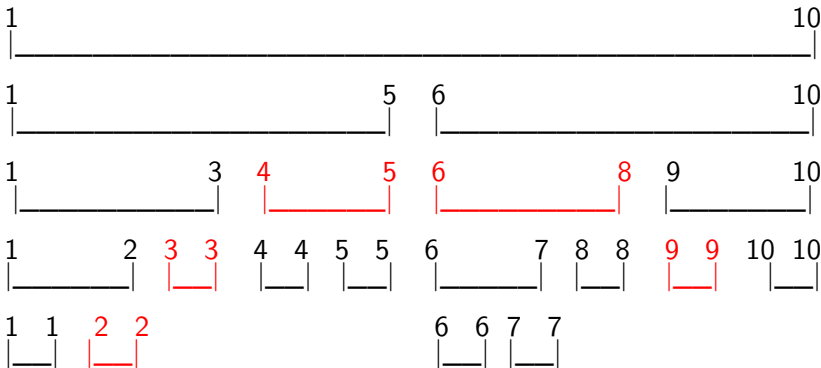




查询  $A[2..9]$



查询  $A[2..9]$



- Q: 为什么不直接查 [2,9]? A: 因为没有 [2,9] 这段区间啊 ...
- 易知能够通过访问不超过  $2 \cdot \log N$  个线段树上的区间来获得任意区间  $[l, r]$  的答案

```
void Query(int o, int l, int r) { // A[x..y]
    if (x <= l && r <= y) { ans += sum[o]; return; }
    int mid = l + r >> 1;
    if (x <= mid) Query(o << 1, l, mid);
    if (mid < y) Query(o << 1 | 1, mid + 1, r);
}
```

- 区间修改类似

- Q: 为什么不直接查 [2,9]? A: 因为没有 [2,9] 这段区间啊 ...
- 易知能够通过访问不超过  $2 \cdot \log N$  个线段树上的区间来获得任意区间  $[l, r]$  的答案

```
void Query(int o, int l, int r) { // A[x..y]
    if (x <= l && r <= y) { ans += sum[o]; return; }
    int mid = l + r >> 1;
    if (x <= mid) Query(o << 1, l, mid);
    if (mid < y) Query(o << 1 | 1, mid + 1, r);
}
```

- 区间修改类似吗?

## ① Introduction

长啥样?

修改

查询

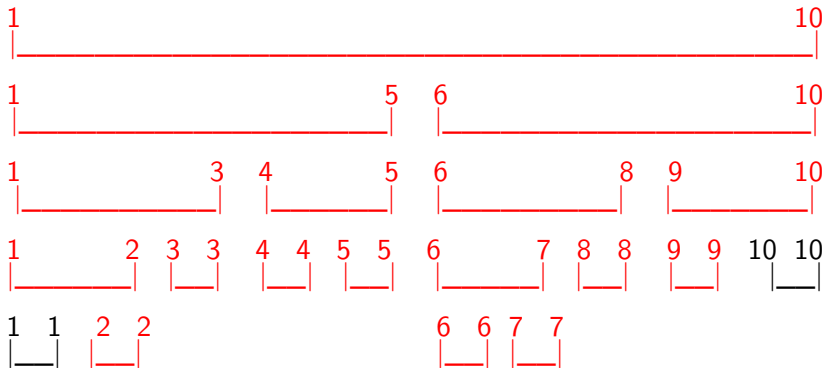
Lazy tag

代码

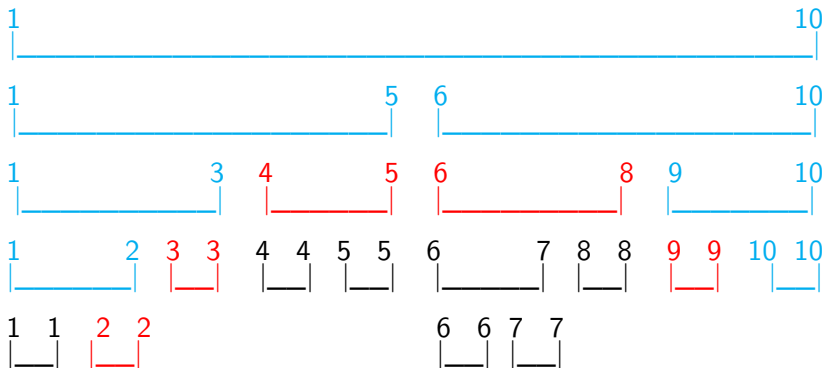
## ② Application

- Lazy-Tag 记录的是每一个线段树节点的变化值
- 当这部分区间的一致性被破坏时, 就将这个变化值传递给子区间
- 每个节点存一个 Tag 值, 表示这个区间进行的变化
- 每当访问到某一个节点时, Tag 下传

