Binomial Heaps

Binomial Heaps

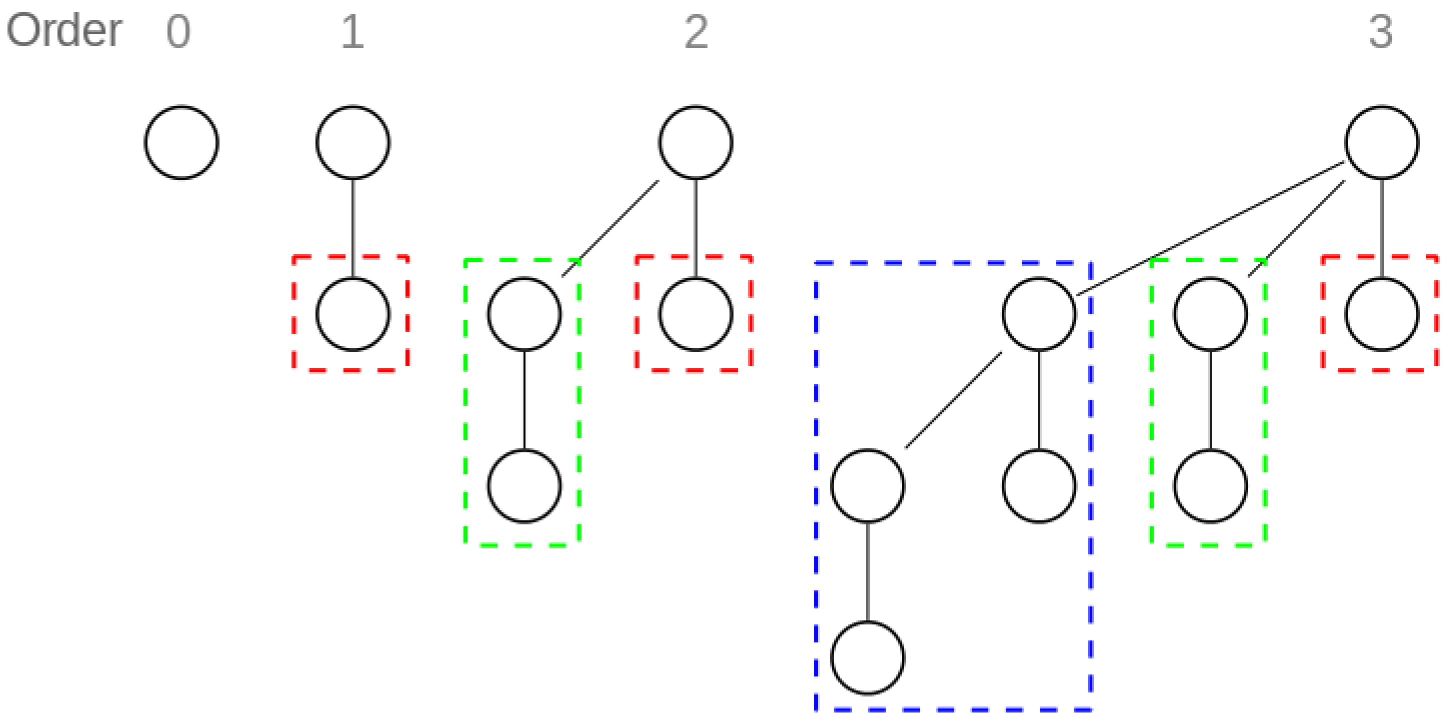
- Binomial Tree
  - A binomial tree of order 0 is a single node.

Binomial Heaps

- Binomial Tree
  - A binomial tree of order 0 is a single node.
  - In general, a binomial tree of order k (B_k) has a root node whose children are roots of binomial trees of orders k-1, k-2, ..., 2, 1, 0 (in this order (left-to-right)).

Binomial Heaps

- Binomial Tree
  - A binomial tree of order 0 is a single node.
  - In general, a binomial tree of order k (B_k) has a root node whose children are roots of binomial trees of orders k-1, k-2, ..., 2, 1, 0 (in this order (left-to-right)).



Binomial trees of order 0 to 3: Each tree has a root node with subtrees of all lower ordered binomial trees, which have been highlighted. For example, the order 3 binomial tree is connected to an order 2, 1, and 0 (highlighted as blue, green and red respectively) binomial tree.

Binomial Heaps

- Binomial Tree
  - A binomial tree of order k has 2^k nodes, and height k.

Binomial Heaps

- Binomial Tree
  - A binomial tree of order k has 2^k nodes, and height k.

  - The name comes from the shape: a binomial tree of order k has kC_d nodes at depth d, a binomial coefficient.

Binomial Heaps

- Binomial Tree
  - A binomial tree of order k has 2^k nodes, and height k.
  - The name comes from the shape: a binomial tree of order k has kC_d nodes at depth d, a binomial coefficient.
  - Because of its structure, a binomial tree of order k can be constructed from two trees of order k-1 by attaching one of them as the leftmost child of the root of the other tree.

Binomial Heaps

- Binomial Tree
  - A binomial tree of order k has 2^k nodes, and height k.
  - The name comes from the shape: a binomial tree of order k has kC_d nodes at depth d, a binomial coefficient.
  - Because of its structure, a binomial tree of order k can be constructed from two trees of order k-1 by attaching one of them as the leftmost child of the root of the other tree.
  - This feature is central to the merge operation of a binomial heap, which is its major advantage over other conventional heaps.

Binomial Heaps

- Binomial Tree
  - A binomial tree of order k has 2^k nodes, and height k.
  - The name comes from the shape: a binomial tree of order k has kC_d nodes at depth d, a binomial coefficient.
  - Because of its structure, a binomial tree of order k can be constructed from two trees of order k-1 by attaching one of them as the leftmost child of the root of the other tree.
  - This feature is central to the merge operation of a binomial heap, which is its major advantage over other conventional heaps.
  - The degree of the root node in a binomial tree is k.

Binomial Heaps

- Binomial Tree
  - A binomial tree of order k has 2^k nodes, and height k.
  - The name comes from the shape: a binomial tree of order k has kC_d nodes at depth d, a binomial coefficient.
  - Because of its structure, a binomial tree of order k can be constructed from two trees of order k-1 by attaching one of them as the leftmost child of the root of the other tree.
  - This feature is central to the merge operation of a binomial heap, which is its major advantage over other conventional heaps.
  - The degree of the root node in a binomial tree is k.
  - Deleting the root yields k binomial trees: B_{k-1}, B_{k-2}, …, B_0

Binomial Heaps

- A binomial heap is implemented as a **set** of binomial trees that satisfy the binomial heap properties:

Binomial Heaps

- A binomial heap is implemented as a set of binomial trees that satisfy the binomial heap properties:
  - Each binomial tree in a heap obeys the minimum-heap (assuming min binomial heap) property: the key of a node is greater than or equal to the key of its parent.

Binomial Heaps

- A binomial heap is implemented as a set of binomial trees that satisfy the binomial heap properties:
  - Each binomial tree in a heap obeys the minimum-heap (assuming min binomial heap) property: the key of a node is greater than or equal to the key of its parent.
  - There can be at most one binomial tree for each order, including zero order.

Binomial Heaps

- A binomial heap is implemented as a set of binomial trees that satisfy the binomial heap properties:
  - Each binomial tree in a heap obeys the minimum-heap (assuming min binomial heap) property: the key of a node is greater than or equal to the key of its parent.
  - There can be at most one binomial tree for each order, including zero order.

- The first property ensures that the root of each binomial tree contains the smallest key in the tree. It follows that the smallest key in the entire heap is one of the roots.

