CSC263 Jan 15, 2018

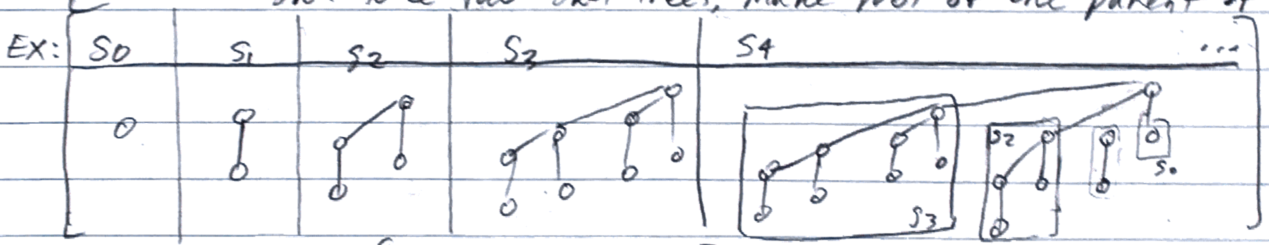
| ADT | DS | INSERT | MIN | EX-MIN | MERGE |
|---|---|---|---|---|---|
| Priority Queues | Heaps | ✓ | ✓ | ✓ | × |
| Mergeable PQs | Binomial Queues | $O(\log n)$ | $O(\log n)$ | $O(\log n)$ | $O(\log n)$ |

- If a CPU has two cores, each with a PQ of tasks, they might need to merge them. Not possible with heaps, possible with ...
- Binomial Heaps:

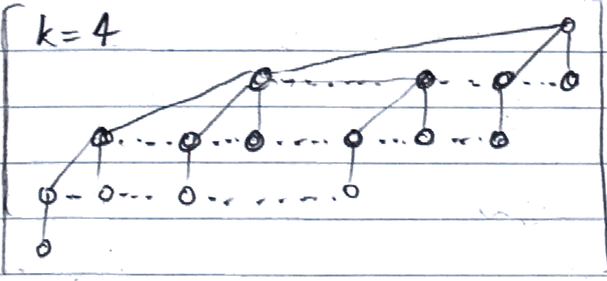
  $S_k$ tree: $S_0 = O$

  $S_k$: take two $S_{k-1}$ trees, make root of one parent of other

Ex:



- $S_k$ attached to $\{S_0, S_1, S_2, \dots S_{k-1}\}$.
- $\binom{k}{d}$ nodes at depth $d$
- $S_k$ tree has $2^k$ nodes.



| | | |
|---|---|---|
| d = 0 | nodes = | $1 = \binom{4}{0}$ |
| d = 1 | nodes = | $4 = \binom{4}{1}$ |
| d = 2 | | $6 = \binom{4}{2}$ |
| d = 3 | | $4 = \binom{4}{3}$ |
| d = 4 | | $1 = \binom{4}{4}$ |

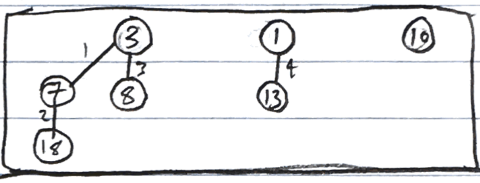
- a binomial forest of size $m$ (denoted $F_m$) is a sequence of $S_k$ trees with increasing $k$, with $m$ total nodes
  - $S_k$ trees have multiples of 2 nodes for all $k$
  - ∴ represent $m$ in binary to find which $S_k$ trees are contained in the forest
  - Ex: $m = 7 = \langle 111 \rangle_2 = 2^2 + 2^1 + 2^0 \therefore F_7 \ni \{S_2, S_1, S_0\}$
  - $F_m$ with $m = \langle b_\ell, b_{\ell-1}, \dots, b_0 \rangle$: $F_m \equiv \langle S_j \text{ st. } b_j = 1 \rangle$ $(\ell = \lfloor \log_2 m \rfloor)$
  - $F_m$ has $\alpha(m)$ trees, where $\alpha(m) = \#$ 1's in binary representation of $m$
  - $F_m$ has $m - \alpha(m)$ edges

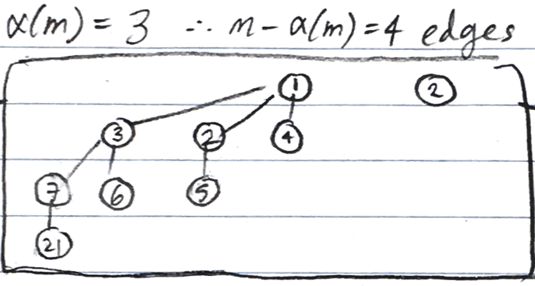
- min heap - has property that key of parent is smaller than key of children nodes
- note: "forest" is just the tree structure, it becomes a heap when keys are added and min property holds

$m = 7 = \langle 111 \rangle_2$ $F_m = \langle S_2, S_1, S_0 \rangle$

$S = \{10, 13, 1, 3, 8, 18, 7\}$

$\alpha(m) = 3 \therefore m - \alpha(m) = 4$ edges



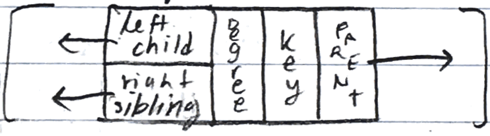


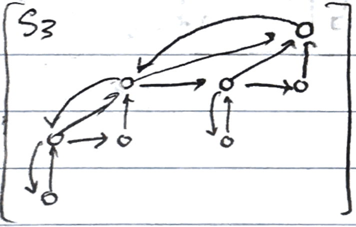
$m = 9 = \langle 1001 \rangle$; $F_m = \{S_4, S_0\}$

$\alpha(m) = 2 \therefore m - \alpha(m) = 7$ edges

$S = \{6, 1, 7, 3, 4, 5, 2, 2, 21\}$

- these forests must have pointers to allows us to navigate between trees & interact with the DS. These pointers are not edges.
- every node has the same pointer structure:





here we see the pointers visualized on an $S_3$ tree