如果把  $A[2..9]$  每个数都  $+C$ , 那么真正要修改的节点大概这么多

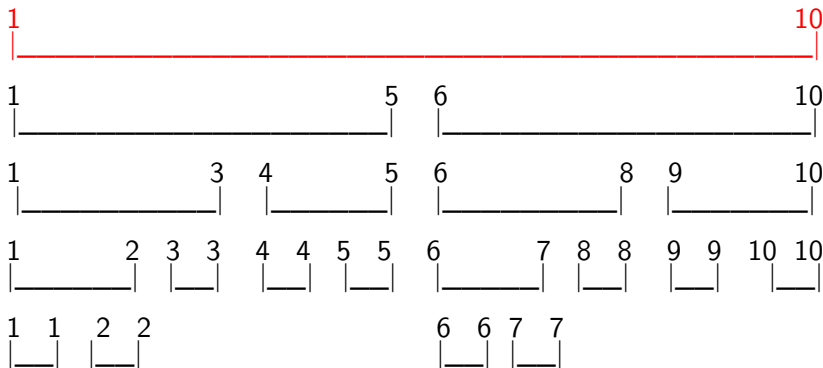


使用 Lazy Tag 以后, 情况是这样的:

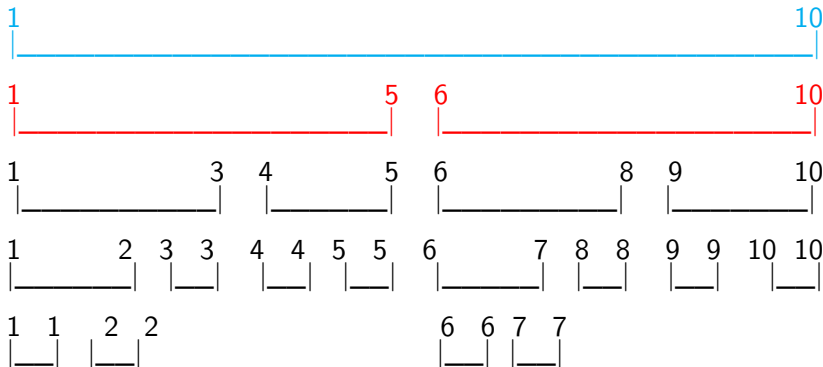




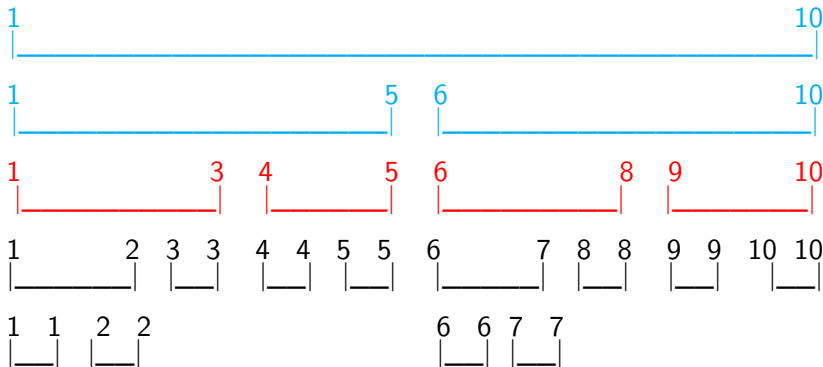
- 蓝色是要把信息及时维护的节点, 红色是本次区间修改操作 Lazy Tag 下传停止的位置.