Binomial Heaps

- A binomial heap is implemented as a set of binomial trees that satisfy the binomial heap properties:
  - Each binomial tree in a heap obeys the minimum-heap (assuming min binomial heap) property: the key of a node is greater than or equal to the key of its parent.
  - There can be at most one binomial tree for each order, including zero order.

- The first property ensures that the root of each binomial tree contains the smallest key in the tree. It follows that the smallest key in the entire heap is one of the roots.

- The second property implies that a binomial heap with n nodes consists of at most 1 + floor($\log_2 n$) binomial trees. The height of the heap is also at most floor($\log_2 n$).

Binomial Heaps

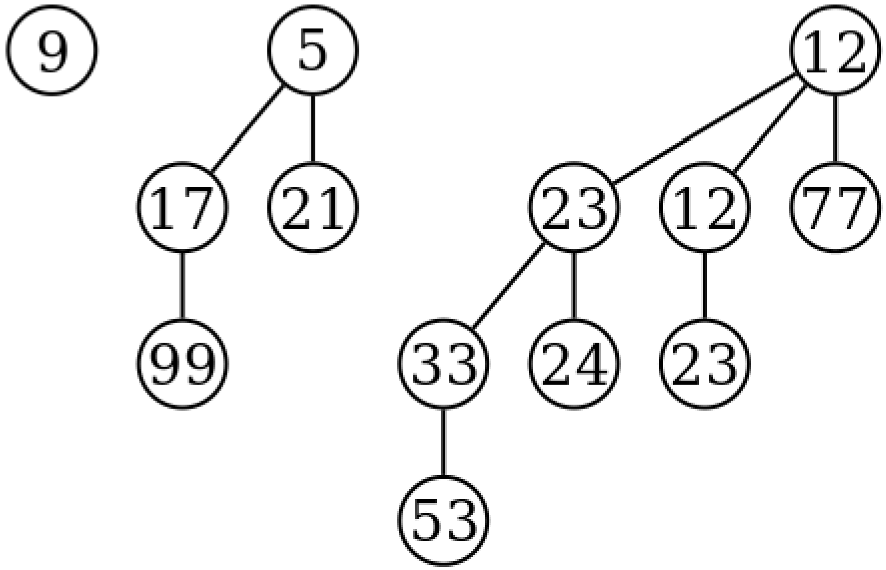
- A binomial heap is implemented as a set of binomial trees that satisfy the binomial heap properties:
  - Each binomial tree in a heap obeys the minimum-heap (assuming min binomial heap) property: the key of a node is greater than or equal to the key of its parent.
  - There can be at most one binomial tree for each order, including zero order.

- The first property ensures that the root of each binomial tree contains the smallest key in the tree. It follows that the smallest key in the entire heap is one of the roots.

- The second property implies that a binomial heap with n nodes consists of at most 1 + floor($\log_2 n$) binomial trees. The height of the heap is also at most floor($\log_2 n$).

- The number and orders of these trees are uniquely determined by the number of nodes n: there is one binomial tree for each nonzero bit in the binary representation of the number n. For example, the decimal number 13 is 1101 in binary, 2^3 + 2^2 + 2^0, and thus a binomial heap with 13 nodes will consist of three binomial trees of orders 3, 2, and 0.

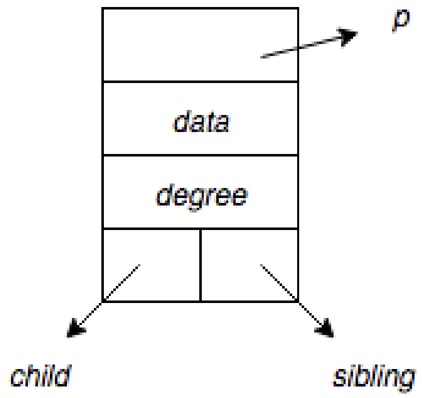
Binomial Heaps



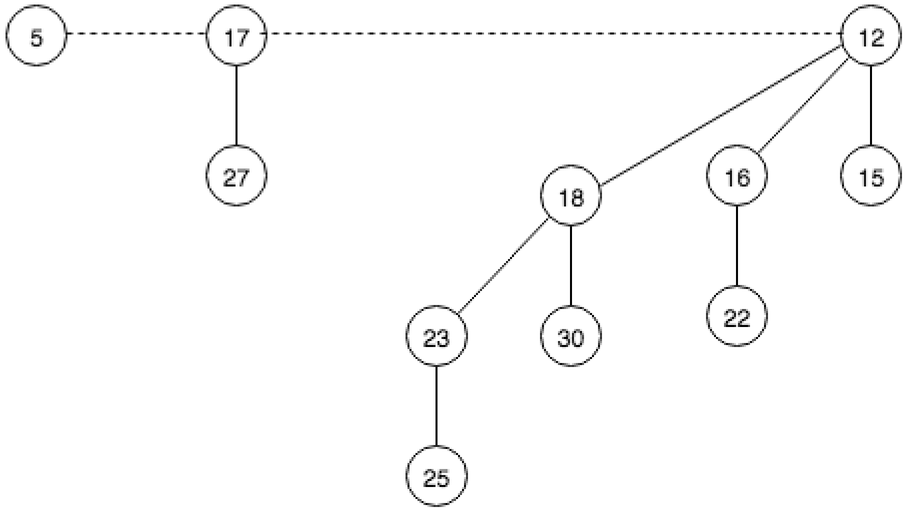
Example of a binomial heap containing 13 nodes with distinct keys. The heap consists of three binomial trees with orders 0, 2, and 3.

