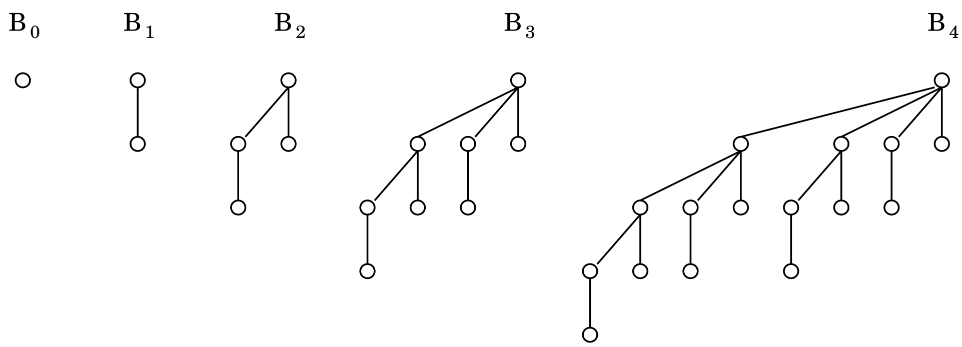
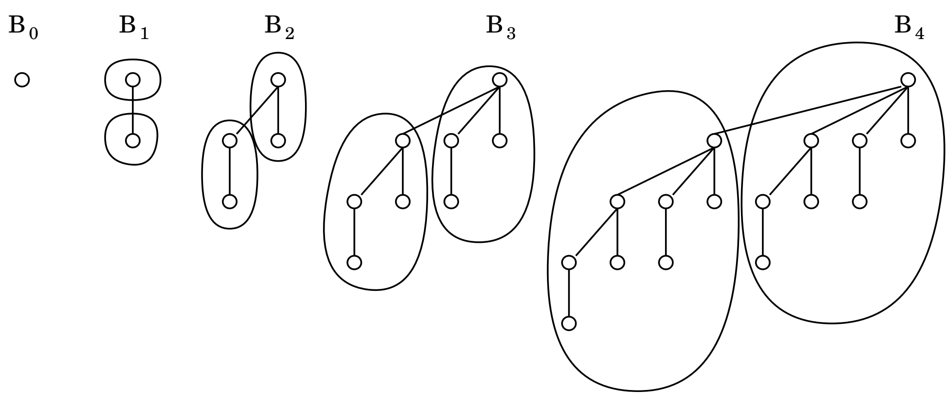
20-0: **Binomial Trees**

- B_0 is a tree containing a single node
- To build B_k:
  - Start with B_{k-1}
  - Add B_{k-1} as left subtree

