CS4311
Design and Analysis of Algorithms

Lecture 17: Binomial Heap

About this lecture

- Binary heap supports various operations quickly: extract-min, insert, decrease-key
- If we already have two min-heaps, *A* and *B*, there is no efficient way to combine them into a single min-heap
- Introduce Binomial Heap
  - can support efficient union operation

Mergeable Heaps

- Mergeable heap : data structure that supports the following 5 operations:
  - Make-Heap( ) : return an empty heap
  - Insert(H,x,k) : insert an item x with key k into a heap H
  - Find-Min(H) : return item with min key
  - Extract-Min(H) : return and remove
  - Union(H_1, H_2) : merge heaps H_1 and H_2

Mergeable Heaps

- Examples of mergeable heap :

  Binomial Heap (this lecture)

  Fibonacci Heap (next lecture)

- Both heaps also support:
  - Decrease-Key(H,x,k) :
    - assign item x with a smaller key k
  - Delete(H,x) : remove item x

Binary Heap vs Binomial Heap

	Binary Heap	Binomial Heap
Make-Heap	$\Theta(1)$	$\Theta(1)$
Find-Min	$\Theta(1)$	$\Theta(\log n)$
Extract-Min	$\Theta(\log n)$	$\Theta(\log n)$
Insert	$\Theta(\log n)$	$\Theta(\log n)$
Delete	$\Theta(\log n)$	$\Theta(\log n)$
Decrease-Key	$\Theta(\log n)$	$\Theta(\log n)$
Union	$\Theta(n)$	$\Theta(\log n)$

Binomial Heap

- Unlike binary heap which consists of a single tree, a binomial heap consists of a small set of component trees
  - no need to rebuild everything when union is perform
- Each component tree is in a special format, called a binomial tree

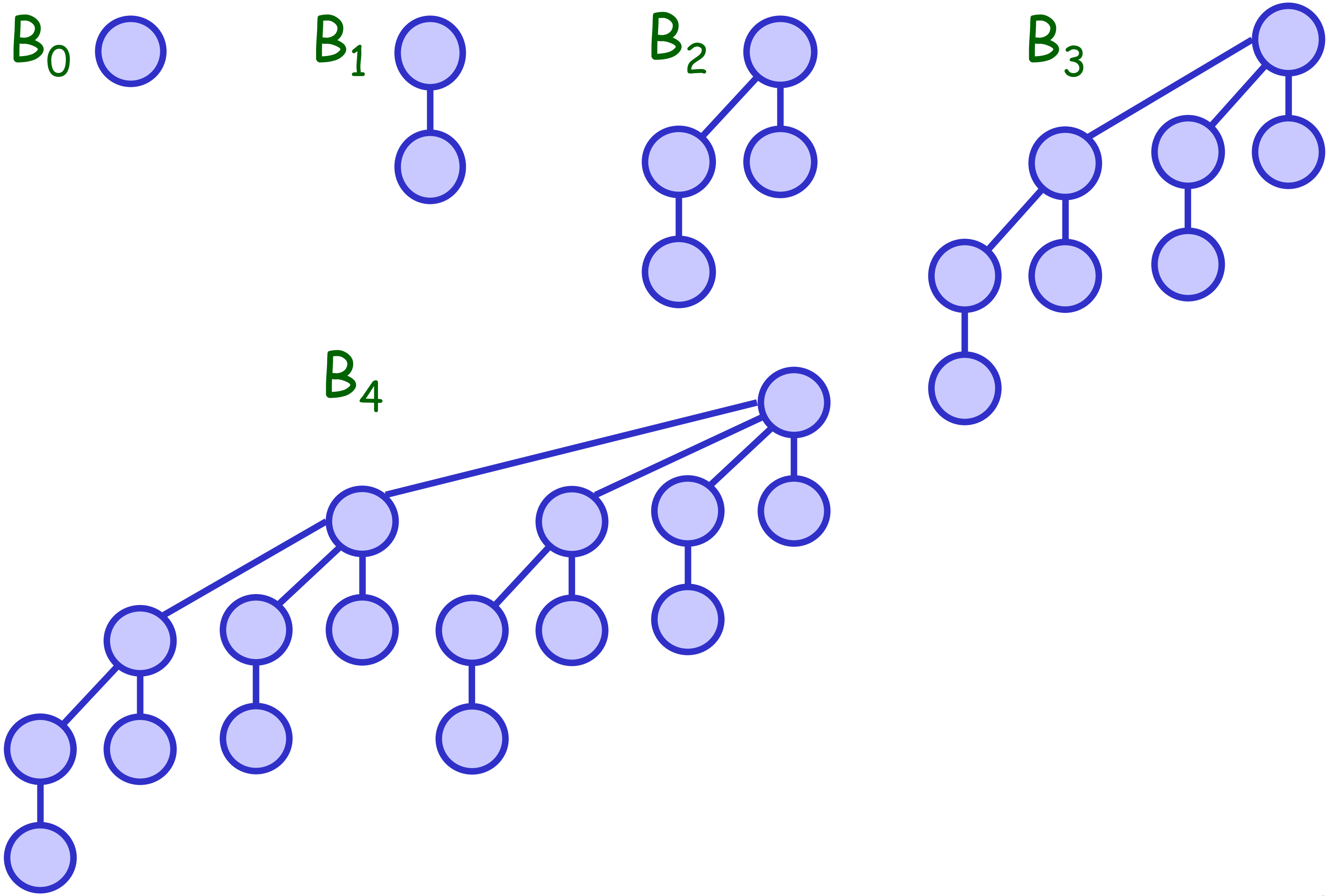
Binomial Tree

Definition:

