优先级队列

概述——需求与动机

仍然需要给关键字之间**定义**一个完整的全序关系，（也就是重载小于号）

然而，并不需要**维护**一个那么完善的全序关系。

实现：Vector + PQ

堆

完全二叉堆

下文都以最

结构

一个数组 / 向量，把完全二叉树的层次遍历拍扁。

`lc(x) = x << 1, rc(x) = x << 1 | 1, f(x) = x >> 1`。

“树”高：$O(\log n)$

操作

插入

逐层上滤。如果祖先比之小，就进行交换。$O(\log n)$

删除

把最后一个元素和堆顶交换，将堆大小 --，随后下滤堆顶。$O(\log n)$

建堆—— Floyd 建堆

Robert Floyd, 1964. 核心思想：由上而下的下滤；二叉树到底端的距离和是 $O(n)$ 的，而到顶端的距离和是 $O(n\log n)$ 的。

实现：如果有 $\mathcal{H}_0 \cup \{p\} \cup \mathcal{H}_1$ ，那么只需要把 $\mathcal{H}_0$ 和 $\mathcal{H}_1$ 都当作 p 的子节点，那么只需要对 p 做下滤即可。但具体实现上，只是从下往上，由深向浅的遍历所有位置，做一次下滤即可。

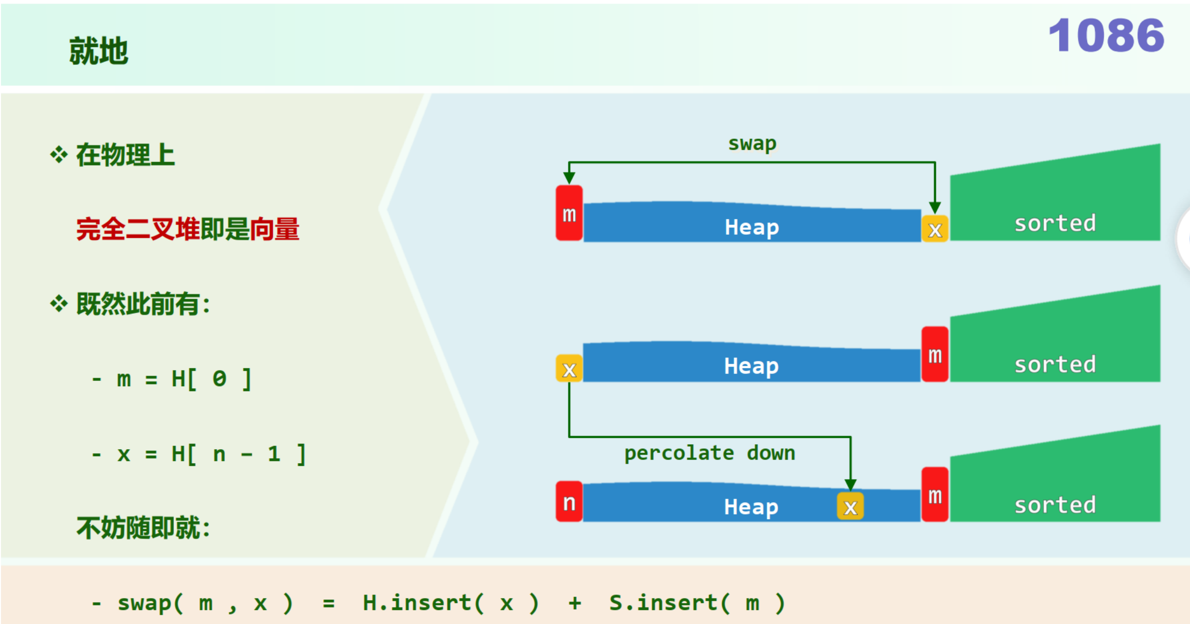
时间复杂度 $O(n)$ 。

堆排序

本质：用堆来选择排序。

时间复杂度：$O(n\log n)$ 空间复杂度： $O(1)$

操作技巧：在一个vector内实现排序和建堆，vector的前面用来保存堆，每次交换首元素和堆底元素，从而扩大排序好的向量，并使下次要下滤的元素就位。



多叉堆

顾名思义，也就是每个节点有 d 个孩子。

（插入）上滤成本： $\log_d n$ // （删除）下滤成本：$d \cdot \log_d n = \frac{d\cdot\ln 2}{\ln d} \cdot \log_2 n$