20-1: **Binomial Trees**



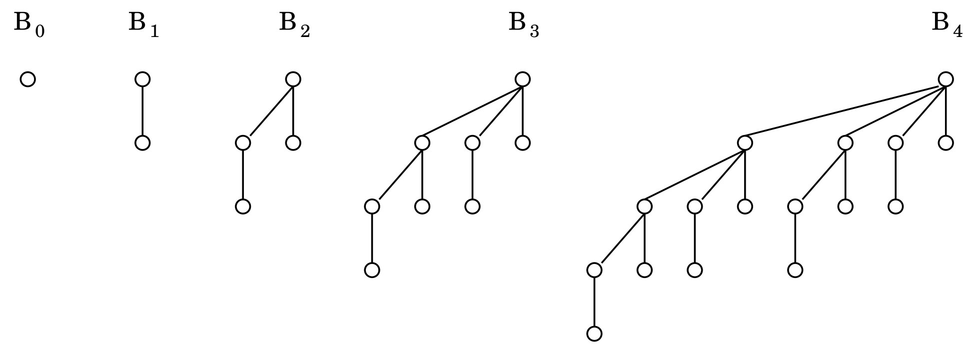
20-2: **Binomial Trees**



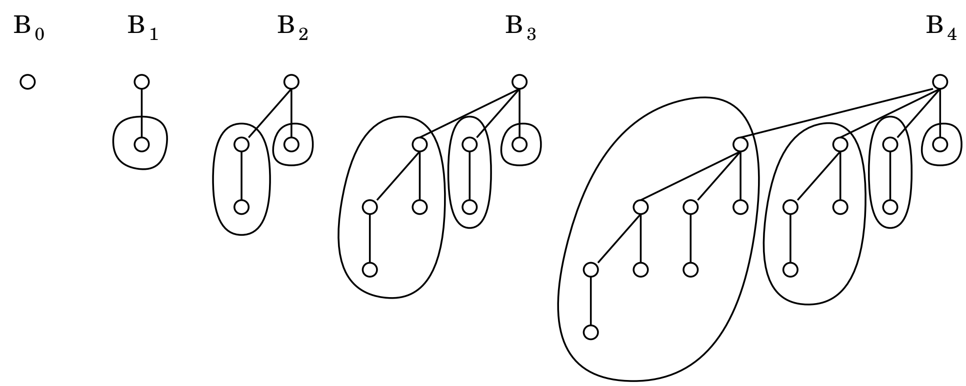
20-3: **Binomial Trees**

- Equivalent defintion
  - B_0 is a binomial heap with a single node
  - B_k is a binomial heap with k children:
    - $B_0 \ldots B_{k-1}$

20-4: **Binomial Trees**



20-5: **Binomial Trees**



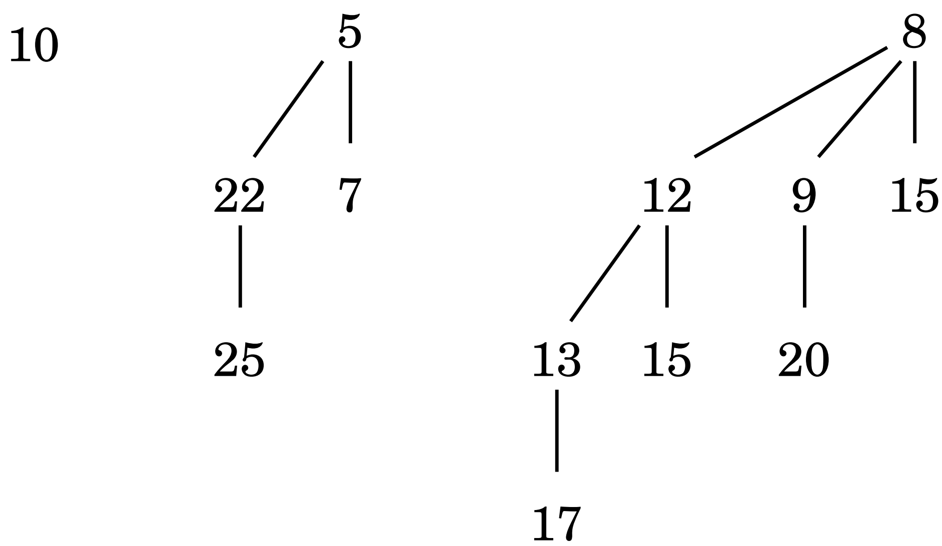
20-6: **Binomial Trees**

- Properties of binomial trees B_k
  - Contains 2^k nodes
  - Has height k
  - Contains $\binom{k}{i}$ nodes at depth i for $i = 0 \ldots k$

20-7: **Binomial Heaps**

- A Binomial Heap is:
  - Set of binomial trees, each of which has the heap property
    - Each node in every tree is $<=$ all of its children
  - All trees in the set have a different root degree
    - Can't have two B_3's, for instance

