

二分法

与统计问题

江苏省淮阴中学 李

睿

简介

- 一定范围内计数问题特点：
 - 1 描述简单
 - 2 要求对数据动态维护，动态计算
- 解决手段：特殊的统计模式和结构

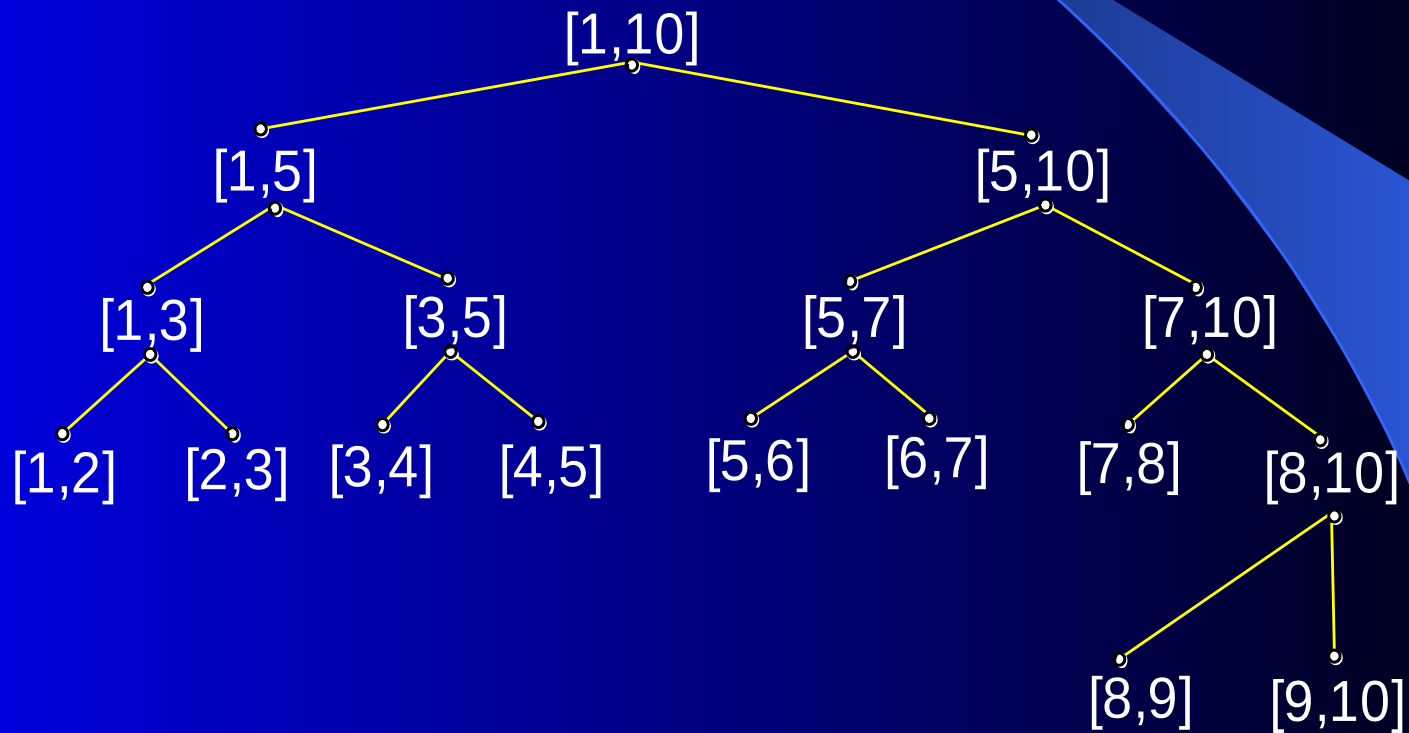
线段树

动态处理可以映射在一个坐标轴上的一些固定线段，例如求并区间的总长度，或者并区间的个数等等。

优点：随时插入一个区间或删除一个已有区间，并同时用低耗费维护需要的性质。

线段树 - 构造思想

- 下图显示了一个能够表示 $[1, 10]$ 的线段树:



线段树 - 动态数据结构

Type

Tnode=^Treenode;

Treenode=record

B,E:integer;

Count:integer;

LeftChild,Rightchild:Tnode;

End;

其中 B 和 E 表示了该区间为 $[B,E]$; Count 为一个计数器, 通常记录覆盖到此区间的线段的个数。LeftChild 和 RightChild 分别是左右子树的根。