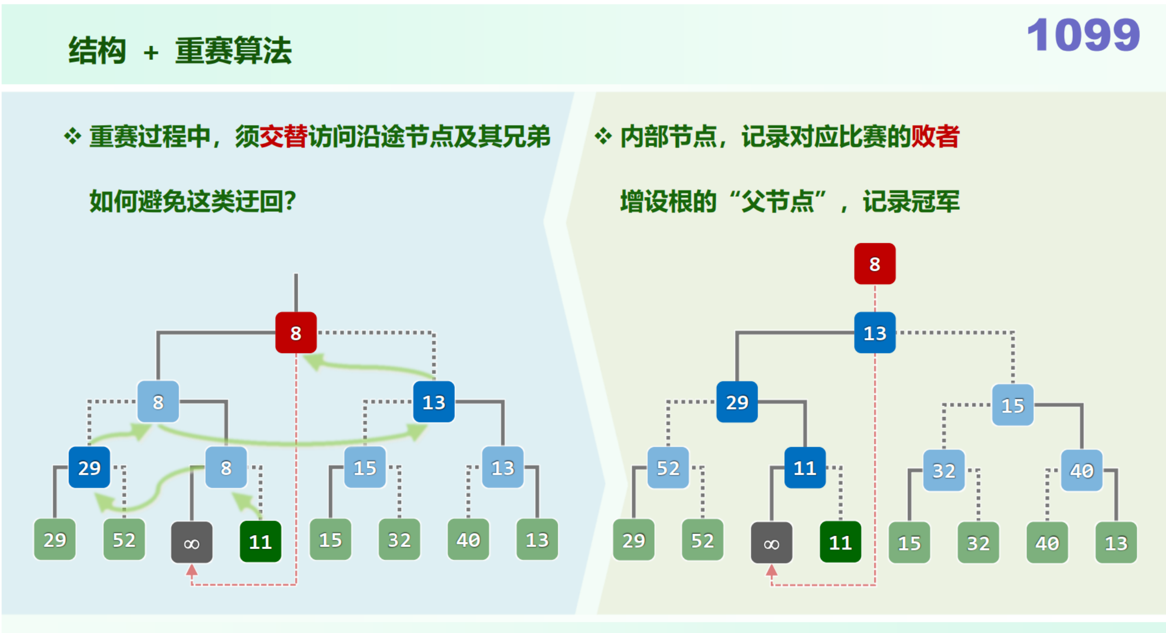
那么，对于图的 PFS 来说，就有： $n \cdot d \cdot \log_d n + e \log_d n$

$d \approx e/n + 2$ 时候，对于稀疏图是 $O(n\log n)$ ，对于稠密图是 $O(e)$ ，这里的d可以理解为图的平均度数、

仅从单次操作的角度来看，三叉树比二叉树操作复杂度要低（$\frac{2}{ln2} > \frac{3}{ln3}$）

锦标赛树

胜者树&败者树



左式堆

C.A.Crane, 1972; Knuth, 1973;