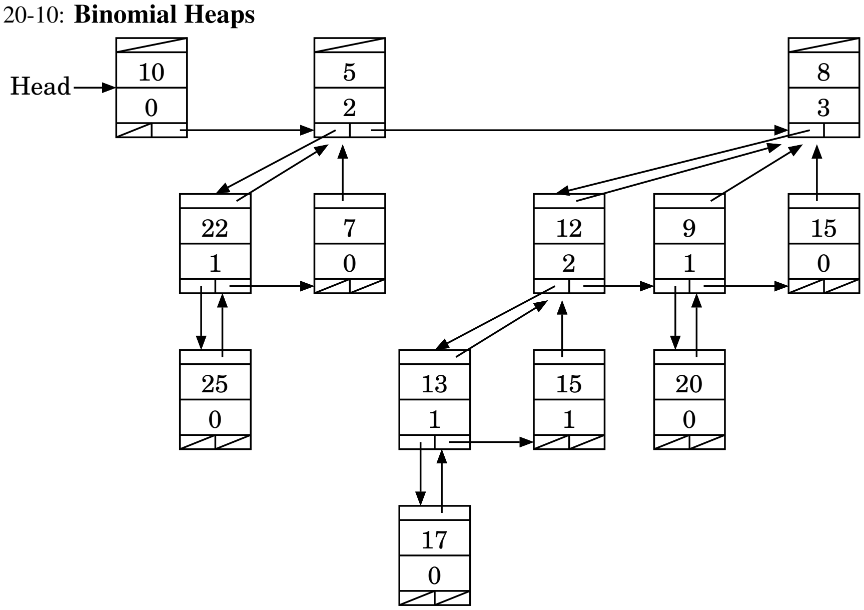
20-8: **Binomial Heaps**



20-9: **Binomial Heaps**

- Representing Binomial Heaps
  - Each node contains:
    - left child, right sibling, parent pointers
    - degreee (is the tree rooted at this node B_0, B_1, etc.)
    - data
  - Each list of children sorted by degree

20-10: **Binomial Heaps**



20-11: **Binomial Heaps**

- How can we find the minimum element in a binomial heap?

- How long does it take?

20-12: **Binomial Heaps**

- How can we find the minimum element in a binomial heap?
  - Look at the root of each tree in the list, find smallest value
- How long does it take?
  - Heap has n elements
  - Represent n as a binary number
  - B_k is in heap iff kth binary digit of n is 1
  - Number of trees in heap $\in O(\lg n)$

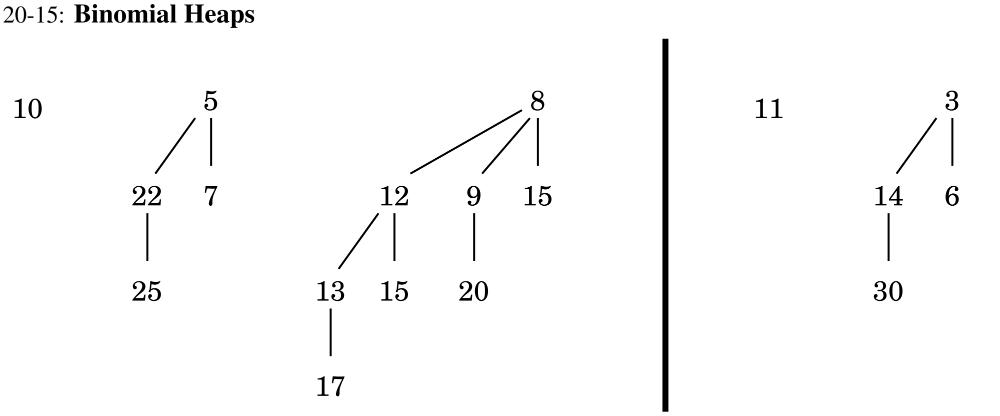
20-13: **Binomial Heaps**

- Merging Heaps H_1 and H_2
  - Merge root lists of H_1 and H_2
  - What property of binomial heaps may be broken?
  - How do we fix it?

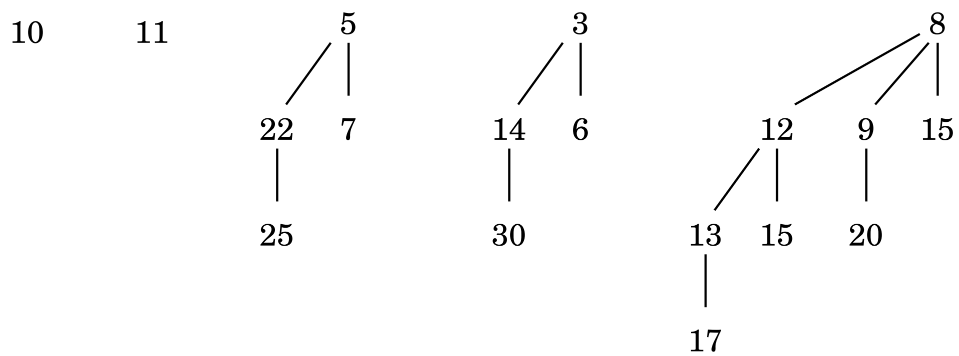
20-14: **Binomial Heaps**

- Merging Heaps H_1 and H_2
  - Merge root lists of H_1 and H_2
    - Could now have two trees with same degree
  - Go through list from smallest degree to largest degree
    - If two trees have same degree, combine them into one tree of larger degree
    - If three trees have same degree (how can this happen?) leave one, combine other two into tree of larger degree