线段树 - 静态数据结构

- 用数组

$B[]$, $E[]$, $C[]$, $LSON[]$, $RSON[]$ 。
设一棵线段树的根为 v 。那么 $B[v], E[v]$ 就是它所表示区间的界。 $C[v]$ 用来作计数器。 $LSON[v]$, $RSON[v]$ 分别表示了它的左儿子和右儿子的根编号。

线段树 - 建树

- 从根对应的区间 $[a, b]$ 出发，每次分成两个部分，分别建立对应的左右子树，直到面临一个初等区间 $[x, x+1]$ 。

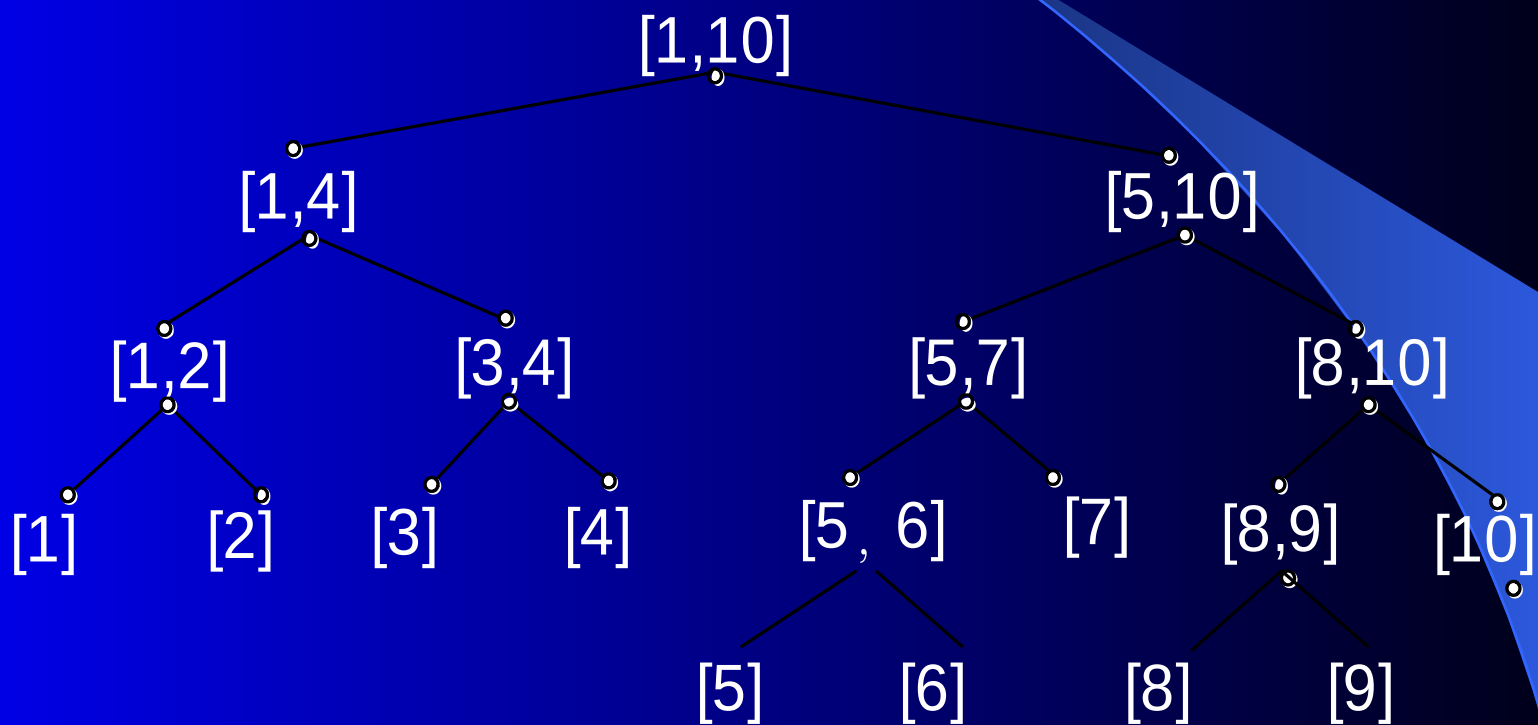
线段树 - 插入删除线段

- 将区间 $[c,d]$ 插入线段树 $T(a,b)$, 并设 $T(a,b)$ 的根编号为 v
- **procedure** INSERT($c,d;v$)
- **Begin**
- $\text{mid} = (\text{B}[v] + \text{E}[v]) \text{ div } 2;$
- **if** $c \leq \text{B}[v]$ and $\text{E}[v] \leq d$ **then** $\text{C}[v] \leftarrow \text{C}[v] + 1$
- **else if** $c < \text{mid}$ **then** INSERT($c,d;\text{LSON}[v]$);
- **if** $d > \text{mid}$ **then** INSERT($c,d;\text{RSON}[v]$);
- **end**