A binomial tree of order k, denoted by B_k, is defined recursively as follows:

- B_0 is a tree with a single node
- For $k \geq 1$, B_k is formed by joining two B_{k-1}, such that the root of one tree becomes the leftmost child of the root of the other

Binomial Tree



Properties of Binomial Tree

Lemma: For a binomial tree B_k,

1. There are 2^k nodes
2. height = k
3. deg(root) = k ; deg(other node) < k
4. Children of root, from left to right, are B_{k-1}, B_{k-2}, …, B_1, B_0
5. Exactly $C(k,i)$ nodes at depth I

How to prove? (By induction on k)

Binomial Heap

- Binomial heap of n elements consists of a specific set of binomial trees

  - Each binomial tree satisfies min-heap ordering: for each node x,

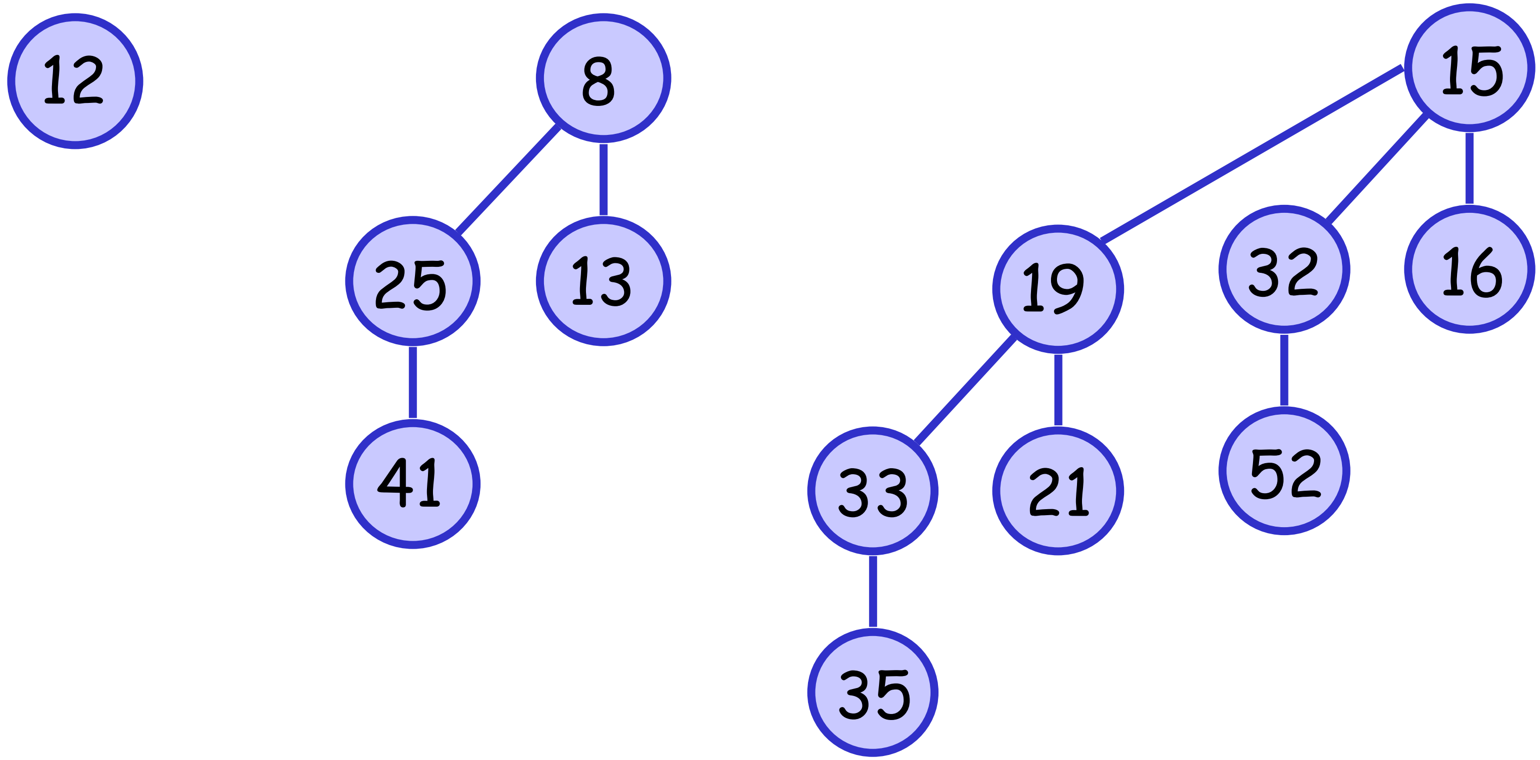
    $$key(x) \geq key(parent(x))$$

  - For each k, at most one binomial tree whose root has degree k

    (i.e., for each k, at most one B_k)