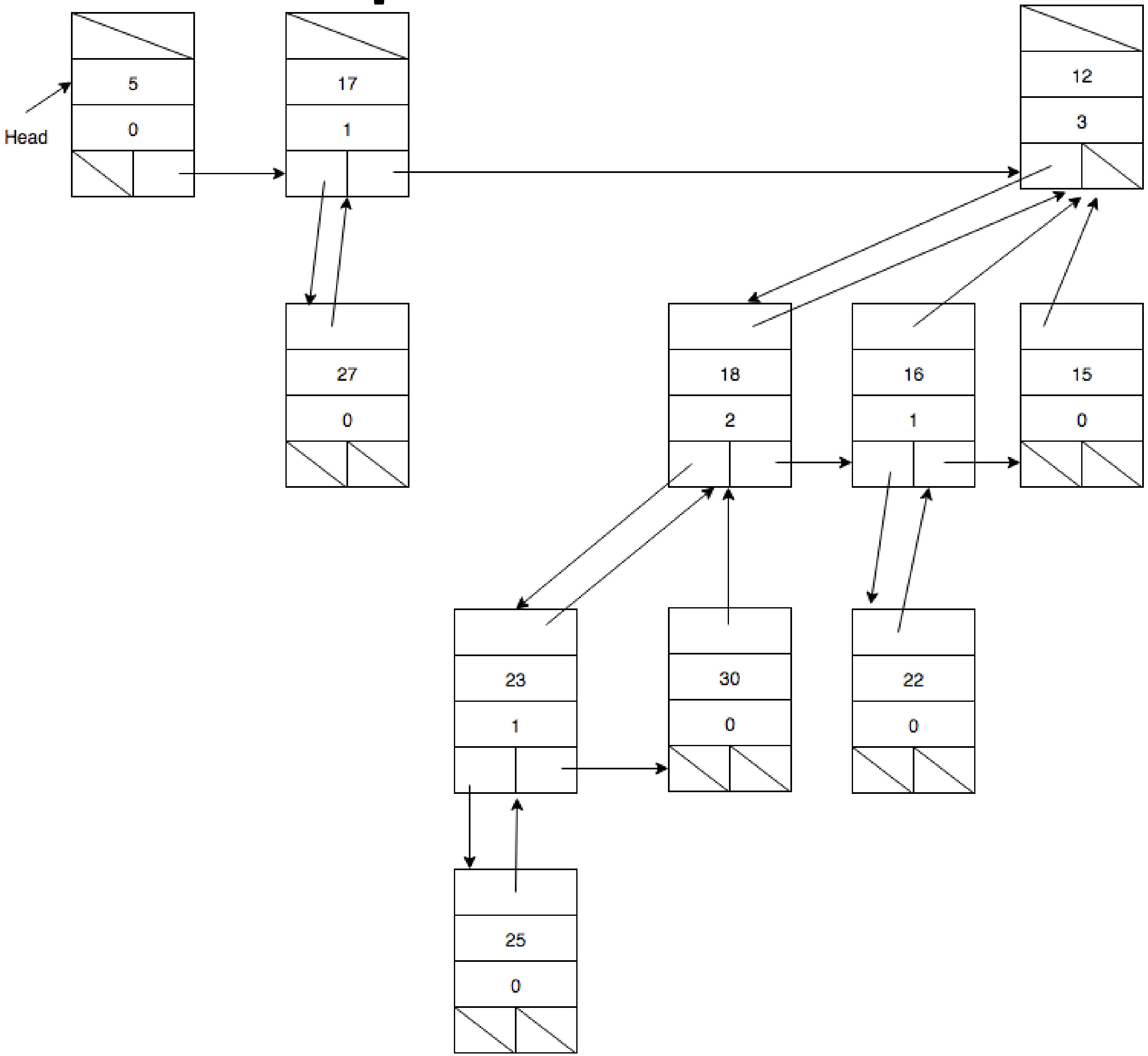
Binomial Heaps

- Each node in a binomial heap has 5 fields as follows.
  - p: A pointer to the parent node.
  - data: The key of the node.
  - degree: Number of children.
  - child: The pointer to the left-most child of the node.
  - sibling: The pointer to the right sibling of the node. In case of a root node, the sibling points to the root of another tree in the right.