线段树 - 一个特殊性质举例

- 要得到线段树上线段并集的长度，增加一个数据域 $M[v]$ ，讨论：
- $C[v]>0, M[v] = E[v]-B[v]$;
- $C[v]=0$ 且 v 是叶子结点， $M[v]=0$;
- $C[v]=0$ 且 v 是内部结点， $M[v]=M[LSON[v]]+M[RSON[v]]$;

线段树 - 变形，对点统计



[例一]

- 一位顾客要进行 n ($1 \leq n \leq 5000$) 天的购物，每天他会有一些帐单。每天购物以后，他从以前的所有帐单中挑出两张帐单，分别是面额最大的和面额最小的一张，并把这两张帐单从记录中去掉。剩下的帐单留在以后继续统计。输入的数据保证，所有 n 天的帐单总数不超过 1000000，并且每份帐单的面额值是 1 到 1000000 之间的整数。保证每天总可以找到两张帐单。

- 建立点线段树，范围是 $[1, 1000000]$ ，即每种面额的帐单设为一个叶结点。
- 如果 $C[LSON[v]] > 0$ ，那么树 v 中的最小值一定在它的左子树上。
- 如果 $C[RSON[v]] > 0$ ，它的最大值在右子树上；

一种静态统计方法

- [例二] IOI2001 MOBILES :
- 在一个 $N \times N$ 的方格中，开始每个格子里的数都是 0。现在动态地提出一些问题和修改：提问的形式是求某一个特定的子矩阵 $(x1, y1) - (x2, y2)$ 中所有元素的和；修改的规则是指定某一个格子 (x, y) ，在 (x, y) 中的格子元素上加上或者减去一个特定的值 A 。现在要求你能对每个提问作出正确的回答。 $1 \leq N \leq 1024$, 提问和修改的总数可能达到 60000 条。

一维序列求和

- 设序列的元素存储在 $a[]$ 中， a 的下标是 $1..n$ 的正整数，需要动态地更新某个 $a[x]$ 的值，同时要求出 $a[x1]$ 到 $a[y1]$ 这一段所有元素的和。

- 对于序列 $a[1..n]$ ，我们设一个数组 $C[1..n]$ ， $C[i] = a[i - 2^k + 1] + \dots + a[i]$ ，其中 k 为 i 在二进制下末尾 0 的个数

1001010110010000 $k = 4$

- 计算 $C[x]$ 对应的 2^k LOWBIT (x)

$$2^k = x \text{ and } (x \text{ xor } (x-1))$$

- 修改一个 $a[x]$ 的值 与哪些 C 相关?

例如 $x=76=(1001010)_2$, 从形式上进行观察, 可以得到

:

$p1=1001010$

$p2=1001100$

$p3=1010000$

$p4=1100000$

$p5=10000000$

$$P(1) = x$$

$$P(i) < P(i+1)$$

$$P(i+1) = P(i) + LOWBIT(Pi)$$

- 修改 $C[p1], C[p2], \dots$

- **procedure** UPDATA(x, A)
- **begin**
- $p \leftarrow x$
- **while** ($p \leq n$) **do**
- **begin**
- $C[p] \leftarrow C[p] + A$
- $p \leftarrow p + \text{LOWBIT}(p)$
- **end**
- **end**

- 维护的费用: $\log n$

- 求 $a[1]-a[x]$ 的和
- **function** SUM(x)
- **begin**
- $\text{ans} \leftarrow 0$
- $p \leftarrow x$
- **while** ($p > 0$) **do**
- **begin**
- $\text{ans} \leftarrow \text{ans} + C[p]$
- $p \leftarrow p - \text{LOWBIT}(p)$
- **end**
- **return** ans
- **end**