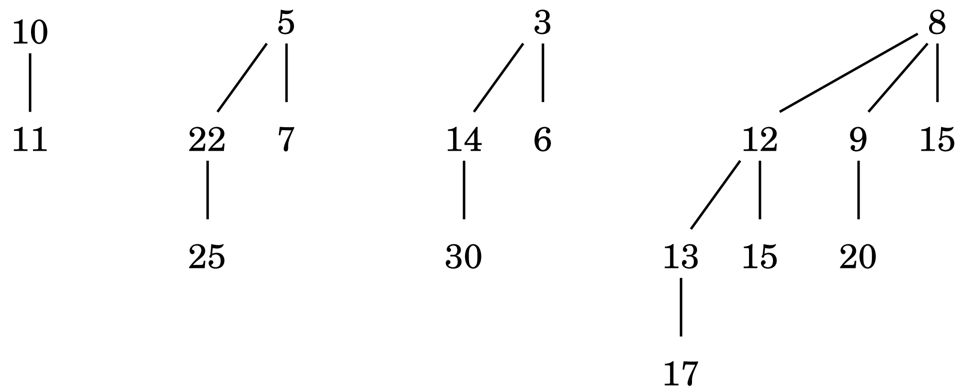
20-15: **Binomial Heaps**



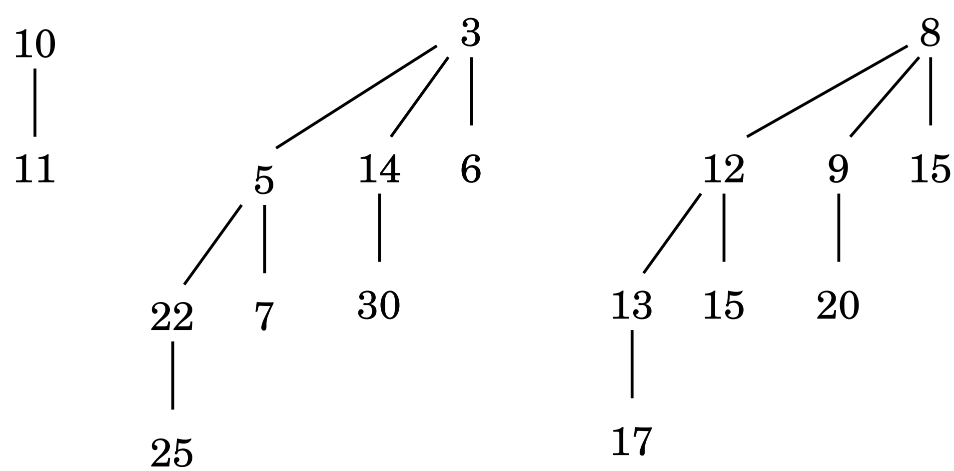
20-16: **Binomial Heaps**



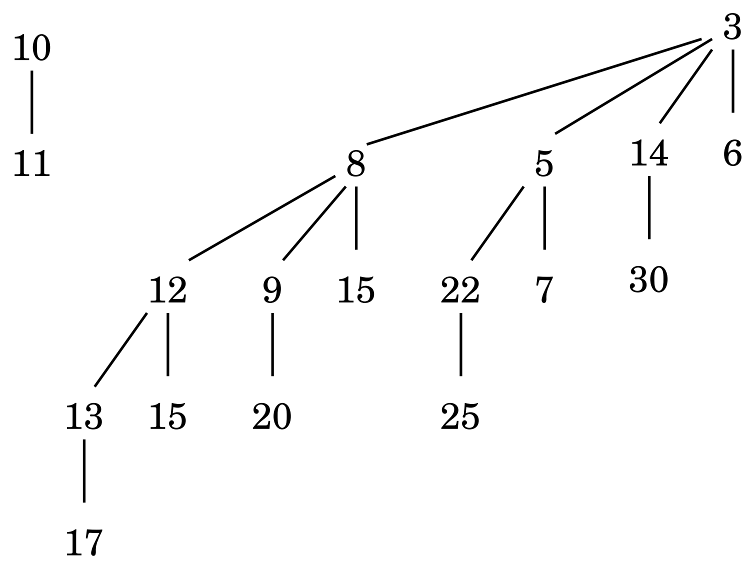
20-17: **Binomial Heaps**



20-18: **Binomial Heaps**



20-19: **Binomial Heaps**

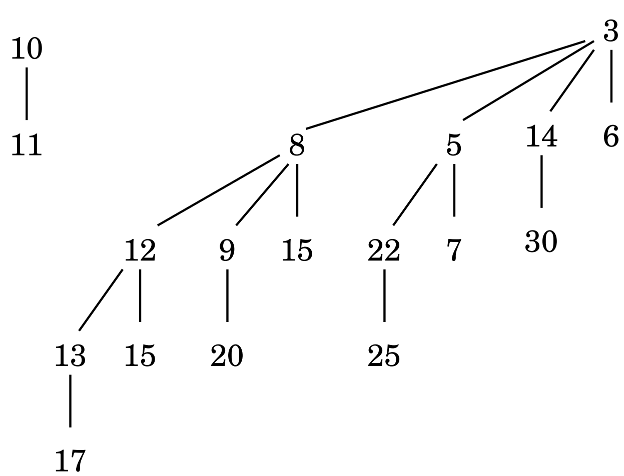


20-20: **Binomial Heaps**

- Removing minimum element
  - Find tree T that has minimum value at root, remove T from the list
  - Remove the root of T
    - Leaving a list of smaller trees
  - Reverse list of smaller trees
  - Merge two lists of trees together

