10.优先级队列

(xb) 多叉堆

邓俊辉

deng@tsinghua.edu.cn

优先级搜索

❖ 回顾图的优先级搜索以及统一框架：g->pfs()...

❖ 无论何种算法，差异仅在于所采用的优先级更新器 `prioUpdater()`

```
Prim算法： g->pfs( 0, PrimPU() );
Dijkstra算法： g->pfs( 0, DijkstraPU() );
```

❖ 每一节点引入遍历树后，都需要

更新树外顶点的优先级（数），并

选出新的优先级最高者

❖ 若采用邻接表，两类操作的累计时间，分别为 $\mathcal{O}(n + e)$ 和 $\mathcal{O}(n^2)$

❖ 能否更快呢？

优先级队列

- 自然地，PFS中的各顶点可组织为 优先级队列 形式
- 为此需要使用 PQ 接口

heapify():	由n个顶点创建初始PQ	总计$\mathcal{O}(n)$
delMax():	取优先级最高（极短）跨边(u, w)	总计$\mathcal{O}(\boxed{n} * \log n)$
increase():	更新所有关联顶点到U的距离，提高优先级	总计$\mathcal{O}(\boxed{e} * \log n)$

- 总体运行时间 = $\boxed{\mathcal{O}(\boxed{(n + e)} * \log n)}$

  对于 稀疏图 ，处理效率很高；对于 稠密图 ，反而不如常规实现的版本

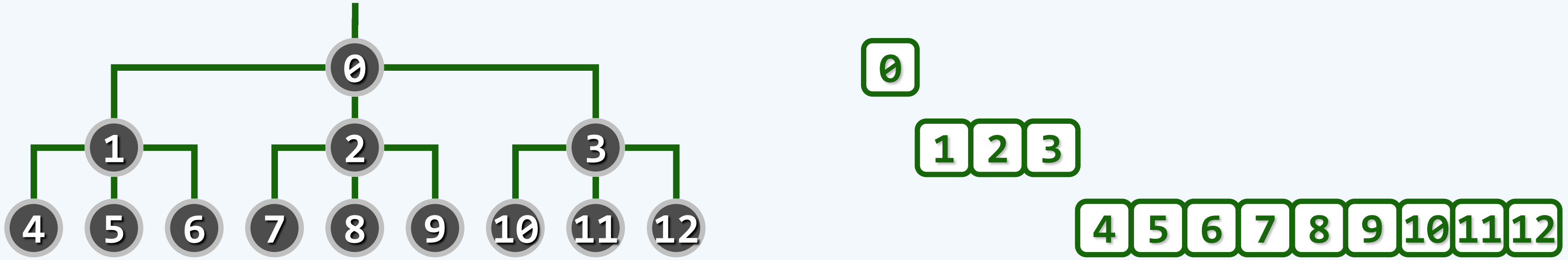
- 有无更好的算法？如果PQ的接口效率能够更高的话...
- 不太现实？异想天开？不妨先试试...

多叉堆

- heapify() : $\mathcal{O}(n)$ 不可能再快了 //直接写入，亦不过如此

  delMax() : $\mathcal{O}(\log n)$ 实质就是percolateDown() //已是极限了

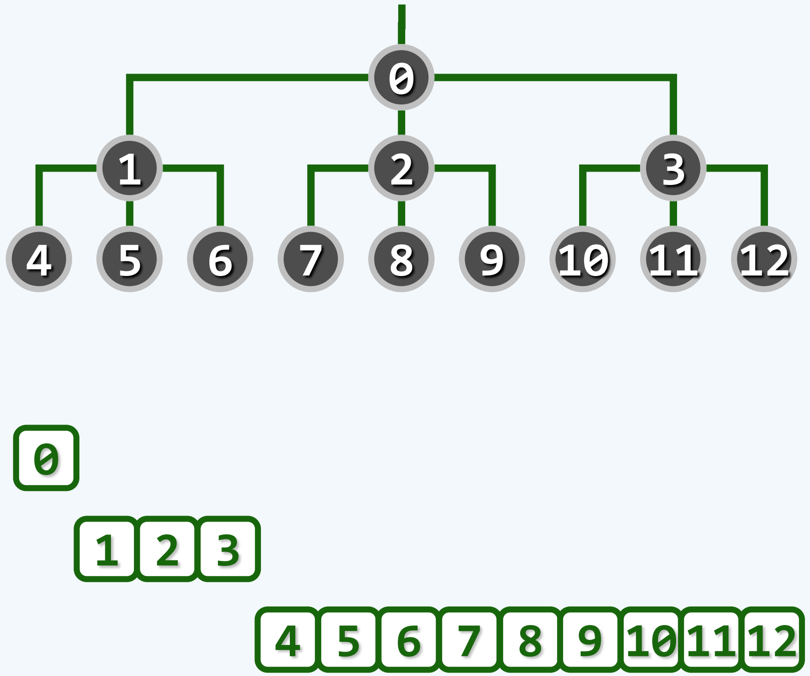
  increase() : $\mathcal{O}(\log n)$ 实质就是percolateUp() //似乎仍有余地



多叉堆

❖ 若将 $\boxed{二叉}$ 堆改成 $\boxed{多叉}$ 堆（d-heap）

则堆高降至 $\boxed{\mathcal{O}(\log_d n)}$



❖ $\boxed{上山容易下山难}$：

$\boxed{上滤}$ 成本可降至 $\boxed{\log_d n}$，但

$\boxed{下滤}$ 成本却增至

$\boxed{d * \log_d n} > \boxed{(d * \ln 2 / \ln d) * \log_2 n}$

❖ 对于稠密图，两类操作的次数相差悬殊 —— 故而 $\boxed{利大于弊}$...