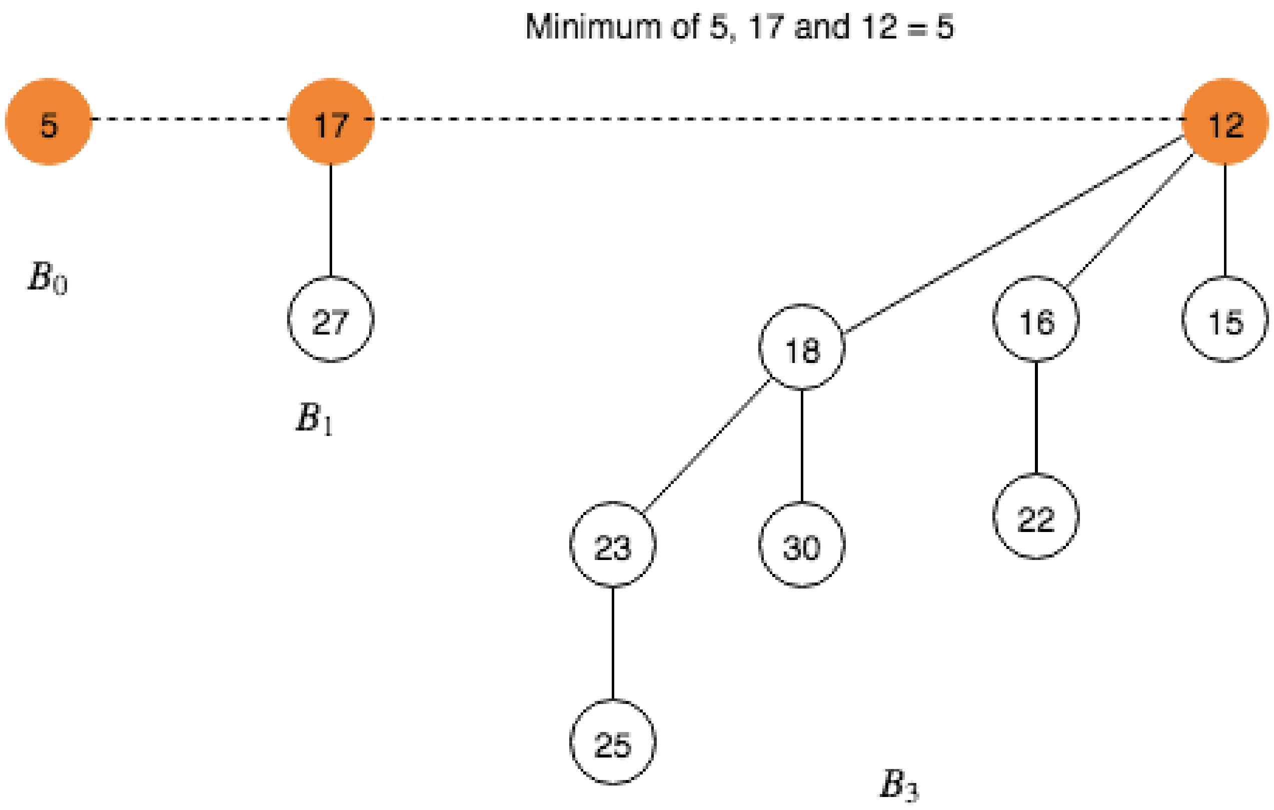
Binomial Heaps





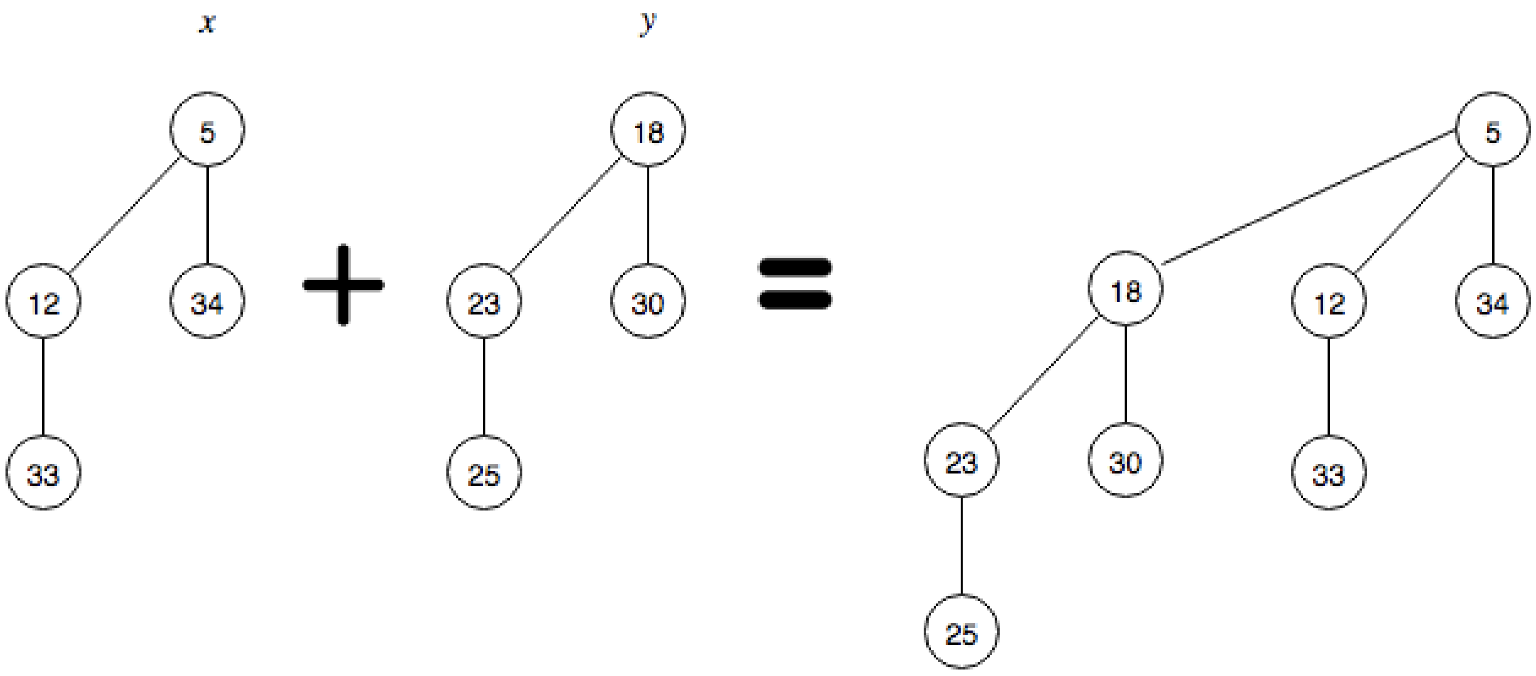
Finding a Minimum

- The root of a min-heap-ordered binomial tree contains the node with minimum data in it. If there are m such trees, we need to find the minimum of m items. If n is a total number of nodes in a binomial heap, there are at most ⌊logn⌋+1 binomial trees. So the running time of this operation is Θ(logn).



Merging

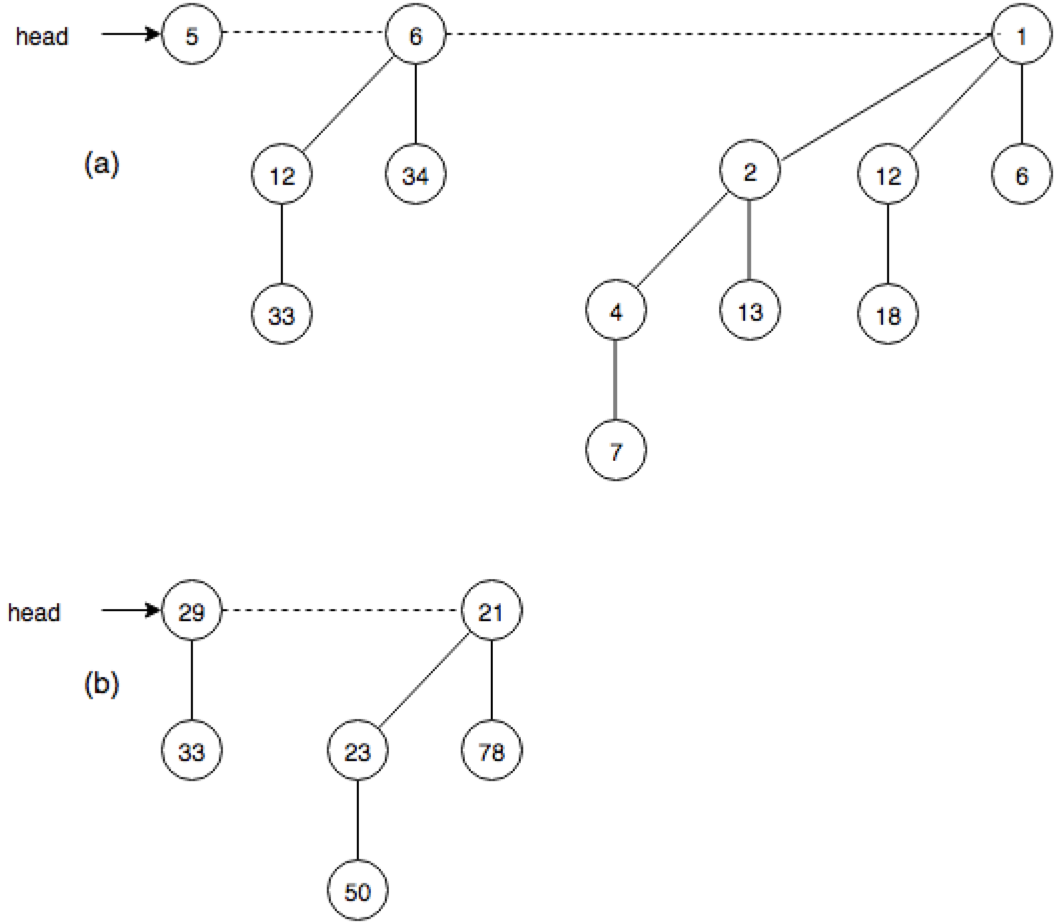
- We repeatedly merge two binomial trees of the same degree until all the binomial trees have a unique degree. To merge two binomial trees of the same degree, we do the following.
  - Compare the roots of two trees (x and y). Find the smallest root. Let x is the tree with the smallest root.
  - Make the x’s root parent of y’s root.



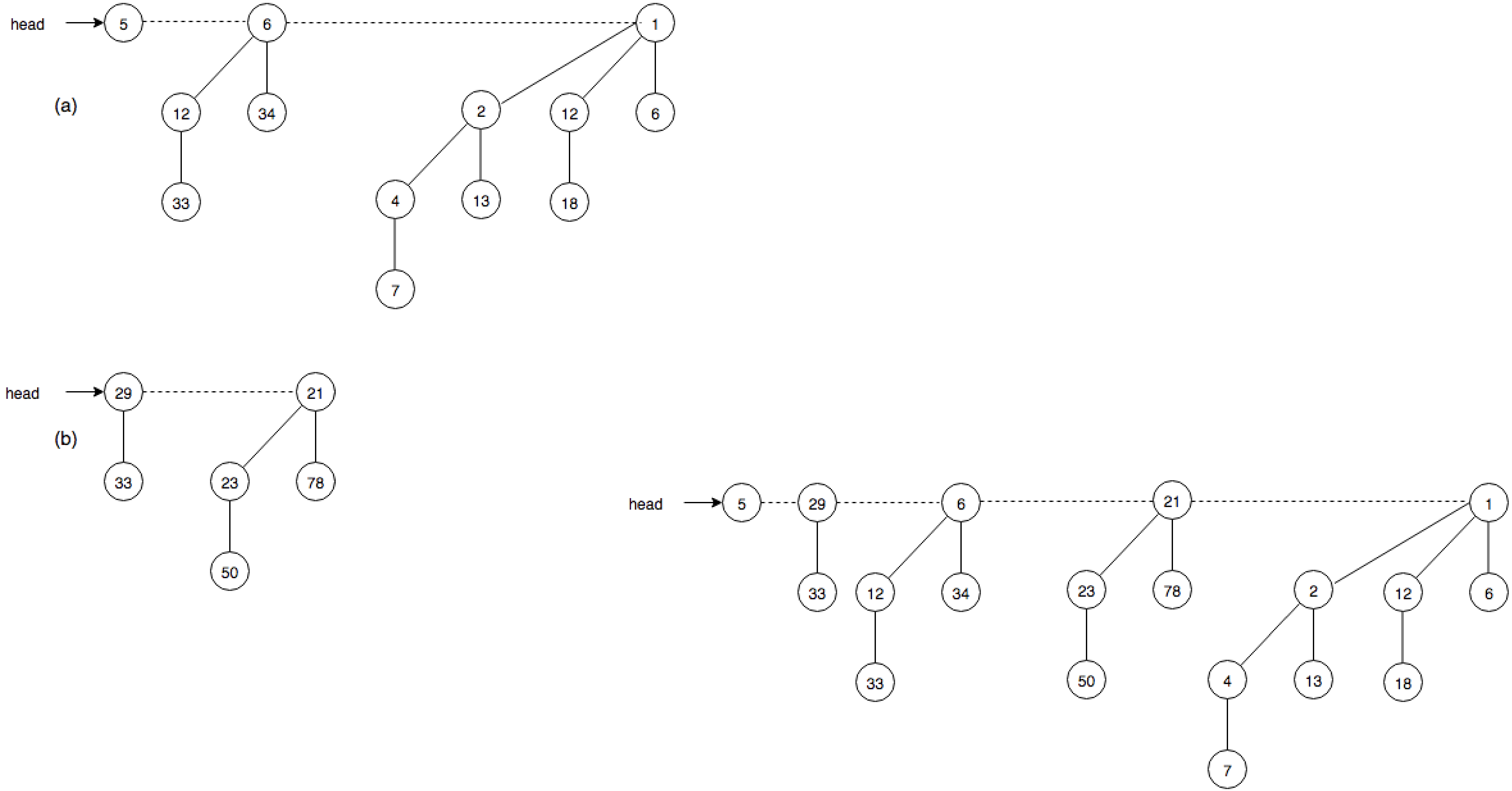
  - This operation clearly takes constant time.