Binomial Heap

Example: A binomial heap with 13 elements



Binomial Heap

- Let $r = \lceil \log (n+1) \rceil$, and

  $$\langle b_{r-1}, b_{r-2}, \ldots, b_2, b_1, b_0 \rangle$$

  be binary representation of n

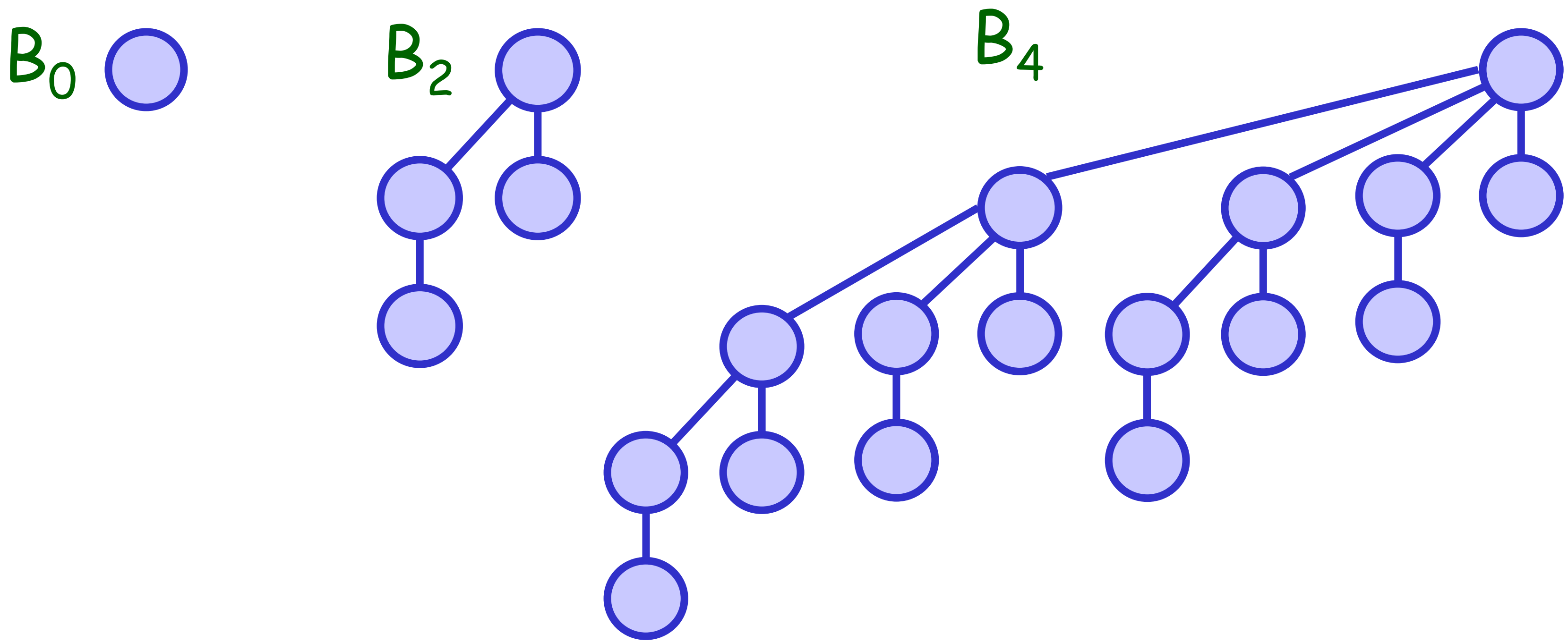
- Then, we can see that an n-node binomial heap contains B_k if and only if $b_k = 1$

- Also, an n-node binomial heap has at most $\lceil \log (n+1) \rceil$ binomial trees

Binomial Heap

E.g., $21_{(dec)} = 10101_{(bin)}$

➔ any 21-node binomial heap must contain:



Binomial Heap Operations

- With the binomial heap,
  - Make-Heap( ): O(1) time
  - Find-Min( ): O(log n) time
  - Decrease-Key( ): O(log n) time

[Decrease-Key assumes we have the pointer to the item x in which its key is changed]

- Remaining operations : Based on Union( )

Union Operation

- Recall that:

an n-node binomial heap

corresponds to

binary representation of n

- We shall see:

Union binomial heaps with n_1 and n_2 nodes

corresponds to

adding n_1 and n_2 in binary representations

Union Operation

- Let H_1 and H_2 be two binomial heaps
- To Union them, we process all binomial trees in the two heaps with same order together, starting with smaller order first
- Let k be the order of the set of binomial trees we currently process