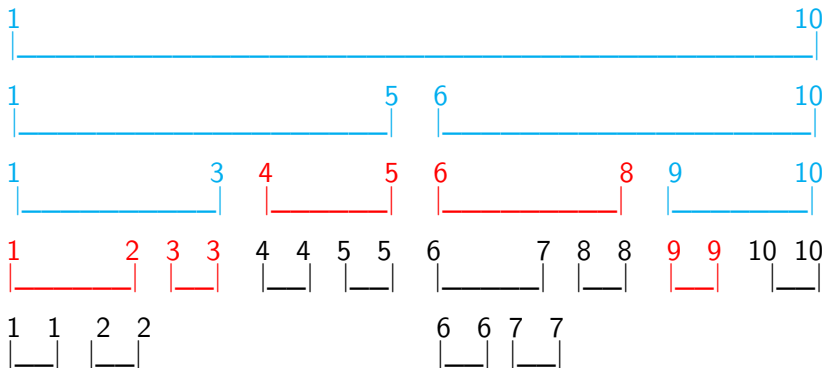
修改 A[2..9]



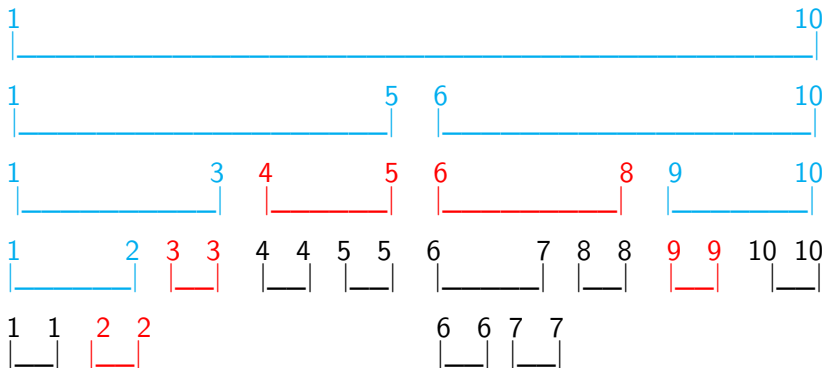
修改 A[2..9]



修改 A[2..9]

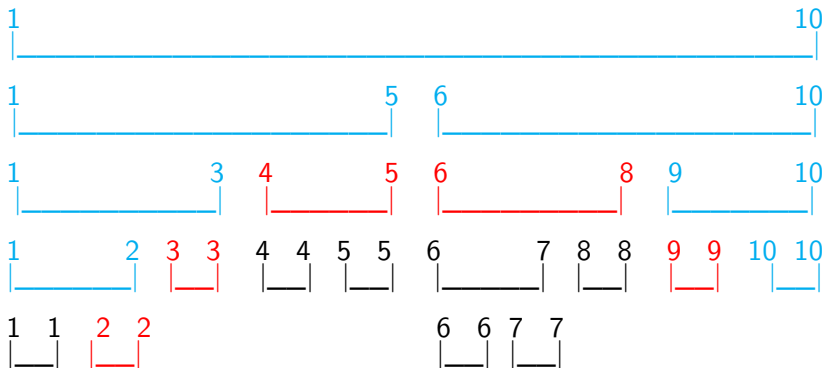


修改 A[2..9]



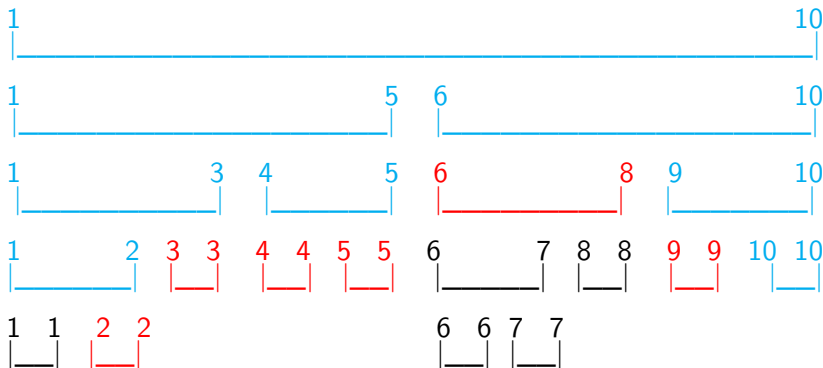
- 由于每一行最多只有两个蓝色区间和两个红色区间, 因此线段树区间修改的自带常数为 4.
- zkw 线段树: 满二叉树, 靠蓝色区间维护上去