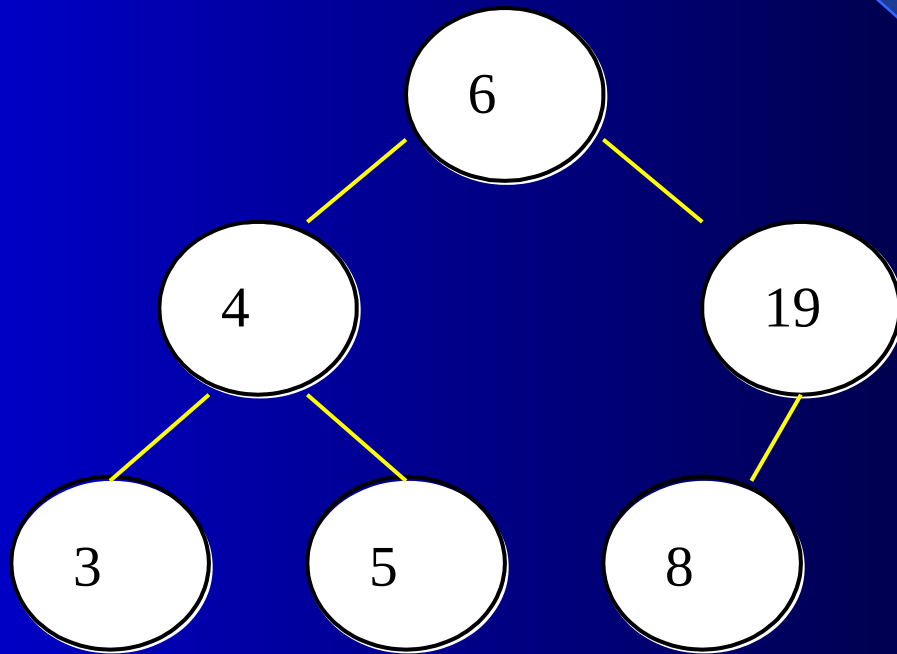
$$C[p] = a[p - 2^k + 1] + \dots + a[p]$$

下一个 $p = x - 2^k$

- 推广为原来的二维问题，把 C 构造成 $C[x, y]$ ，其每一维定义与原来相同。
- 推广后算法：两层嵌套，一次维护费用为 $O(\log^2 n)$

静态二叉排序树实现

- 集合 {3, 4, 5, 8, 19, 6}



构造过程 1 递归:

- 建立所有点坐标的映射 X
- $p \leftarrow 0$ 作为 X 映射中的指针
- **procedure** BUILD(ID:integer)
- **begin**
- if $(ID*2 \leq n)$ then BUILD($ID*2$);
- $p \leftarrow p+1$;
- $V[ID] = X[p]$;
- if $(ID*2+1 \leq n)$ then BUILD($ID*2+1$);
- **end**
- 在主程序中调用 BUILD(1)

构造过程 2 非递归:

- 方法, 依次找出当前点的后继点的下标
- 第一个点 **first** 一定为最下层最左边的一个位置, 若 n 个点有 L 层, 则 $\text{first}=2^{L-1}$
- 若当前的点位置为 **now** :

如果它有右儿子, 即 $\text{now} * 2 + 1 \leq n$, 则下一个位置是右子树最左下的点

如果没有右儿子, 当 **now** 是奇数时, 将 **now** 除以 2, 直到 **now** 是偶数, 最后再将 **now** 除以 2。

插入一个点 x

- **procedure** INSERT(x)
- **begin**
- $now \leftarrow 1$
- **repeat**
- $SUM[now] \leftarrow SUM[now] + 1$
- **if** ($V[now] = x$) **break**
- **if** ($V[now] > x$) $now \leftarrow now * 2$
- **else** $now \leftarrow now * 2 + 1$
- **until** false
- **End**
- 其中 **SUM** 是记录一棵子树上结点总数
- 删除 的方法是类似的