Merging

- Once we know how to merge two binomial trees with the same degree, we can merge two binomial heaps using the following steps.
  - Merge two binomial heaps without worrying about trees with the same degree. That means put the trees in the increasing order of degree.
  - Starting from the head, repeatedly merge trees with the same degree until all the trees in the heap have a unique degree.

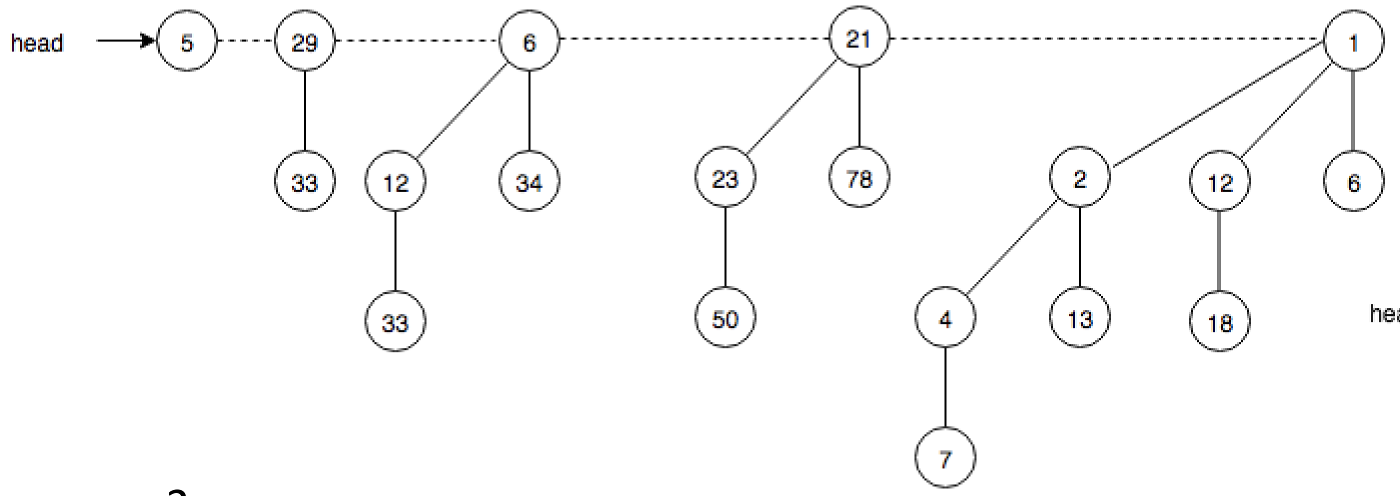


Merging

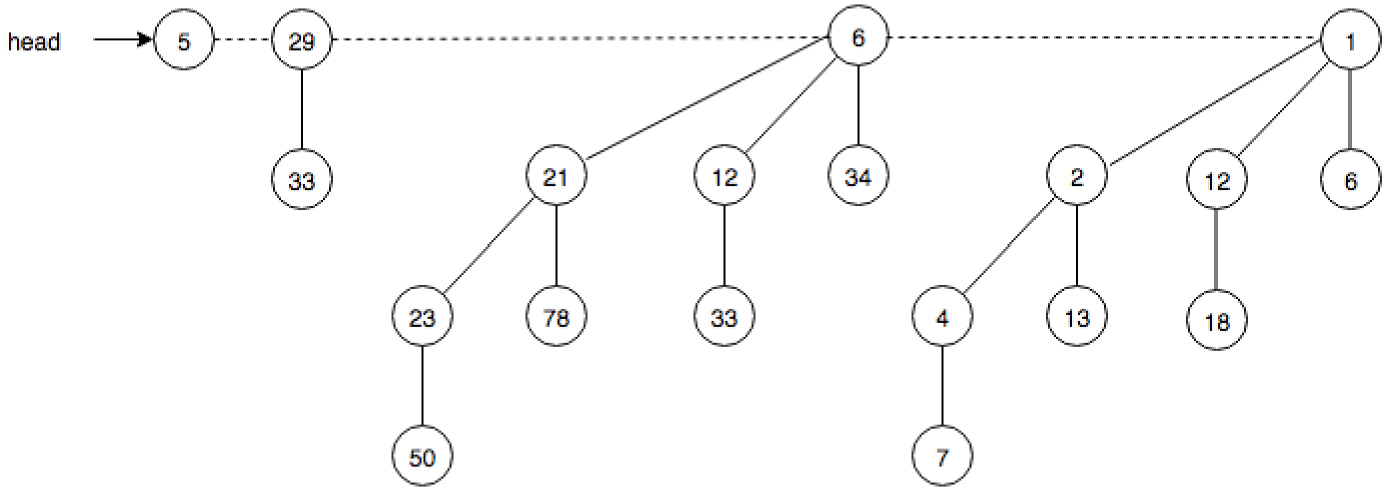


Merging

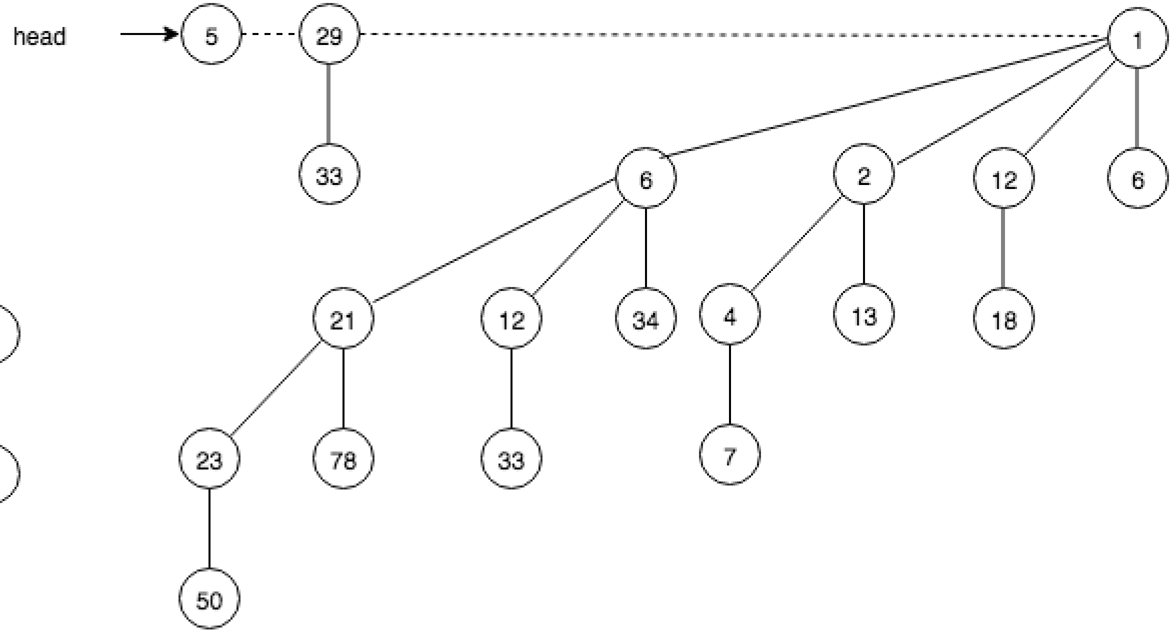
1.


head
5
29
6
21
1
33
12
34
23
78
2
12
6
33
50
4
13
18
7

2.


head
5
29
6
1
33
21
12
34
2
12
6
23
78
33
4
13
18
50
7

3.