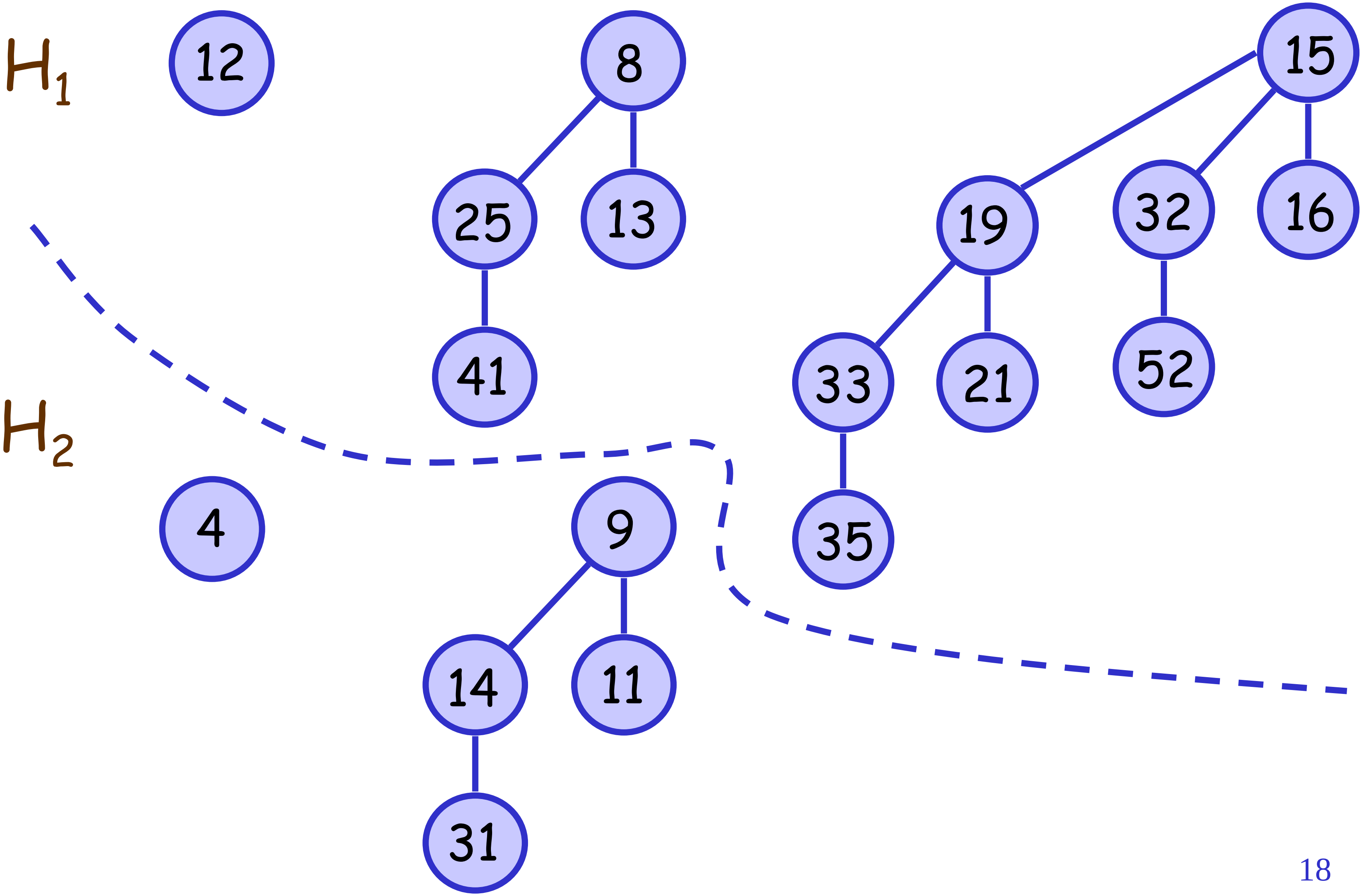
Union Operation

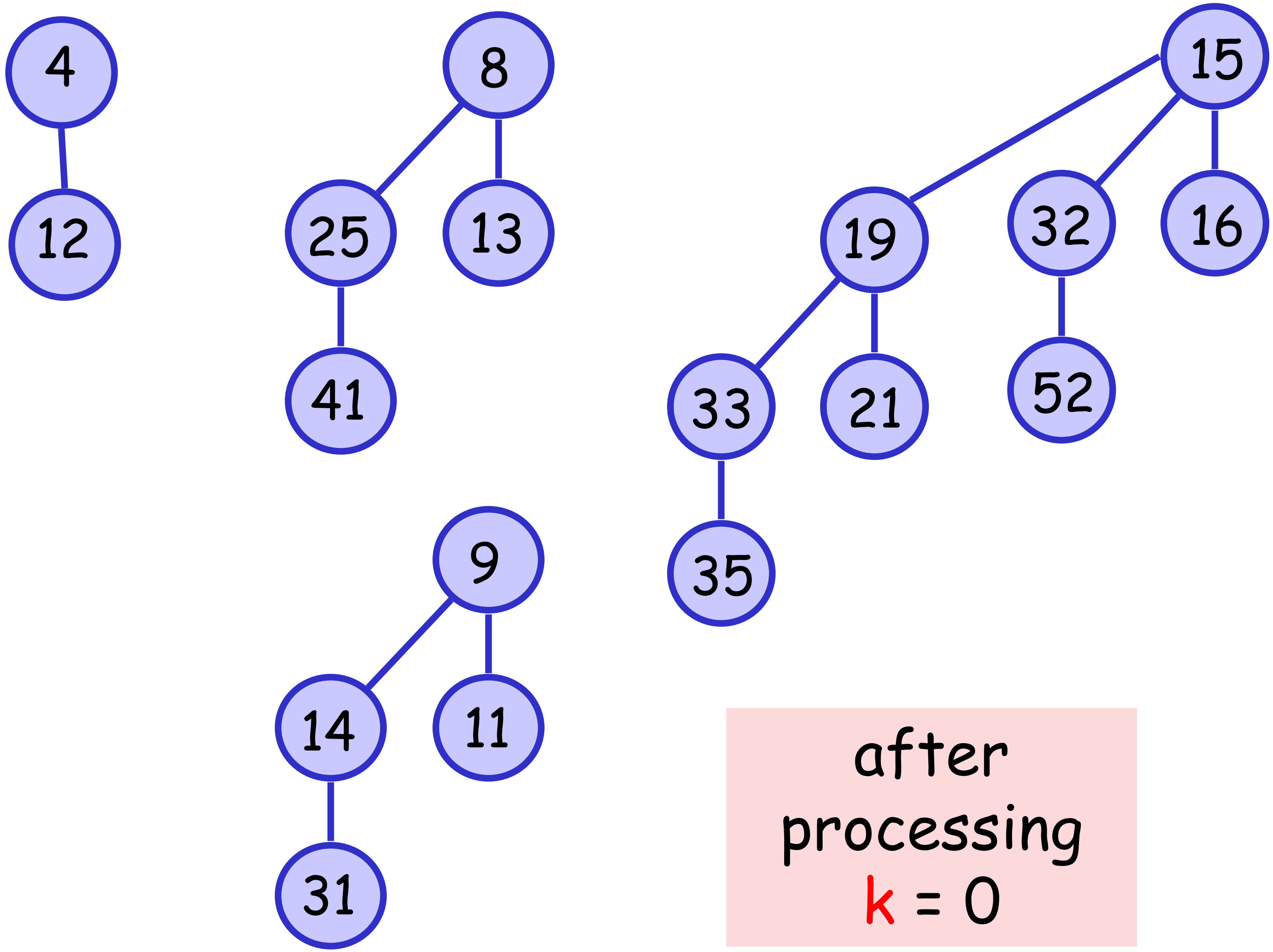
There are three cases:

1. If there is only one B_k → done
2. If there are two B_k
   → Merge together, forming B_{k+1}
3. If there are three B_k
   → Leave one, merge remaining to B_{k+1}