怎样解决例二

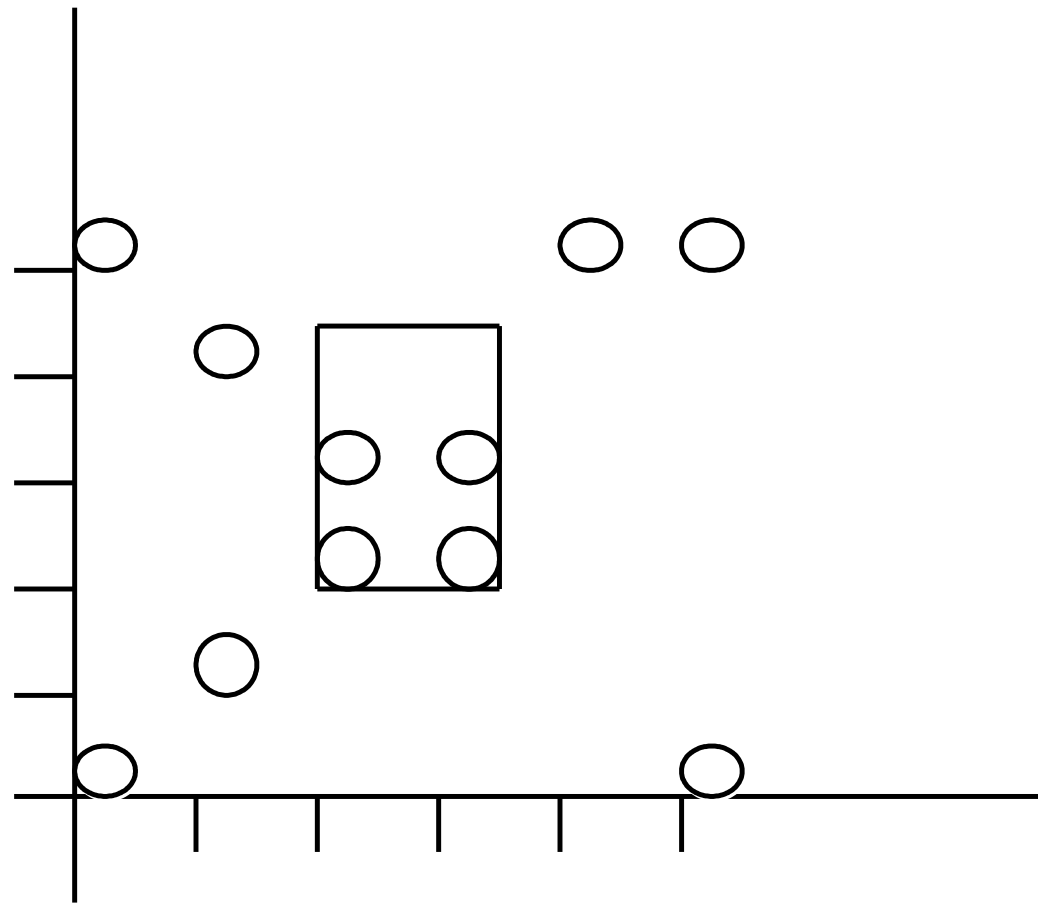
- **procedure** INSERT2(x,A)
- **begin**
- now \leftarrow 1
- repeat
- **if** (x \leq V[now]) **then**
- LESS[now] \leftarrow LESS[now]+A
- **if** (V[now]=x) **break**
- **if** (V[now]>x) now \leftarrow now*2
- **else** now \leftarrow now*2+1
- **until** false
- **end**
- LESS 为根及其左子树上所有点位置的权和

- 求 $a[1..x]$ 的元素和
- **function SUM(x):longint**
- **begin**
- **ans** \leftarrow 0
- **now** \leftarrow 1
- **repeat**
- **if (V[now]≤x) ans** \leftarrow **ans+LESS[now]**
- **if (V[now]=x) break**
- **if (V[now]>x) now** \leftarrow **now*2**
- **else now** \leftarrow **now*2+1**
- **until false**
- **return ans**
- **end**

