

Min-Heap Operations

	Binary Heap (worst-case)	Binomial Heap (worst-case)	Fibonacci Heap (amortized)
<i>Make-Heap()</i>	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
<i>Insert(H, x)</i>	$\Theta(\lg n)$	$O(\lg n)$	$\Theta(1)$
<i>Minimum(H)</i>	$\Theta(1)$	$O(\lg n)$	$\Theta(1)$
<i>Delete-Min(H)</i>	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$
<i>Delete(H, x)</i>	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(\lg n)$
<i>Decrease(H, x, k)</i>	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$
<i>Union(H₁, H₂)</i>	$\Theta(n)$	$O(\lg n)$	$\Theta(1)$

1

Binomial Heaps (cont)

Properties

B_k has 2^k nodes.

$$|B_k| = |B_{k-1}| + |B_{k-1}| = 2^{k-1} + 2^{k-1} = 2^k$$

B_k has height k .

$$h(B_k) = h(B_{k-1}) + 1$$

The root of B_k has degree k .

$$\deg(\text{root}(B_k)) = \deg(\text{root}(B_{k-1})) + 1$$

3

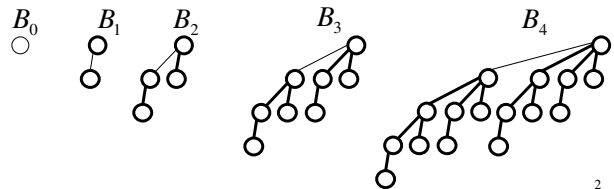
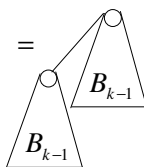
Binomial Heaps

Children are ordered

A *binomial tree* is an ordered tree defined by:

$$B_0 = \circ$$

$$\text{For } k > 0 \quad B_k =$$

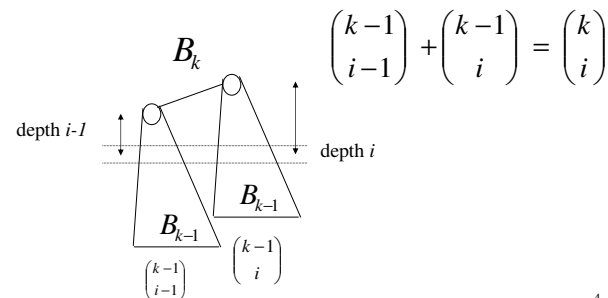


2

Binomial Heaps (cont)

Properties

B_k has $\binom{k}{i}$ nodes at depth i for $0 \leq i \leq k$.

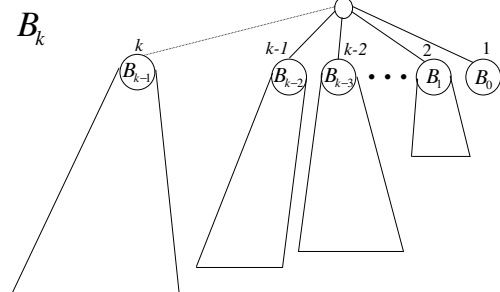


4

Binomial Heaps (cont)

Properties

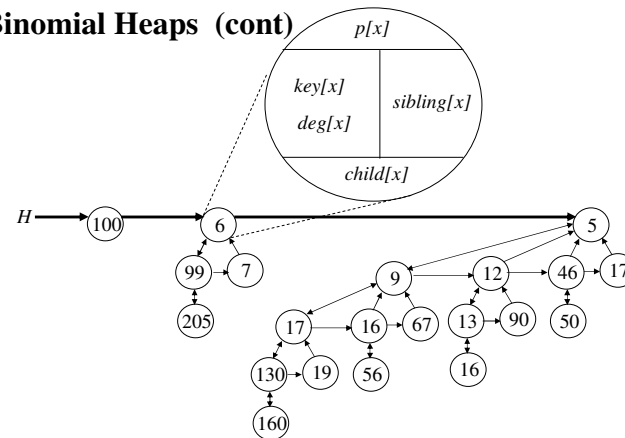
The i^{th} child of the root from right to left is B_i .



