2

Tuesday, September 17, 2019 2:16 PM

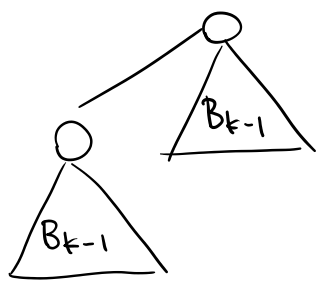
	Binary Heap	Binomial Heap
Insert	$\Theta(\log n)$	$\Theta(\log n)$
Min	$\Theta(1)$	
Extract-Min	$\Theta(\log n)$	
Union*	$\Theta(n)$	

Copy into new array and Build()

* Union(S_1, S_2): return a new pq with the union of S_1 and S_2.

Binomial Trees

- Construction: B_k :



and B_0 : O

1) B_k : 2^k nodes, 2) B_k has height k

3) the root of B_k has degree k

4) Hanging from the root of B_k are $B_0, \ldots, B_{k-1}$

- Binomial Forests

Def: Collection of binomial trees containing ≤ 1 B_k for each k.

Claim: There is a unique binomial forest with n nodes for every $n \geq 0$.

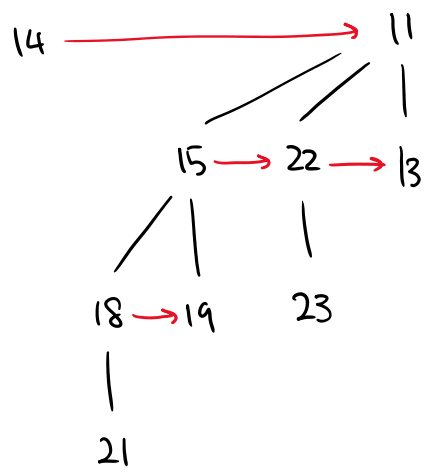
Write n in binary as $\overline{b_{k-1} b_{k-2} \cdots b_0}$ where $b_i \in \{0, 1\}$.

- Binomial Heaps

  Def: binomial forest, every node stores an element, such that the keys are heap-ordered.

  Binomial forests with n nodes has $\lceil \log_2(n+1) \rceil = O(\log n)$ trees.

<u>Implementation</u> (2nd ed CLRS)



Each node keeps track of:

(determines k)

a) object, key, degree (down)

Pointers:
- b) parent
- c) left-most child
- d) right sibling ($\rightarrow$)

Operations:

- Min(): Search the roots of the trees with sibling pointers. As many as $\log(n)$ trees.

- Union(S_1, S_2):

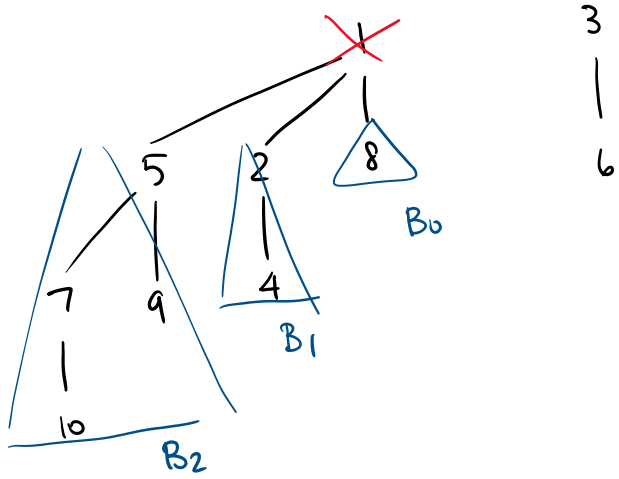
  Size of resulting trees can be computed with binary addition.

  Compute the sum and carry of the 2 trees.

c	x	y	c'	s
0	1	1		x+y
1	1	1	x+y	c

- Insert (S, x) : Trivial with Merge()

- Extract Min (S) :



Remove root, generate more binomial trees (new binomial heap) $O(\log n)$

Then, merge the new and the original. $O(\log n)$

Dictionary ADT

Maintains a set S of elements and supports:

- Insert (S, x)
- Find (S, x)
- Delete (S, x)