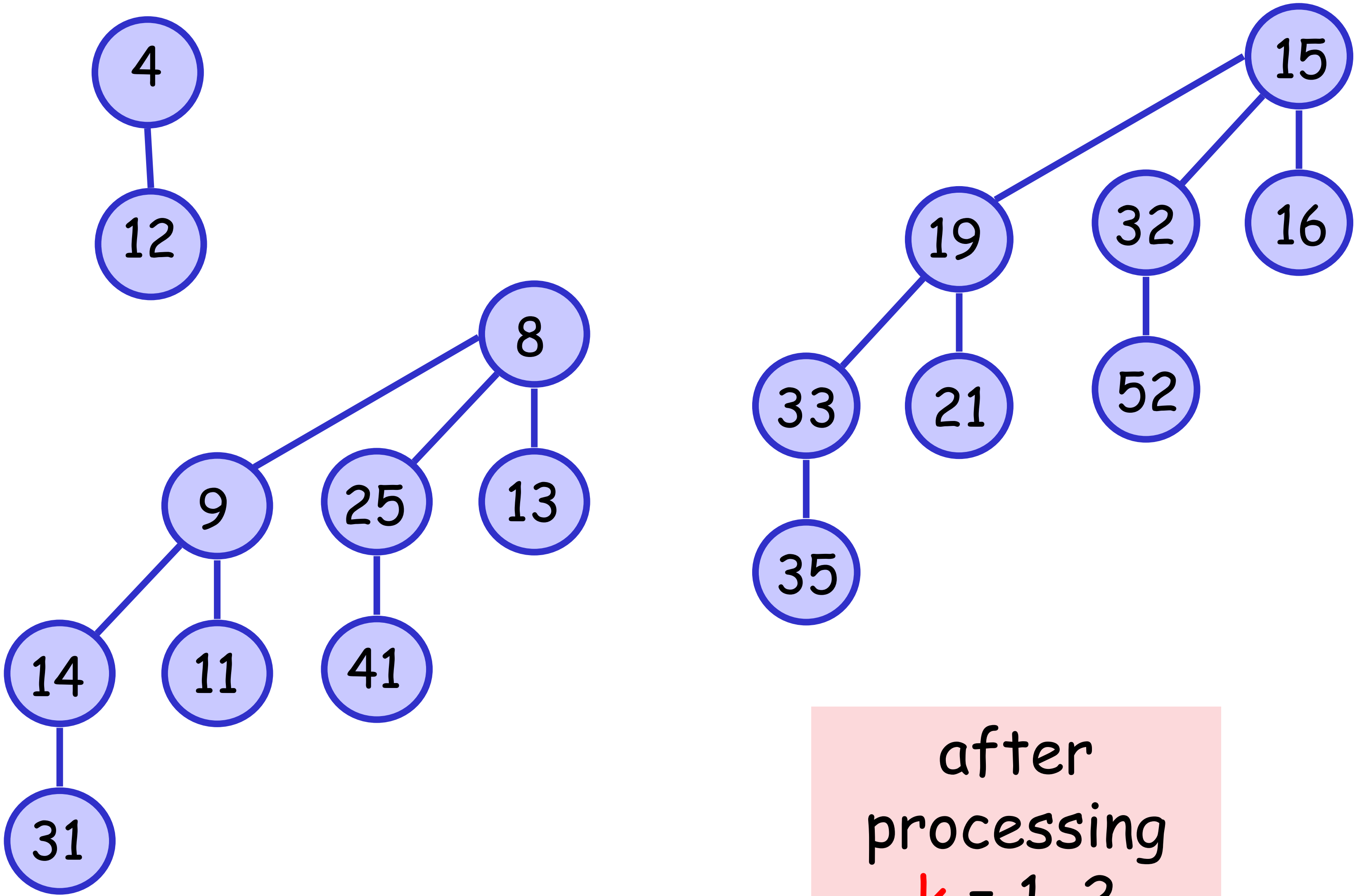
After that, process next k

Union two binomial heaps with 5 and 13 nodes

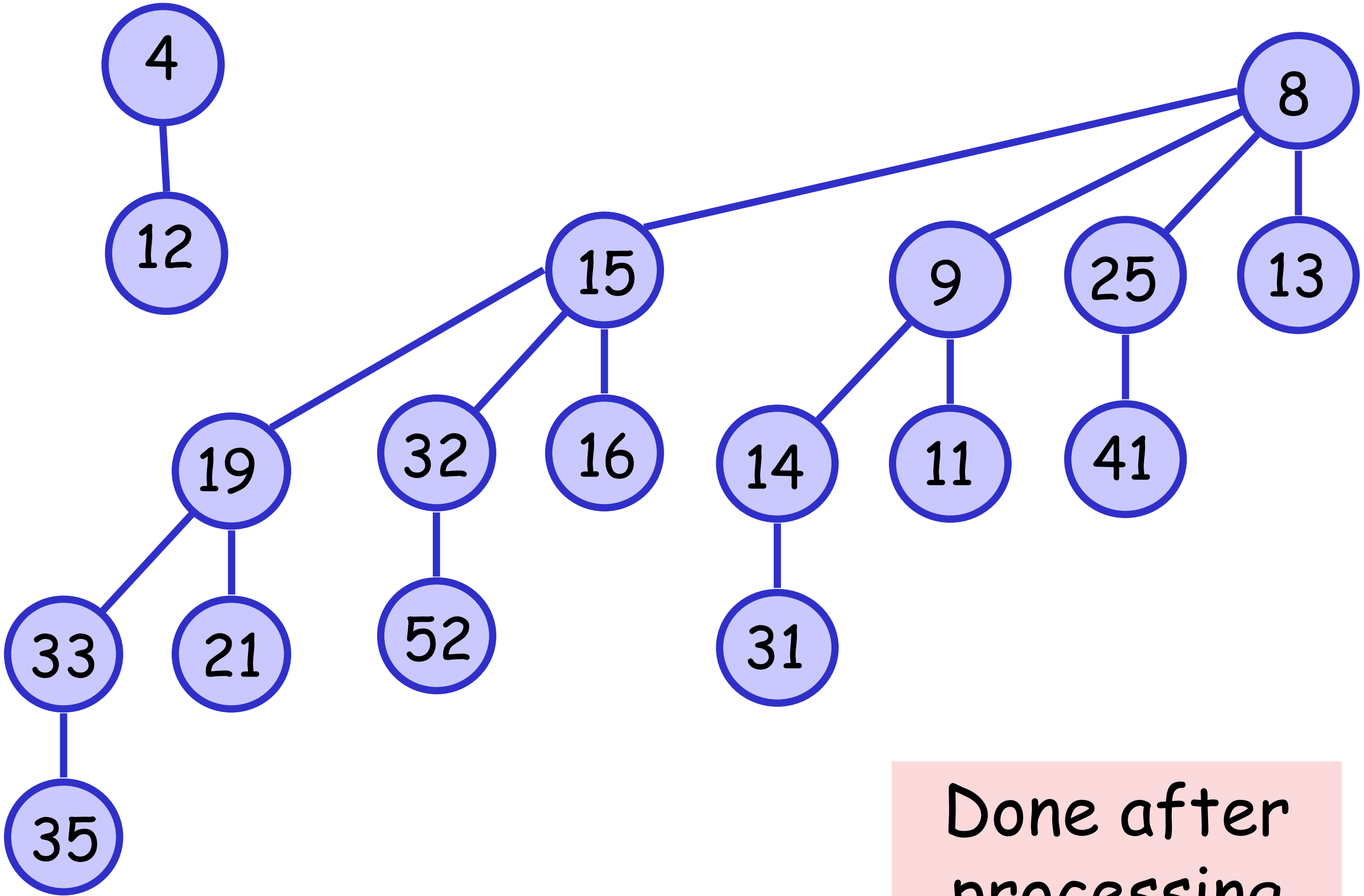




after
processing
k = 0



after processing
k = 1, 2



Done after processing
k = 3

Binomial Heap Operations

- So, Union( ) takes O(log n) time
- For remaining operations,

    Insert( ), Extract-Min( ), Delete( )