要查询  $A[5]$

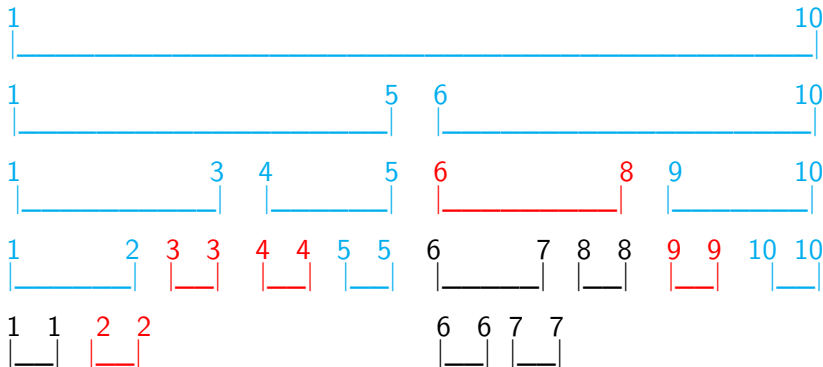




要查询  $A[5]$



要查询  $A[5]$



- 多个 Lazy Tag 咋办?
- 考虑打标记运算的优先级, 优先级高的先下传.

## ① Introduction

长啥样?

修改

查询

Lazy tag

代码

## ② Application

- **这里**
- 线段树里面每个节点都要记录统计量和 Lazy Tag (修改量)
- 建议写相关的函数都传 3 个参: (int o, int l, int r)
- 网络上说线段数要开 4 倍空间, 实际上如果没写挂的话, 正常只要这样开:  
先把 N 补成 2 的幂次, 再  $\times 2$  即可

\_\_\_\_\_

## ② Application

## 什么时候要用线段树?

## 什么时候要用线段树?

- 统计量可合并
- 修改量可合并
- 通过修改量可直接修改统计量

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26



## 例题 1

- 给定一个  $n$  个节点的堆，第  $i$  号结点的左儿子编号为  $2i$ ，右儿子为  $2i+1$

什么时候要用线段树?

## 例题 1

- 给定一个  $n$  个节点的堆，第  $i$  号结点的左儿子编号为  $2i$ ，右儿子为  $2i+1$
- 有  $m$  个操作，分为 2 类，每个结点初始权值为 0
- 操作 1：将  $u, v$  路径上的所有点权值加上  $w$
- 操作 2：询问第  $x$  号结点的权值

什么时候要用线段树?

## 例题 1

- 给定一个  $n$  个节点的堆，第  $i$  号结点的左儿子编号为  $2i$ ，右儿子为  $2i+1$
- 有  $m$  个操作，分为 2 类，每个结点初始权值为 0
- 操作 1：将  $u, v$  路径上的所有点权值加上  $w$
- 操作 2：询问第  $x$  号结点的权值
- $n \leq 10^9, m \leq 10^5, w \leq 10^9$

什么时候要用线段树?

## Solution

- 和线段树没啥关系 OwO
- 每次操作涉及的点只有  $\log$  个
- 暴力修改

什么时候要用线段树?

接下来的题目中如果  $N$  和  $M$  没说明范围, 默认  $10w$

什么时候要用线段树?

## 例题 2

- 对于一个序列维护以下操作:
  - ① 修改某个  $a[i]$  的值
  - ② 输入  $l, r$ , 选出区间  $[l, r]$  中的所有  $a[i]$ , 问他们两两之差的和是多少, 差的平方和是多少.
- 答案  $\text{mod } 10^9 + 7$  输出

什么时候要用线段树?

## Solution

- 第一问答案是 0 hhhh
- 第二问拆式子, 发现要维护区间和、区间平方和

什么时候要用线段树?

## 例题 3

- 给出  $A[]$ , 要求支持:
  - ① 询问  $A[l..r]$  中最大的数
  - ② 删除  $A[x]$ , 并且  $x+1$  以后的元素整体前移
  - ③ 在末尾增加一个数
- 1853



什么时候要用线段树?

## Solution

- 用 Splay 也是可以的我并没有意见
- 用线段树记录每个位置的数是否存在, 存在则标为 1, 不存在则标为 0
- 当然相应的 l,r 也要修改
- 转成线段树中第 k 小的数是哪个, 在线段树上二分
 

```
if (k <= sum[o]) Query(o << 1, l, mid);
else k -= sum[o], Query(o << 1 | 1, mid + 1, r);
```

什么时候要用线段树?