思想

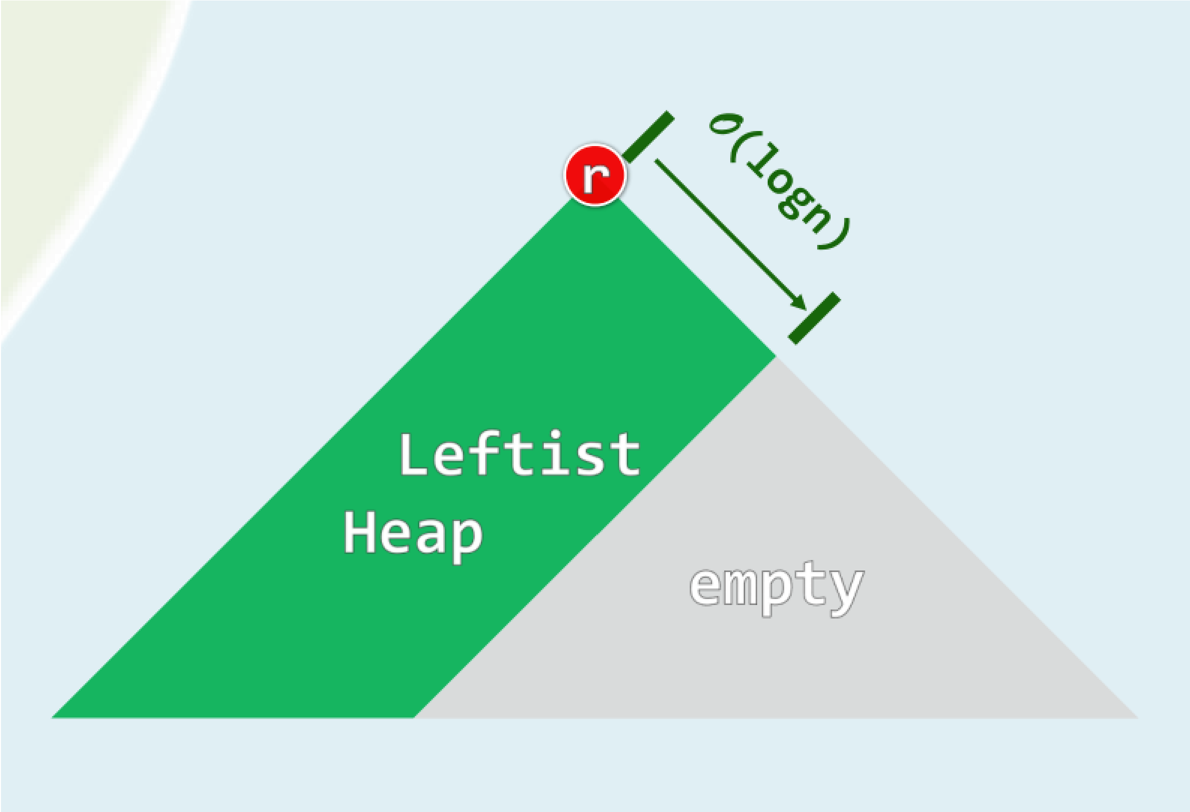
沿藤合并的时间与藤的长度成正比。本质上是一个归并排序。

于是要控制藤的长度。不妨使用最右侧的藤。

让节点分布趋于左侧而让合并操作趋于右侧。

是的，实际上，结构性并非堆结构的本质要求。

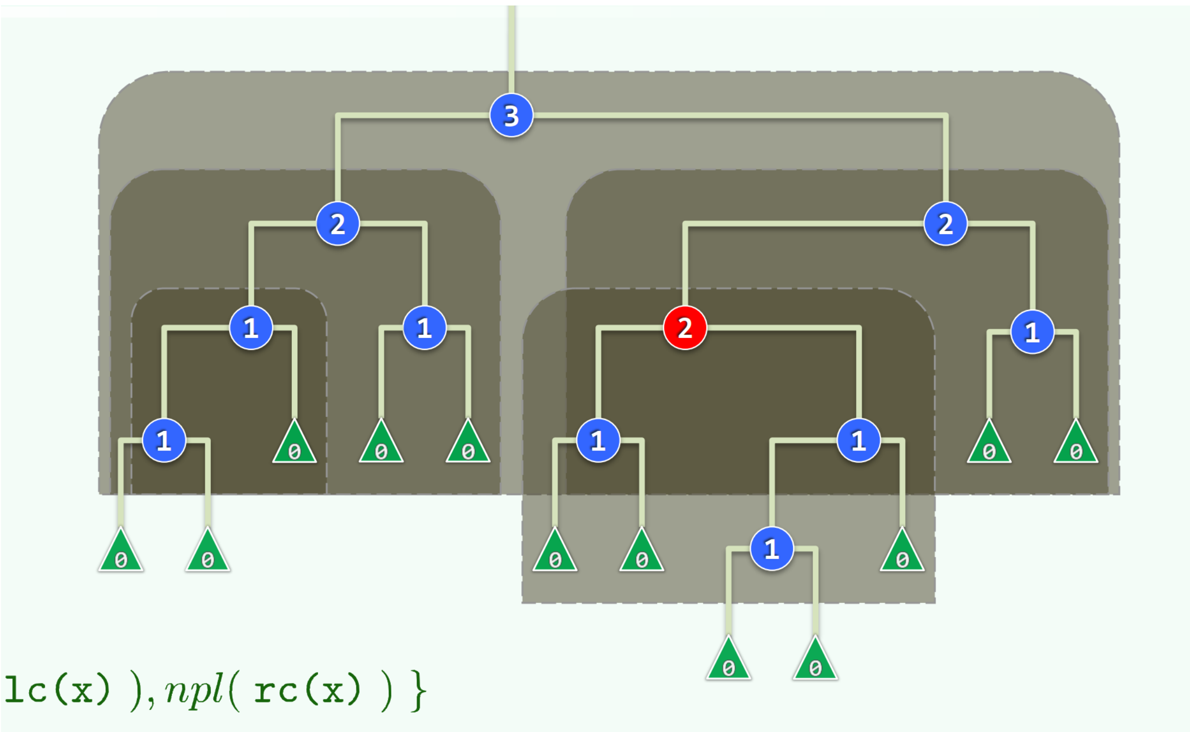
左倾



空节点路径长度

`npl(x)` 最短空节点路径，和定义和结点高度刚好相反。

`npl(x)`另一形象定义是x到外部节点的最短距离/以x为根的最大满子树高度。



左倾性