多叉堆

❖ 如此，PFS的运行时间将是：

$n * d * \log_d n \quad + \quad e * \log_d n$

$(n * d + e) * \log_d n$

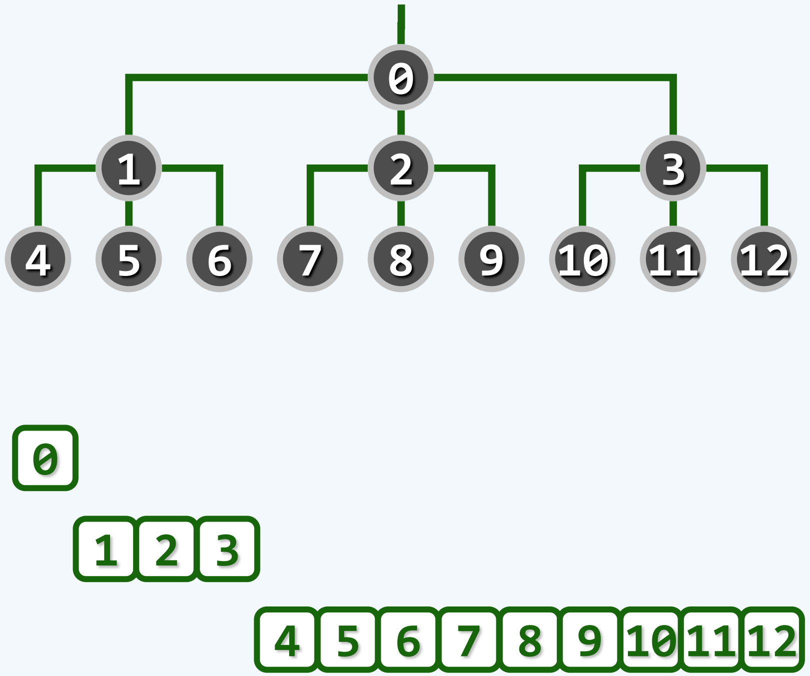
❖ 两相权衡，大致取 $\boxed{d = e/n + 2}$ 时

总体性能达到最优的

$$\boxed{\mathcal{O}(e * \log(e/n + 2, n))}$$

❖ 对于 稀疏 图，接近于 $\boxed{\mathcal{O}(n\log n)}$ //保持高效

对于 稠密 图，接近于 $\boxed{\mathcal{O}(e)}$ //改进极大

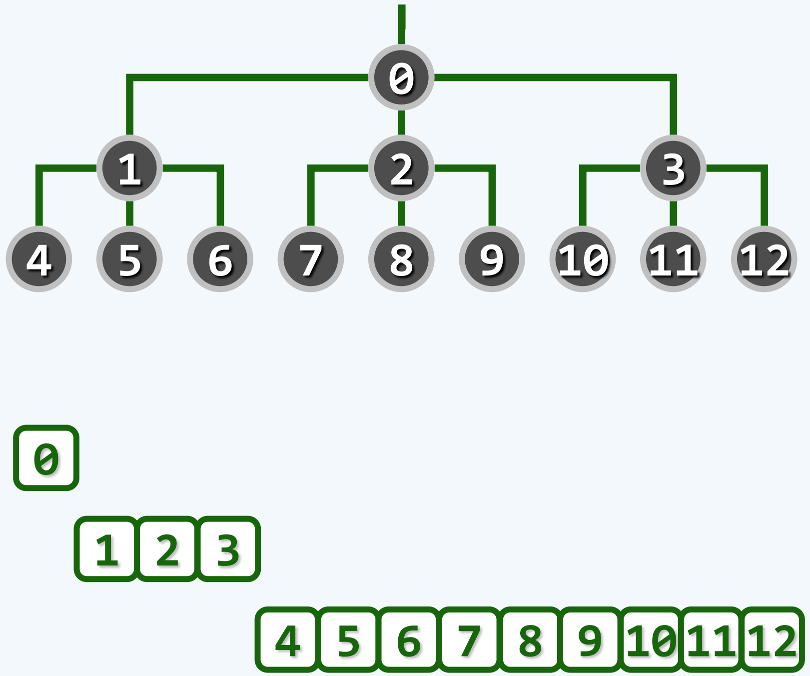


多叉堆

- 实现方面，依然可以基于向量

  $$\mathrm{parent}(\ k\ ) = \lfloor (k - 1) / d \rfloor$$

  $$\mathrm{child}(\ k,\ i\ ) = kd + i, \quad 0 < i \le d$$



- 当然，d不再是2的幂时

  将不再能够借助移位加速秩的换算

- 不过反过来，特别适用于

  不主要取决依赖于秩换算效率的场合

  比如，数据规模大到需要跨越存储层次时 //策略上，与B-树完全一致

Fibonacci堆

❖ 左式堆 x （ 新的上滤算法 + 懒惰合并 ）

❖ 各接口的 $\boxed{\text{分摊}}$ 复杂度

delMax()	$\mathcal{O}(\boxed{\log n})$
insert()	$\mathcal{O}(\boxed{1})$
merge()	$\mathcal{O}(\boxed{1})$
increase()	$\mathcal{O}(\boxed{1})$

❖ 于是，基于 $\boxed{\text{PFS框架}}$ 的算法采用Fibonacci堆后，运行时间自然就是

$$n * \mathcal{O}(\log n) + e * \mathcal{O}(1) = \mathcal{O}(\boxed{e + n\log n})$$