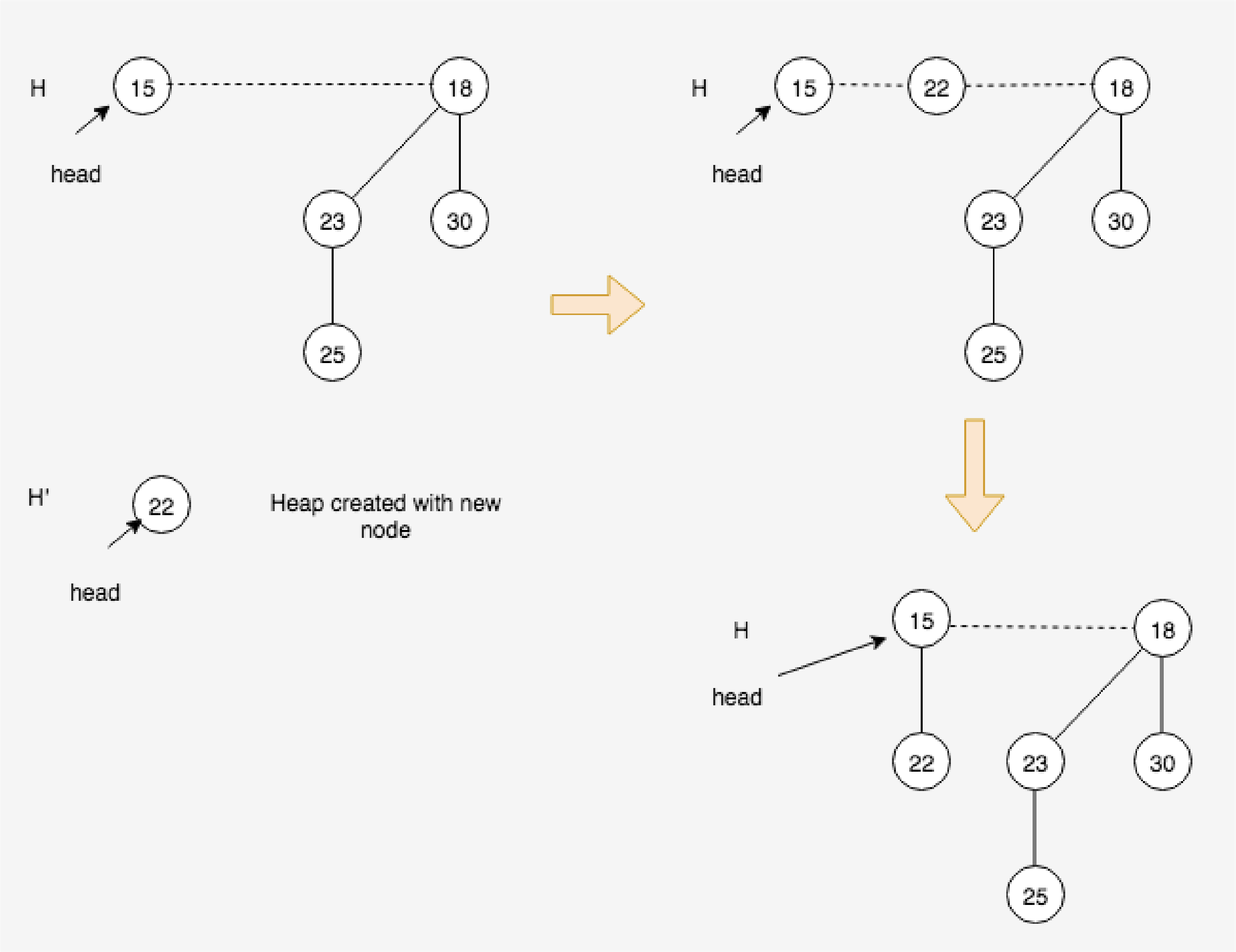

head
5
29
1
33
6
2
12
6
21
12
34
4
13
18
23
78
33
7
50

Merging

- The complexity of merge operation is O(logn). There are at most $\lfloor \log n1 + \log n2 \rfloor + 2$ number of binomial trees (where n1 is the number of nodes in the first heap and n2 is the number of nodes in the second tree). We traverse the roots of these trees constant number of times. That gives the complexity of O(logn).

Insert a Node

- Inserting a node x into a heap H is a three steps process as follows.
  - Create a new empty binomial heap H'. It has a head pointer pointing to null.
  - Create a new node x with all the necessary fields. Point the head of the heap to x.
  - Merge this new heap H' with the existing heap H.