The maximum degree is $\lg n$ in a binomial tree with n nodes.

5

Binomial Heaps (cont)



7

Binomial Heaps (cont)

A *binomial heap* is a list of binomial trees such that

- 1) each tree is heap-ordered $\forall x \text{ key}[x] \geq \text{key}[p[x]]$
- 2) no two trees have roots of the same degree.

The largest degree in a Binomial Heap with n nodes is $\lfloor \lg n \rfloor$.

There are at most $\lfloor \lg n \rfloor + 1$ trees in a Binomial Heap with n nodes.

The trees are ordered by increasing degree.

The children of each node are ordered by decreasing degree.

6

Binomial Heaps (cont)

Operations

<i>Make-Heap()</i>	Create an empty list.	$\Theta(1)$
<i>Minimum(H)</i>	Scan tree list and return minimum key of roots.	$O(\lg n)$
<i>Decrease(H, x, k)</i>	Change key and then SiftUp. (Confined to the tree.)	$\Theta(\lg n)$
<i>Insert(H, x)</i>	Create a single node heap H' and then <i>Union(H, H')</i> .	$\Theta(1)$ + union time
<i>Delete(H, x)</i>	Change key of x to $-\infty$ and then <i>Delete-Min(H)</i> .	$\Theta(\lg n)$ + delete-min time

8

Binomial Heaps (cont)

Operations

Delete-Min(H) Scan tree list to find tree T with minimum key.

Remove T from H . Take child list of

T 's root and reverse its order to form a

new heap, H' and then $Union(H-T, H')$.

$\Theta(\lg n)$ + union time

↑
degree of root of T

9

Binomial Heaps (cont)

BINOMIAL-HEAP-UNION(H_1, H_2)

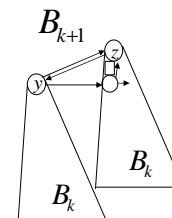
Merge of tree lists H_1 and H_2 ordered by degree

Scan merged list in order of non-decreasing root degrees and combine trees of the same degree.

For any k , there will be either 0,1,2 or 3 trees of the degree k .

If there are 2 or 3 trees of the degree k then link 2 of them to form a tree of degree $k+1$.

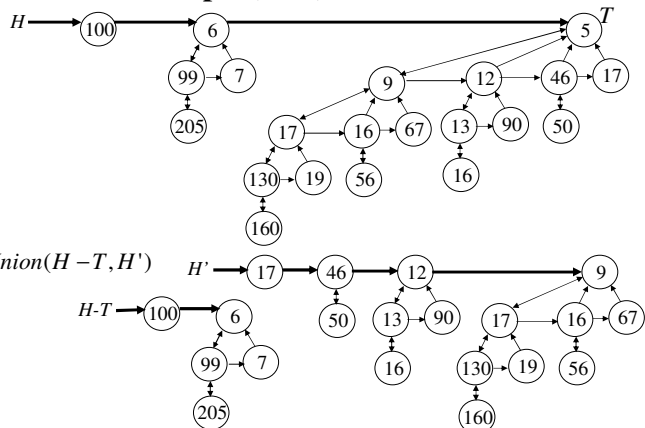
```
p[y] ← z
sibling[y] ← child[z]
child[z] ← y
degree[z] ++
```



11

Binomial Heaps (cont)

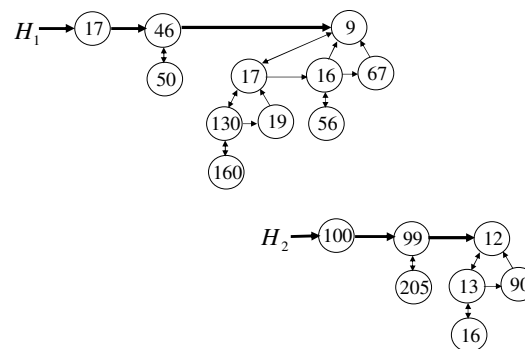
Delete-Min(H)