`npl(lc(x)) >= npl(rc(x))`

那么，就有 `npl(x) = npl(rc(x)) + 1`.

也就是，恰好等于右侧链的长度 + 1。

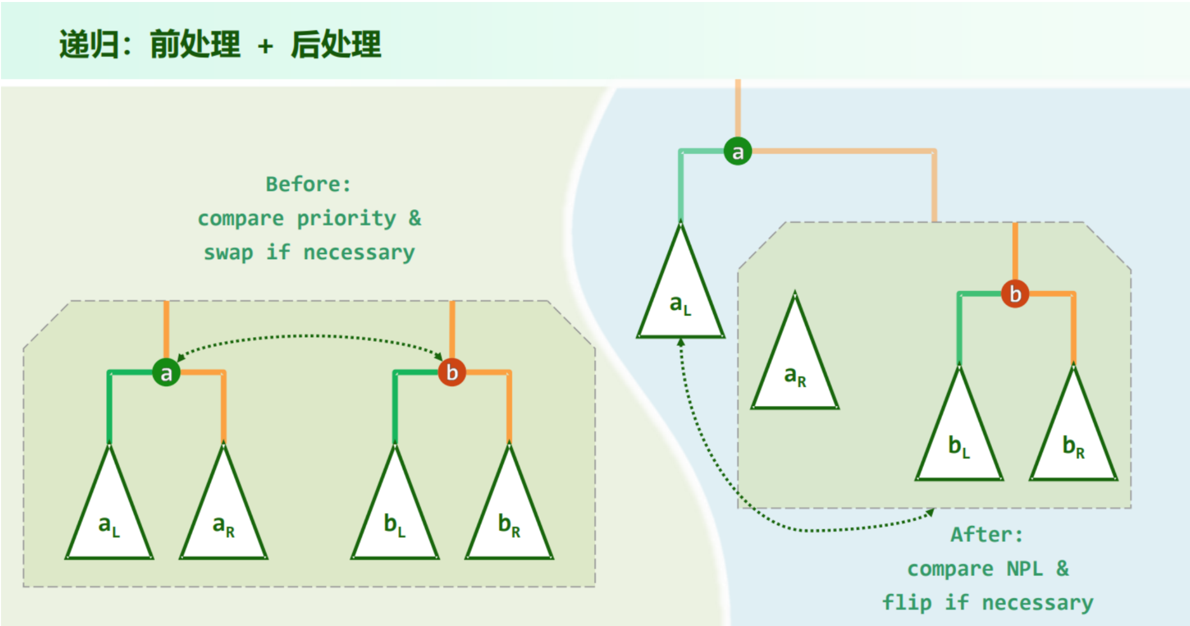
复杂度分析

从上图中可以看出，右侧链一定是最短的链，所以一定包括一个高度为 d 的完全二叉树，所以 $d = O(\log n)$ 。

操作

合并

合并是左式堆的基本操作。往右子树上合并。如果合并后左右子树的 npl 出问题，那么再交换左右子树



插入

把一个节点和一棵树合并。

删除

把根节点的左右子树合并。

优先级搜索树：Treap