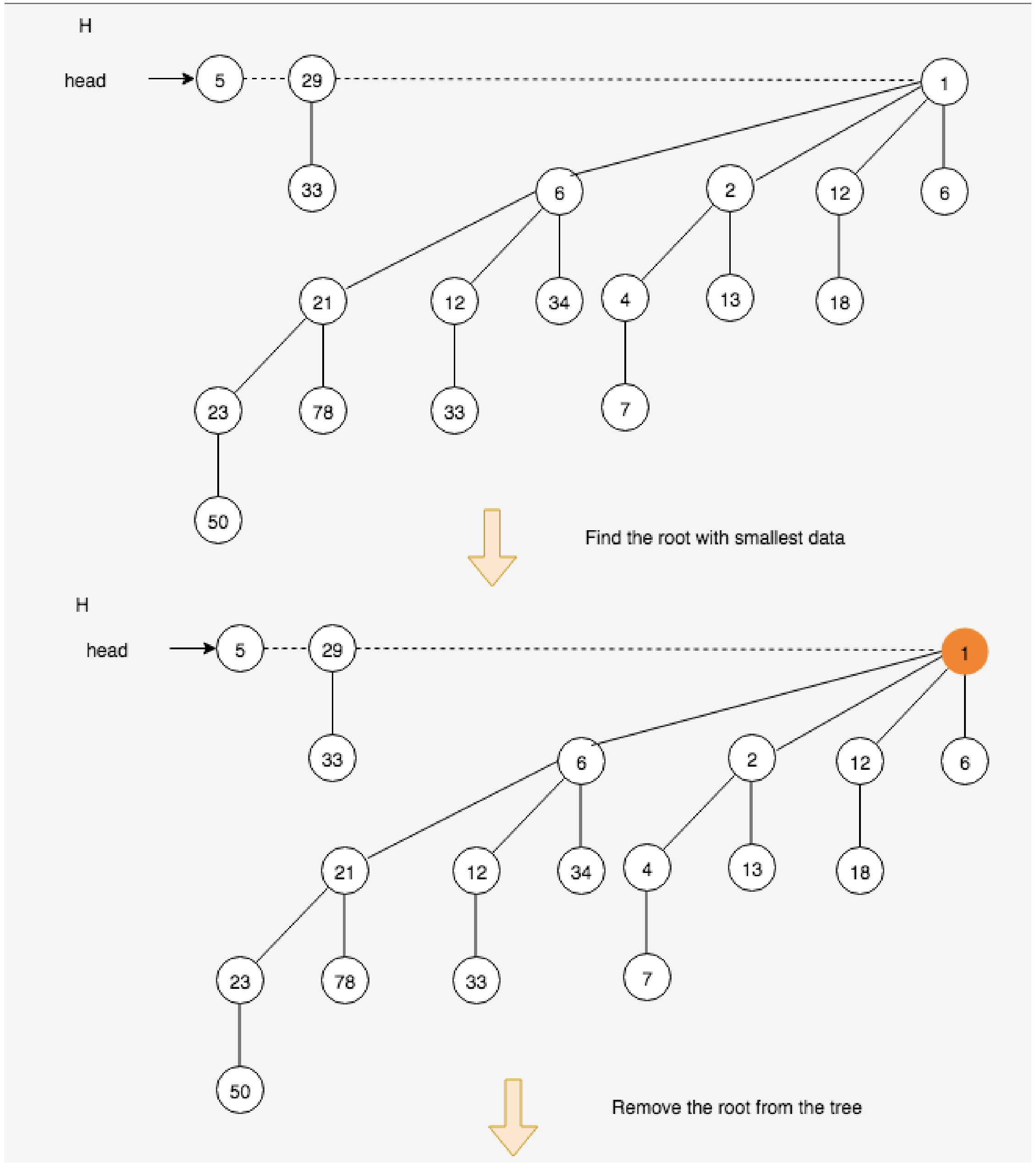
Insert a Node

- The complexity of inserting a new node is O(logn). Creating new heap takes only a constant amount of time and merging two heaps takes O(logn).

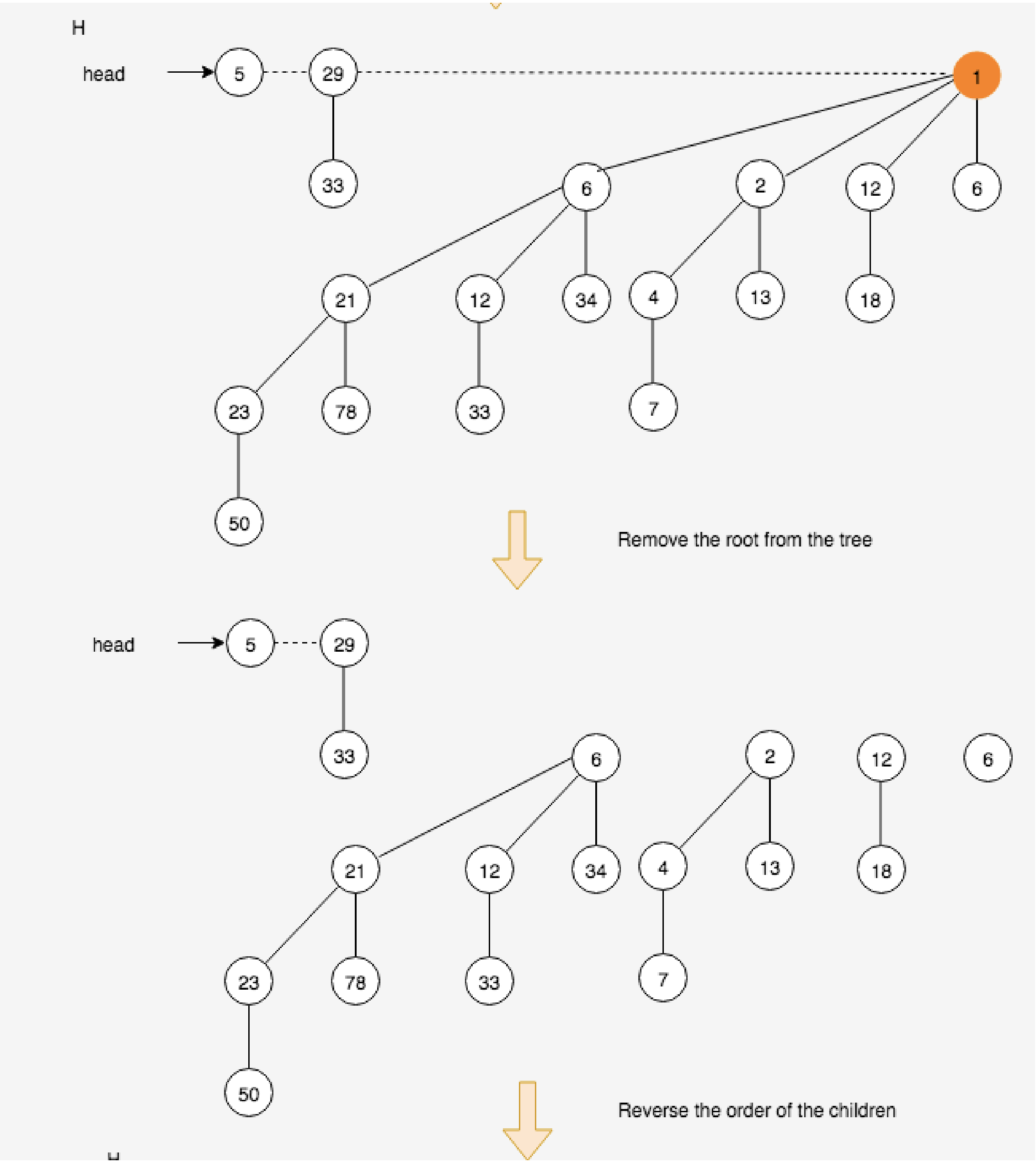
Delete Min

- The steps for deleting the node from a heap H with the smallest key is given below.
  - Search through the roots of binomial trees and find the root with the smallest key and call it x. Remove the x from the tree.
  - Create a new empty heap H’.
  - Reverse the order of x’s children and set the head of H’ to point to the head of the resulting list.
  - Merge H’ with H.