10

Binomial Heaps (cont)

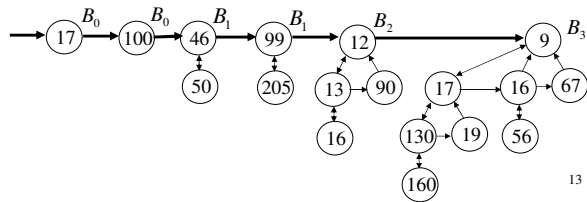
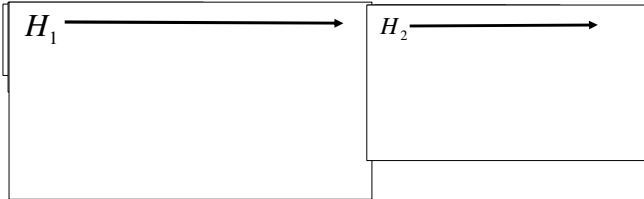
Union(H₁, H₂)



12

Binomial Heaps (cont)

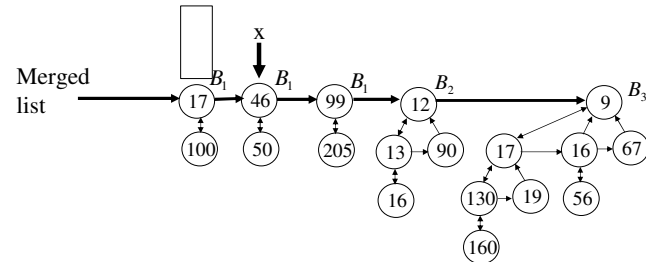
$Union(H_1, H_2)$



13

Binomial Heaps (cont)

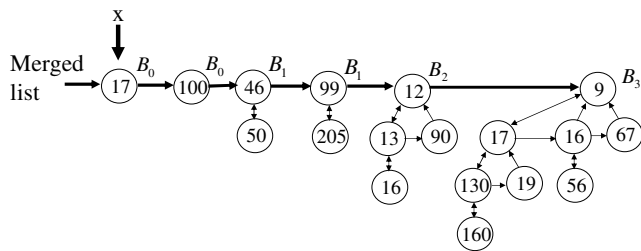
$Union(H_1, H_2)$



15

Binomial Heaps (cont)

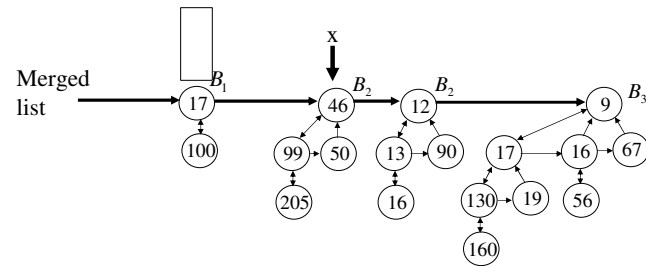
$Union(H_1, H_2)$



14

Binomial Heaps (cont)

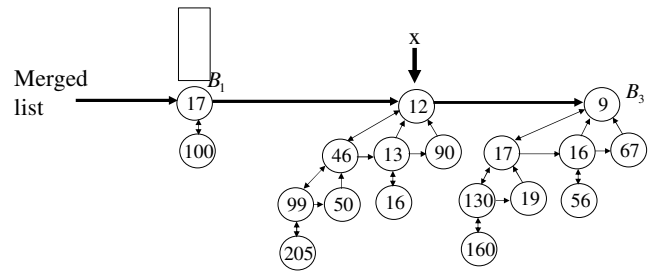
$Union(H_1, H_2)$



16

Binomial Heaps (cont)

Union(H_1, H_2)



Binomial Heaps (cont)

Union(H_1, H_2)