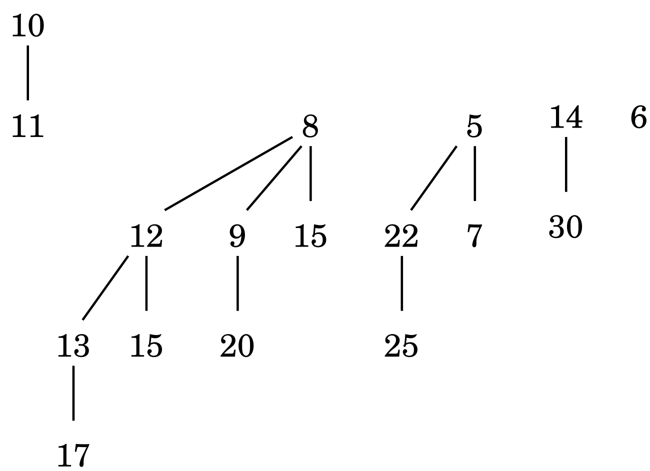
20-21: **Binomial Heaps**

- Removing minimum element



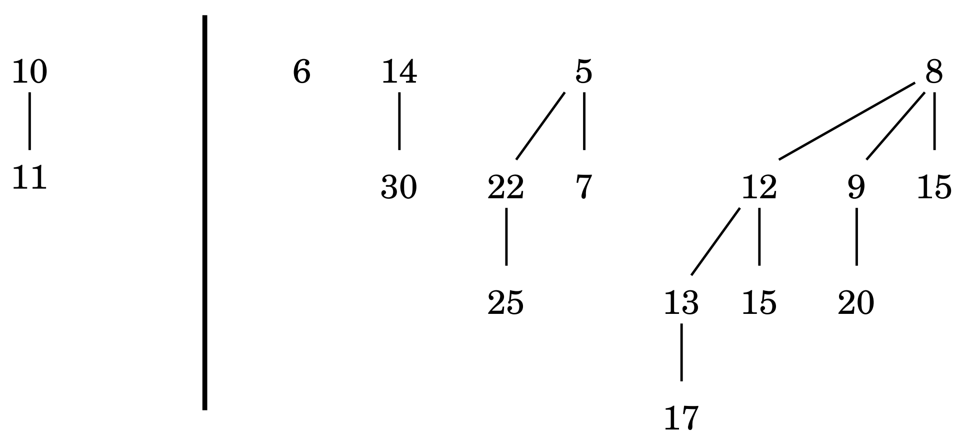
20-22: **Binomial Heaps**

- Removing minimum element



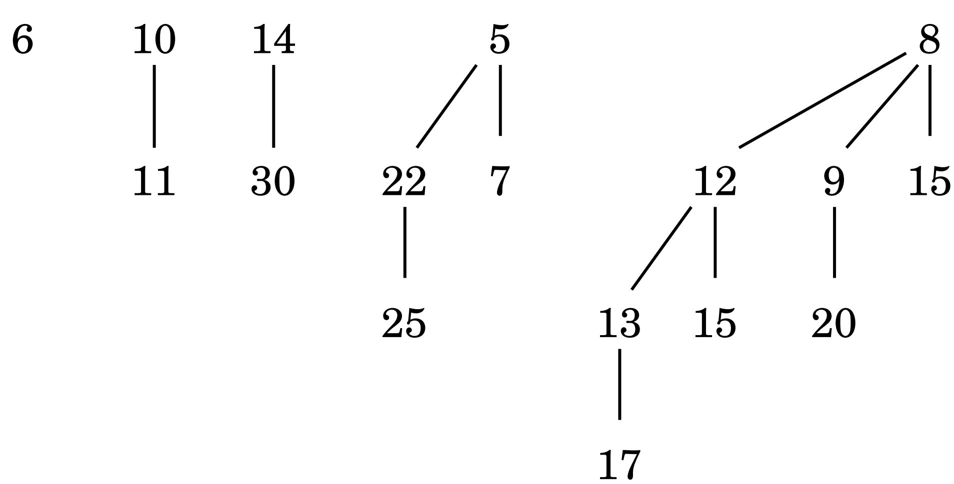
20-23: **Binomial Heaps**

- Removing minimum element



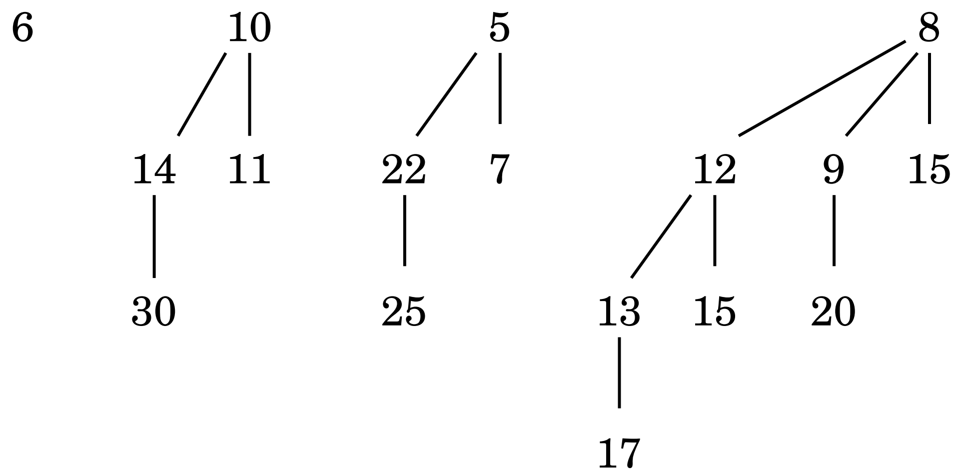
20-24: **Binomial Heaps**

- Removing minimum element



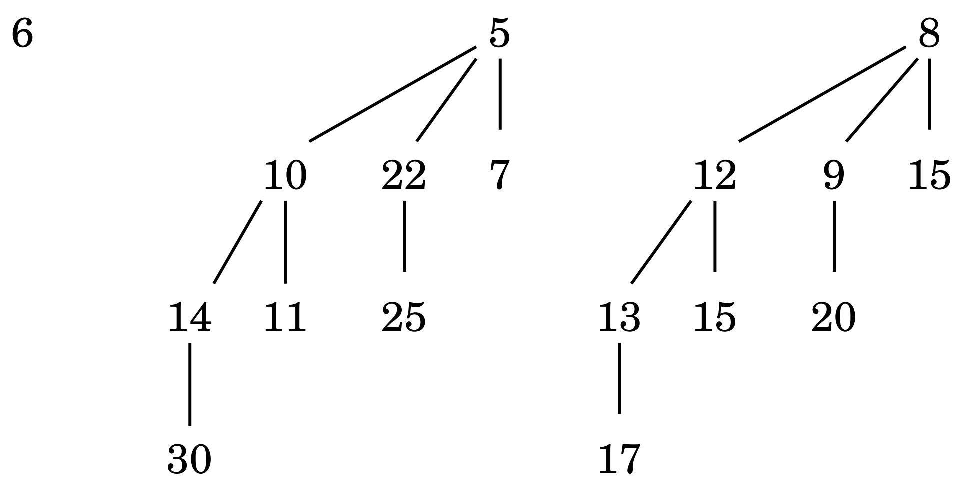
20-25: **Binomial Heaps**

- Removing minimum element



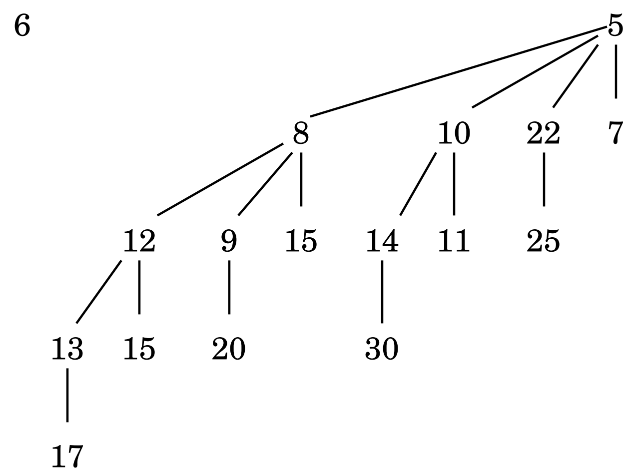