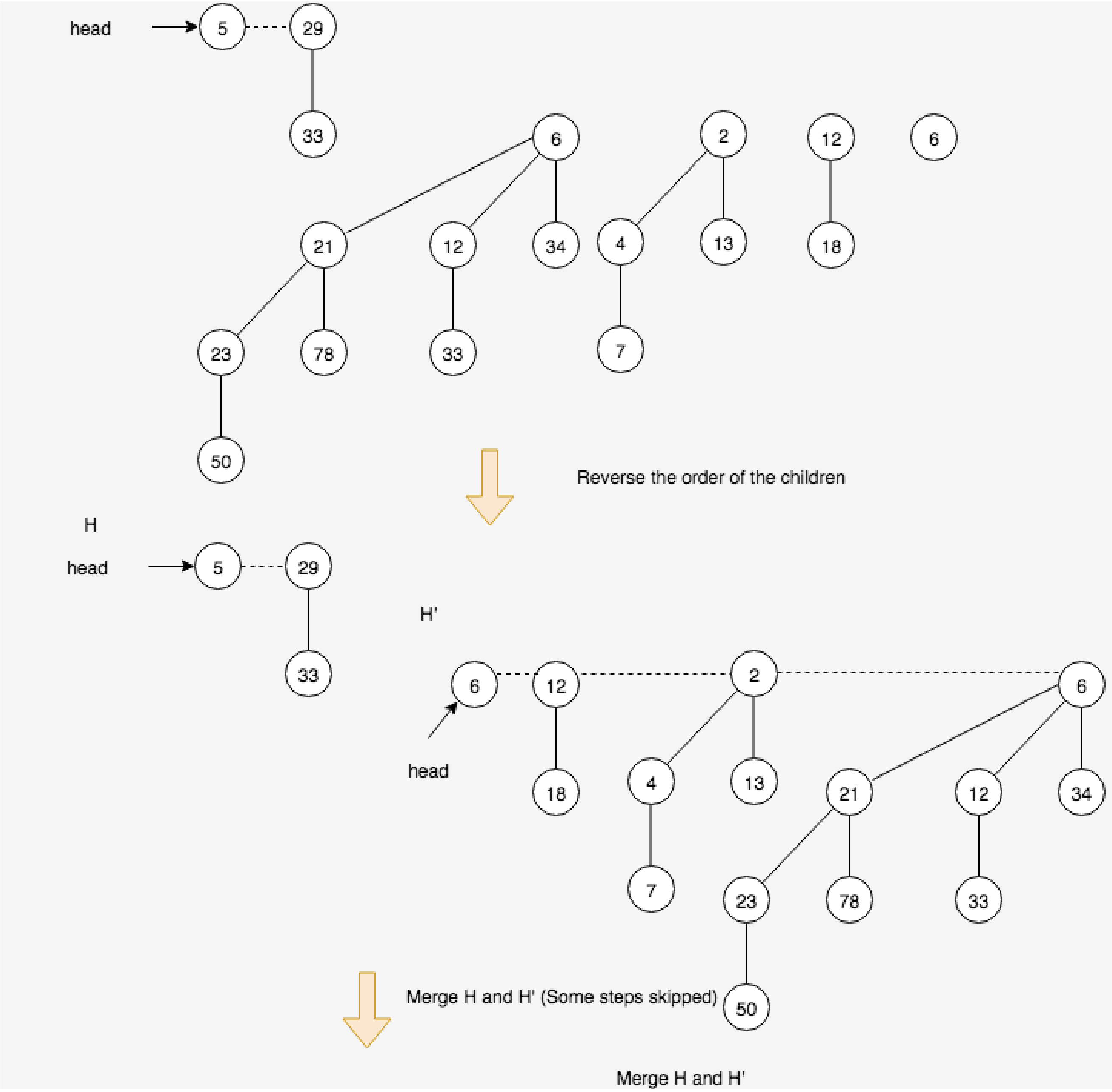
Delete Min


H
head
5
29
1
33
6
2
12
6
21
12
34
4
13
18
23
78
33
7
50
Find the root with smallest data
H
head
5
29
1
33
6
2
12
6
21
12
34
4
13
18
23
78
33
7
50
Remove the root from the tree

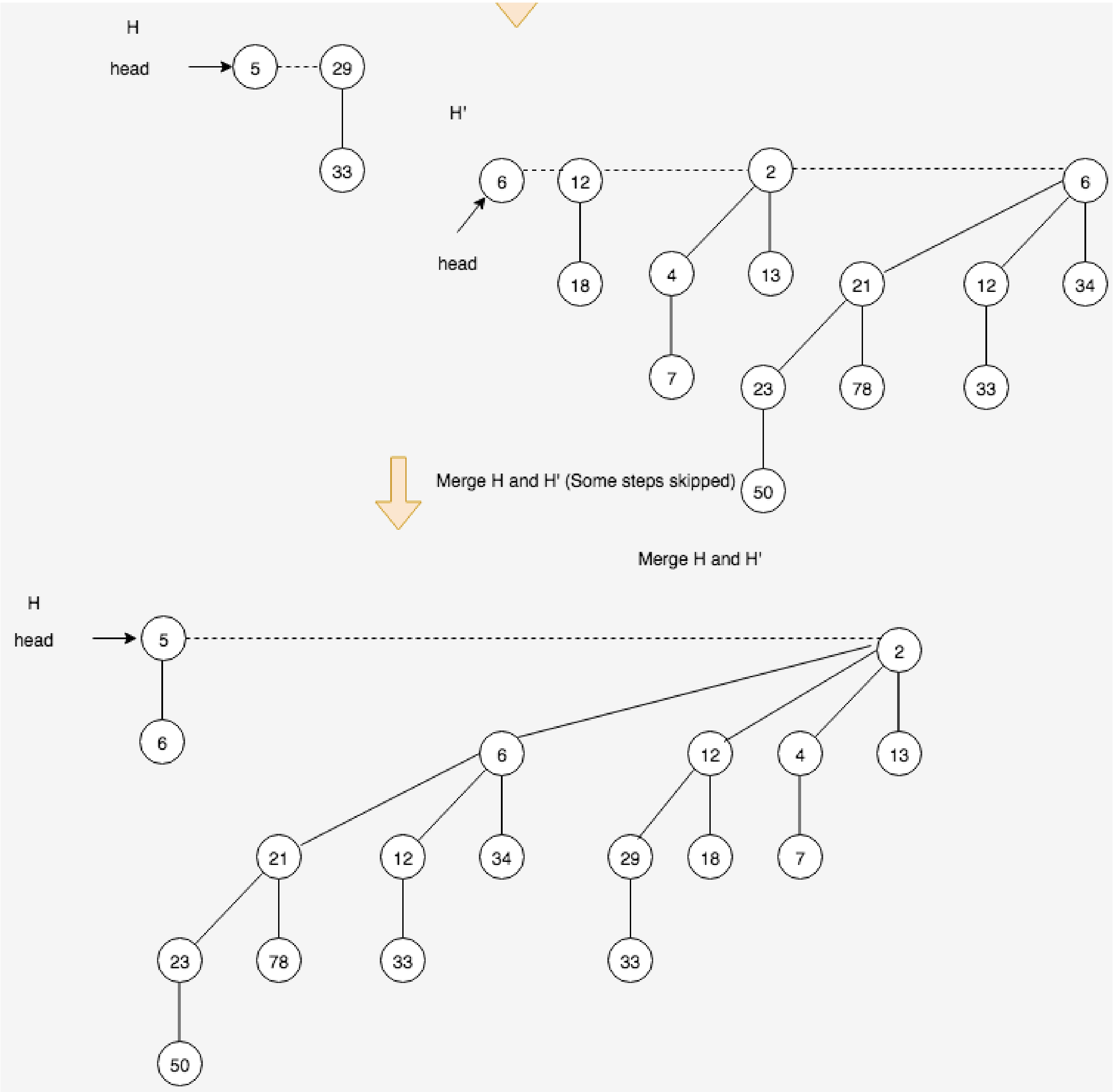
Delete Min


H
head
5
29
1
33
6
2
12
6
21
12
34
4
13
18
23
78
33
7
50
Remove the root from the tree
head
5
29
33
6
2
12
6
21
12
34
4
13
18
23
78
33
7
50
Reverse the order of the children

Delete Min


head
5
29
33
6
21
12
34
23
78
33
50
2
4
13
7
12
18
6
Reverse the order of the children
H
head
5
29
33
H'
6
12
2
6
head
18
4
13
21
12
34
7
23
78
33
50
Merge H and H' (Some steps skipped)
Merge H and H'

Delete Min



H
head
5
29
33
H'
6
12
2
6
head
18
4
13
21
12
34
7
23
78
33
50
Merge H and H' (Some steps skipped)
Merge H and H'
H
head
5
2
6
6
12
4
13
21
12
34
29
18
7
23
78
33
33
50

Delete Min

- The complexity of this operation is O(logn).

Decrease Key

- After decreasing the key of an element, it may become smaller than the key of its parent, violating the minimum-heap property.

Decrease Key

- After decreasing the key of an element, it may become smaller than the key of its parent, violating the minimum-heap property.
- If this is the case, exchange the element with its parent, and possibly also with its grandparent, and so on, until the minimum-heap property is no longer violated.

Decrease Key

- After decreasing the key of an element, it may become smaller than the key of its parent, violating the minimum-heap property.
- If this is the case, exchange the element with its parent, and possibly also with its grandparent, and so on, until the minimum-heap property is no longer violated.
- Each binomial tree has height at most $\log_2 n$, so this takes $O(\log n)$ However, this operation requires that the representation of the tree include pointers from each node to its parent in the tree, somewhat complicating the implementation of other operations.

Delete

- To delete an element from the heap, decrease its key to negative infinity (or equivalently, to some value lower than any element in the heap) and then delete the minimum in the heap.

Disclaimer

- The presentation is not original and its is prepared from various sources for teaching purpose only.