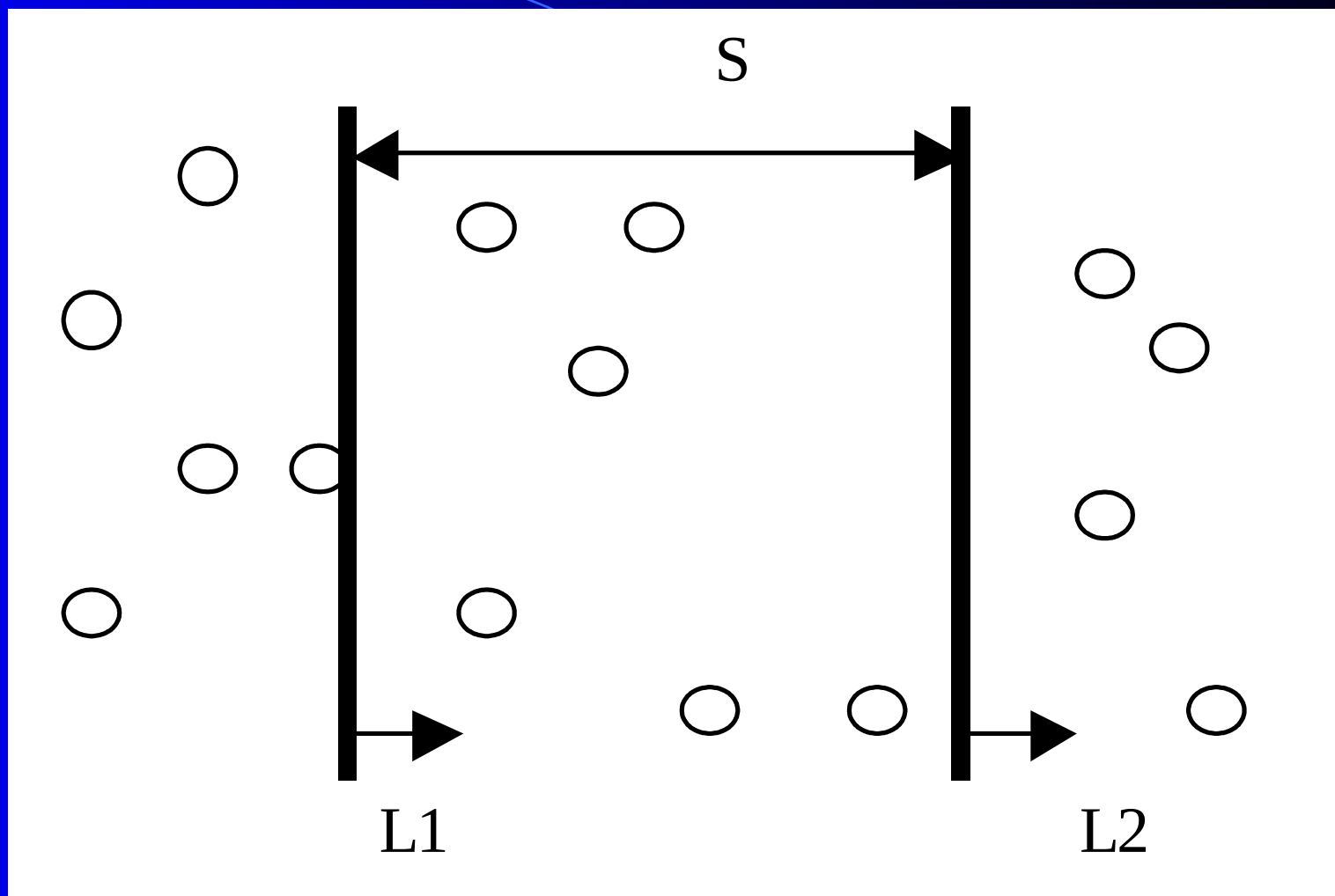
[例三] 采矿 (KOP)

- 金矿的老师傅年底要退休了。经理为了奖赏他的尽职尽责的工作，决定送他一块长方形地。长度为 S ，宽度为 W 。老师傅可以自己选择这块地。显然其中包含的采金点越多越好。你的任务就是计算最多能得到多少个采金点。如果一个采金点的位置在长方形的边上，它也应当被计算在内。
- 任务：
 - 读入采金点的位置。计算最大的价值。
- 输入：
 - 文件 KOP.IN 的第一行是 S 和 W ， $(1 \leq s, w \leq 10\ 000)$ ；他们分别平行于 OX 坐标和 OY 坐标，指明了地域的尺寸。接下来一行是整数 n $(1 \leq n \leq 15\ 000)$ ，表示采金点的总数。然后是 n 行，每行两个整数，给出了一个采金点的坐标。坐标范围是 $(-30\ 000 \leq x, y \leq 30\ 000)$ 。
- 输出：
 - 一个整数，最多的采金点数。

样例图示



- 初步，对 x 离散化后，图示



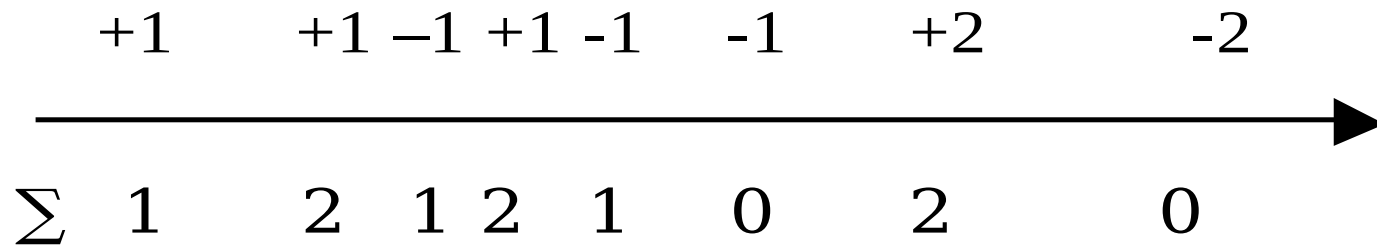
对于每一种坐标 y , 建立成两个点事件 $(y, +1), (y+w+1, -1)$,

例如在一个带状区域内有 5 个点的纵坐标分别是 $\{5, 3, 9, 1, 9\}$, $w=2$, 标成 $(1, +1), (4, -1), (3, +1), (6, -1), (5, +1), (8, -1), (9, +1), (12, -1), (9, +1), (12, -1)$,