  how can they be done with Union?

- Insert(H, x, k):

➔ Create new heap H', storing the item x with key k; then, Union(H, H')

Binomial Heap Operations

- Extract-Min(H):

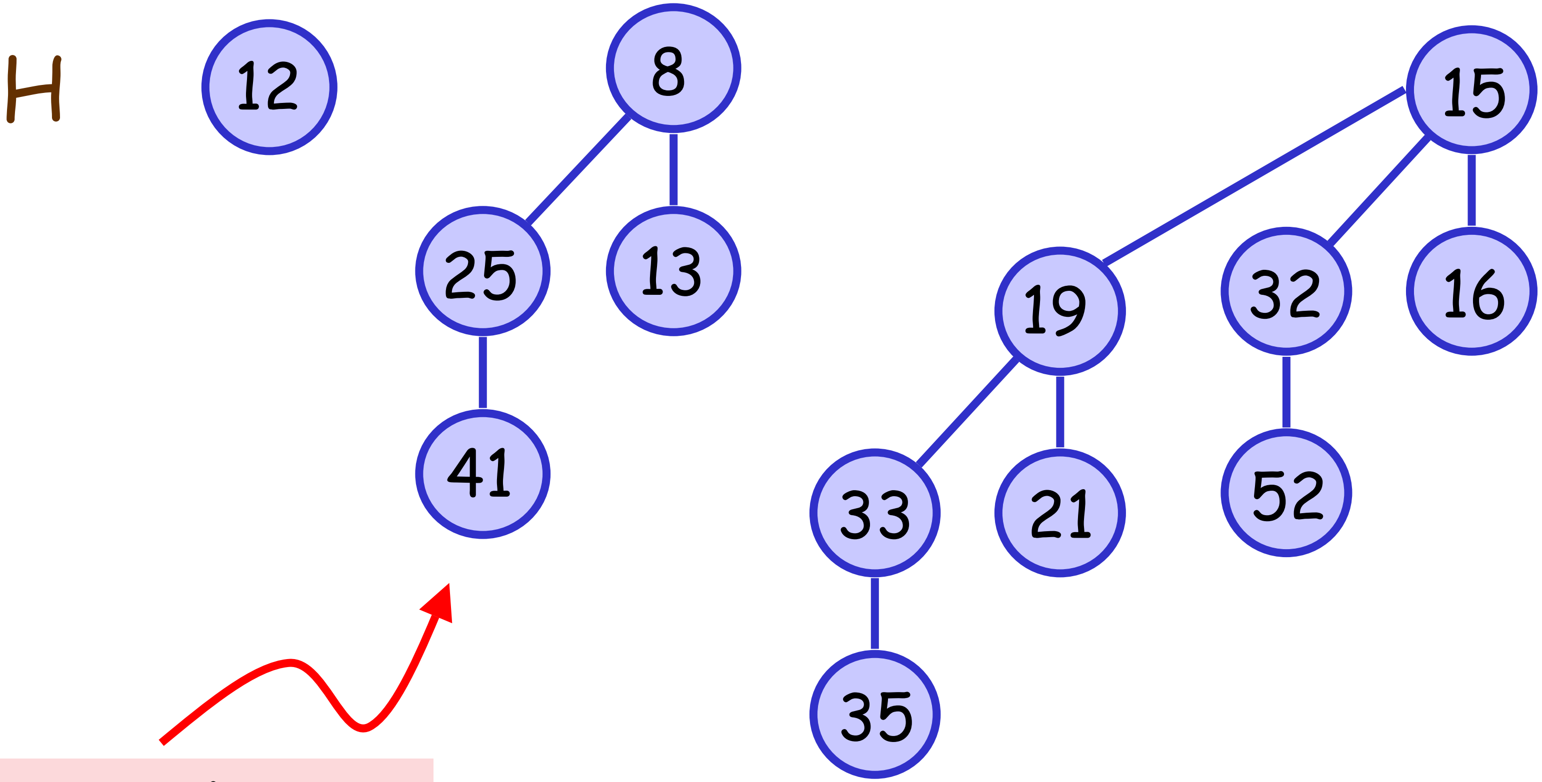
➔ Find the tree B_j containing the min;
Detach B_j from H → forming a heap H_1 ;
Remove root of B_j → forming a heap H_2 ;
Finally, Union(H, H')