20-26: **Binomial Heaps**

- Removing minimum element



20-27: **Binomial Heaps**

- Removing minimum element



20-28: **Binomial Heaps**

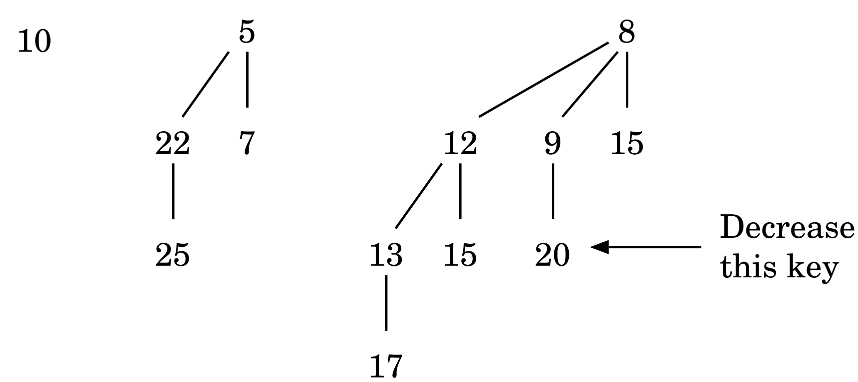
- Removing minimum element
  - Time?

20-29: **Binomial Heaps**

- Removing minimum element
  - Time?
    - Find the smallest element: $O(\lg n)$
    - Reverse list of children $O(\lg n)$
    - Merge heaps $O(\lg n)$

20-30: **Binomial Heaps**

- Decreasing the key of an element (assuming you have a pointer to it)



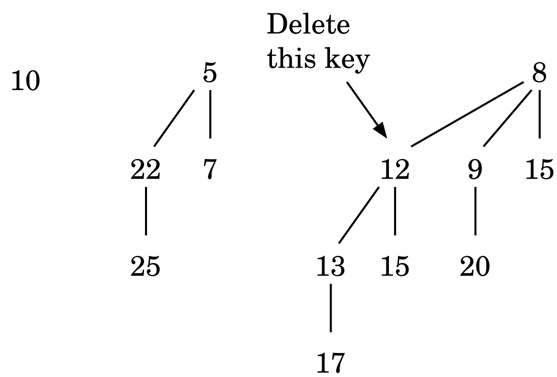
20-31: **Binomial Heaps**

- Decreasing the key of an element (assuming you have a pointer to it)
  - Decrease key value
  - While value < parent, swap with parent

- Exactly like standard, binary heaps
- Time: $O(\lg n)$

20-32: **Binomial Heaps**

- How could we delete an arbitrary element (assuming we had a pointer to this element)?



20-33: **Binomial Heaps**

- How could we delete an arbitrary element (assuming we had a pointer to this element)?
  - Decrease key to $-\infty$, Time $O(\lg n)$
  - Remove smallest, Time $O(\lg n)$