再将他们按照 y 的坐标排序, 得 $(1, +1), (3, +1), (4, -1), (5, +1), (6, -1), (8, -1), (9, +1), (9, +1), (12, -1), (12, -1)$

我们把后面的标号反映在一个 y 坐标的映射上, 然后从低到高求和

- 坐标下的求和，这些和中最大的一个就是该带状区域中一个包含最多点数的矩形
- 在插入或者删除一个点事件之后，能够维持坐标下 Σ 的值；能够在很短时间内得到 Σ 中最大的一个值。



	+1	+1	-1	+1	-1	-1	+2	-2
Σ	1	2	1	2	1	0	2	0

实现:

- SUM[now] 对应子树上所有 +1 , -1 标号的和。实现极简单
- MAXSUM[now] , 子树上和最大的一个前缀的值。MAXSUM[1] 是一种状态下得到最优解。如何维护?
- MAXSUM[] 有哪几种可能?
- 1 最大值在左树上;
- 2 最大值正好包含根结点;
- 3 最大值在右树上。

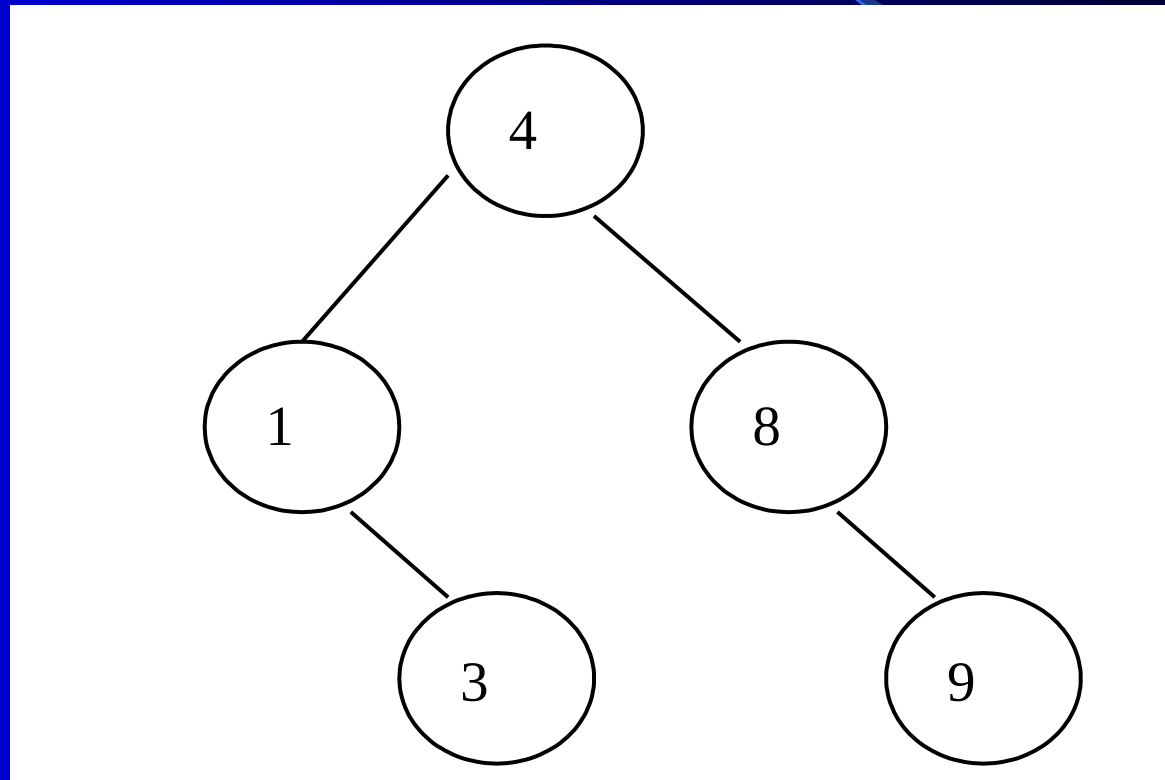
自下而上维护树的特性

- 找到当前坐标对应点序号 **now**, 修正标号为 **k**
- **Repeat**
- $SUM[now] \leftarrow SUM[now] + k$
 $MAXSUM[now] \leftarrow \text{Max}\{$
 - $MAXSUM[now*2],$
 - $SUM[now] - SUM[now*2+1],$
 - $SUM[now] - SUM[now*2+1] + MAXSUM[now*2+1]$
 - $\}$
- $now \leftarrow now \text{ div } 2$
- **until** $now = 0$