- Delete(H, x):

➔ Decrease-Key(H,x,$-\infty$); Extract-Min(H);

Extract-Min(H)

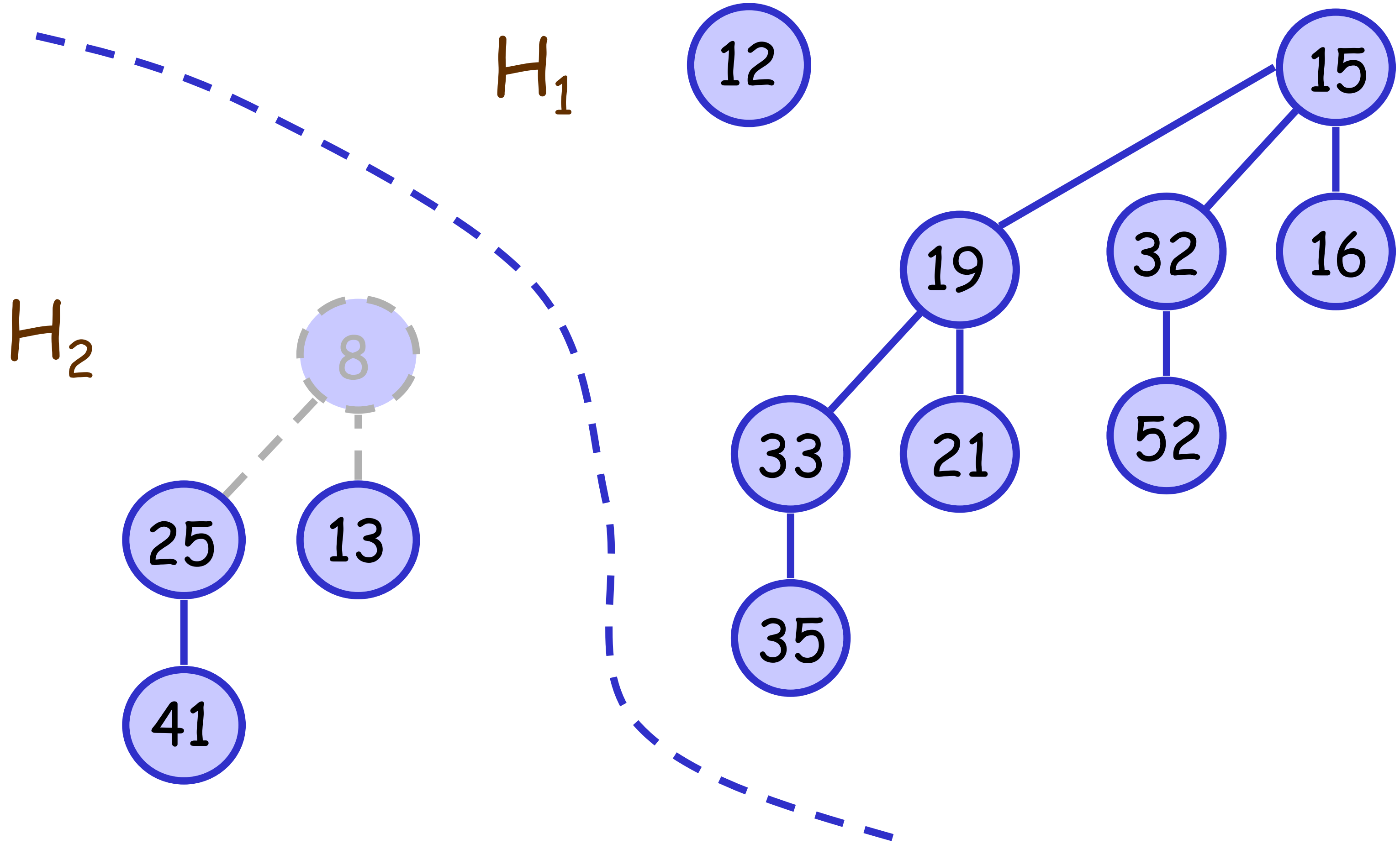
Step 1: Find B_j with Min



B_j with Min

Extract-Min(H)

Step 2: Forming two heaps



Extract-Min(H)
Step 3: Union two heaps