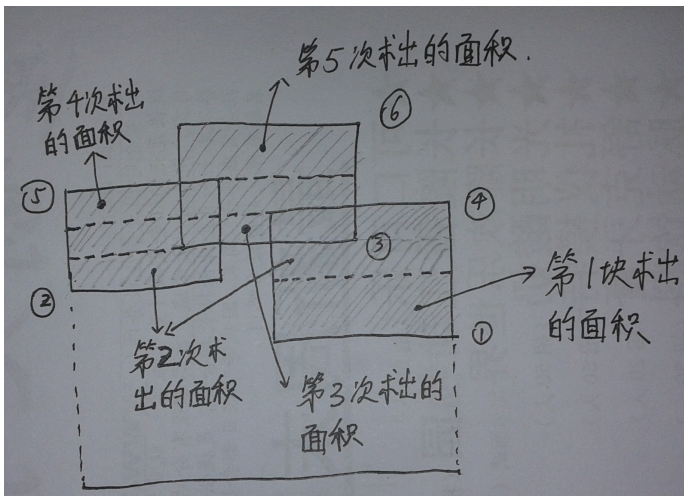
## 矩形面积并

- 二维平面上给出  $N$  个矩形, 求它们覆盖的总面积
- 1820

什么时候要用线段树?

## Solution

- 扫描线 灰不似窝的图!!! 窝的字木有喇么丑!!!



什么时候要用线段树?

## 线段树求最大子段和

- 给出  $A[]$ , 要求支持:
  - ①  $A[x]=y$
  - ② 询问  $A[x..y]$  的最大子段和
- 1647

什么时候要用线段树?

## Solution

- 都是套路.
- 一段答案的存在方式只有三种: 完全被左区间包含 / 完全被右区间包含 / 跨过左右区间分界点
- 只要合并答案就好
- 维护 `sum` && 从左/右端点开始的最大子段和, 答案可顺便维护

什么时候要用线段树?

## 例题 6

- 对长度为  $n$  的数列进行  $m$  次操作, 操作为:
  - ①  $a[l..r]$  每一项都加一个常数  $C$ , 其中  $0 \leq C \leq 10^{11}$
  - ② 求  $F[a[l]] + F[a[l+1]] + \dots F[a[r]] \bmod 10000$  的余数
- 其中  $F[i]$  表示斐波那契数列. 即  $F[0] = F[1] = 1$ ,  
 $F[n+2] = F[n+1] + F[n]$ .

什么时候要用线段树?

## Solution

- 对于每个位置, 保存  $F[a[i]]$  和  $F[a[i]+1]$
- 矩阵乘法具有分配律, Lazy Tag 只要记录幂次即可
- 预处理出循环节, 就不用每次都来矩阵快速幂了

什么时候要用线段树?

## 例题 7

- 对长度为  $n$  的数列进行  $m$  次操作, 操作为:
  - ① 对  $i \in [l, r]$  执行:  $a[i] = a[i]^2$
  - ② 求  $\sum_{i=l}^r a[i] \bmod p$  的余数,  $p$  在程序开始运行时给出
- 2164



什么时候要用线段树?

## Solution

- 找循环节
- 进了循环节后, 打上在循环节整体移动的 Lazy Tag
- 如果还没进, **暴力**递归修改, 根据均摊复杂度的那套理论, 这一部分复杂度不会超过  $O(N\log N)$

什么时候要用线段树?

## 例题 8

- 有一个  $2*n$  的点阵, 平行于坐标轴的方向上相邻的点之间可以连边, 维护以下操作:
  - ① 在某相邻两点之间连边
  - ② 删除某条边
  - ③ 询问某两点是否连通
- BZOJ1018

什么时候要用线段树?

## Solution

- 这个……用 LCT 我是没有意见的, 要试试能不能过.
- 维护  $A[x..x+1][0..1]$  四个格子之间的连通性
- 查询  $[l,r]$  是否联通, 注意有可能先掉头后直行

什么时候要用线段树?

## 例题 9

- 求  $A[]$  的一个最长子序列  $B[]$ , 满足  $B_i - B_{i-1} \leq d$ , 输出长度即可

## Solution

- 说好的 DP:  $f[i] = \max\{f[j] | 1 \leq j < i, |a[j] - a[i]| \leq d\} + 1$
- 按照 A[] 排序, 线段树中存 f[], 维护区间最大值
- 思路: 按顺序枚举右端点, 查询答案/左端点