二分法虚拟实现树

- 二叉树使用之前必须构造出一个空的二叉树
- 对于任何一个有序表，在对其进行二分查找时，实际上就等于在一个二叉树上进行查找

- 对于一个表 {1, 3, 4, 8, 9} 的二分查找，等价于在如下图的二叉排序树上进行查找：



- 举插入结点的例子，来说明这种虚实现的方法，设 LESS 表示根及其左树上结点的个数：

procedure INSERT(x)

begin

$l \leftarrow 1, r \leftarrow n$

while ($l \leq r$) **do**

begin

$m = (l + r) \text{ div } 2$

if $x \leq V[m]$ $LESS[m] \leftarrow LESS[m] + 1$

if $x = V[m]$ **break**

if $x < V[m]$ **then** $r \leftarrow m - 1$

else $l \leftarrow m + 1$

end

end

[例四] 最长下降序列

- 给定一个整数序列 a ，求最长下降序列的长度。

$$a[0] = \infty$$

$$M[0] = 0$$

$$M[i] = \text{MAX} \{M[j] + 1, 0 \leq j < i \text{ 并且 } a[j] > a[i]\}$$

$$P[i] = j (j \text{ 是上式中取 MAX 时的 } j \text{ 值})$$

$$\text{ans} = \text{MAX} \{M[i]\}$$

$$O(n^2)$$

- 对 P 进行特殊的限制，即，在所有等价的决策 j 中， P 选择 $a[j]$ 最大的那一个
- 在处理完 $a[1..x-1]$ 之后，对于所有长度为 $M[x]-1$ 的下降序列， $P[x]$ 的决策只跟其中末尾最大的一个有关。
- 用另外一个动态变化的数组 b ，当我们计算完了 $a[x]$ 之后， $a[1..x]$ 中得到的所有下降序列按照长度分为 L 类，每一类中只需要一个作为代表，这个代表在这个等价类中末尾的数最大，我们把它记为 $b[j], 1 \leq j \leq L$ 。
- 处理当前的一个数 $a[x]$ ，我们无需和前面的 $a[j]$ ($1 \leq j \leq x-1$) 作比较，只需要和 $b[j]$ ($1 \leq j \leq L$) 进行比较。

- 首先，如果 $a[x] < b[L]$ ，把 $a[x]$ 接在这个序列的后面，形成了一个长度为 $L+1$ 的序列。这时 $b[L+1]=a[x]$ ，即 $a[x]$ 作为长度为 $L+1$ 的序列的代表，同时 L 应该增加 1。
- 另一种可能是 $a[x] \geq b[1]$ ，这时 $a[x]$ 仅能构成长度为 1 的下降序列，同时它又必然是最大的，所以它作为 $b[1]$ 的代表， $b[1]=a[x]$ 。
- 如果前面的情况都不存在，我们肯定可以找到一个 $j, 2 \leq j \leq L$ ，有 $b[j-1] > a[x], b[j] \leq a[x]$ ，这时分析， $a[x]$ 必然接在 $b[j-1]$ 后面，行成一个新的长度为 j 的序列。

- 这几种情况实际上都可以归结为：处理 $a[x]$ ，令 $b[L+1]$ 为无穷小，从左到右找到第一个位置 j ，使 $b[j] \leq a[x]$ ，然后则只要将 $b[j] = a[x]$ ，如果 $j = L+1$ ，则 L 同时增加。 x 处以前对应的最长下降序列长度为 $M[x] = j$ 。
- $L \leftarrow 0$
- **for** $x \leftarrow 1$ to n do
- **begin**
- $b[L+1] \leftarrow$ 无穷小
- $j \leftarrow 1$
- *while* $(b[j] > a[x])$ $j \leftarrow j+1$
- $b[j] \leftarrow a[x]$
- if $j > L$ then $L \leftarrow j$
- **end**

- 更改成:
- $j \leftarrow 1, k \leftarrow L$
- while ($j \leq k$) do
- begin
- $m \leftarrow (j+k) \text{ div } 2$
- if $b[m] > a[i]$ $j \leftarrow m+1$
- else $k \leftarrow m-1$
- end
- if $j > L$ $L \leftarrow L+1$
- $b[j] \leftarrow a[i]$

小结

- 离散化
- 转化问题，构造可统计的结构

Thank you all!

The image features the text "Thank you all!" in a 3D, orange, sans-serif font. The text is slanted upwards from left to right. A bright blue spotlight beam originates from the right side of the text and extends towards the top left corner of the frame. The background is a solid dark blue, with a lighter blue curved shape at the bottom right corner.