Binomial Heaps

Last Week...

- Priority Queue Abstract Data Type

Last Week…

- Priority Queue Abstract Data Type

  Object: Set S of elements with “keys” (“priority”) that can be compared

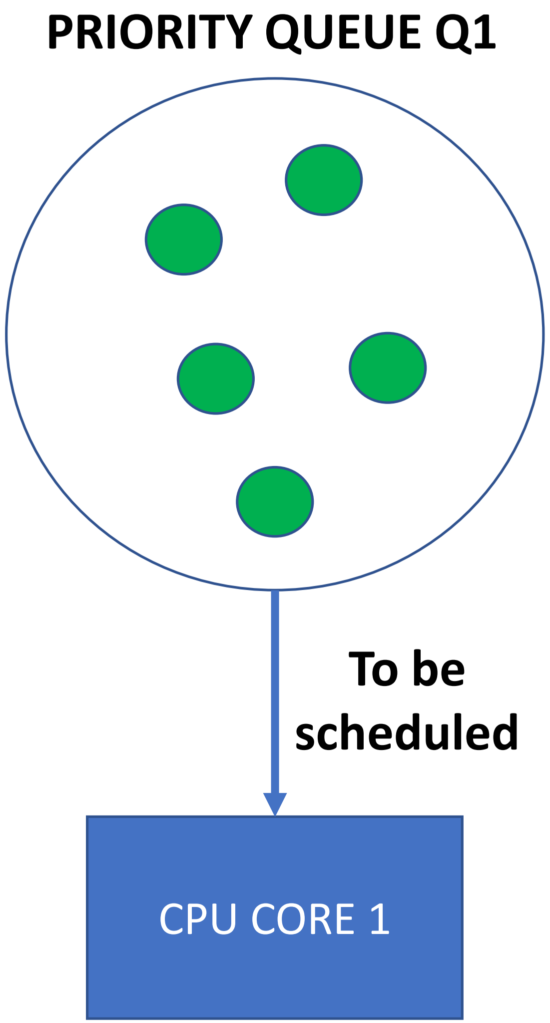
Last Week…

- Priority Queue Abstract Data Type

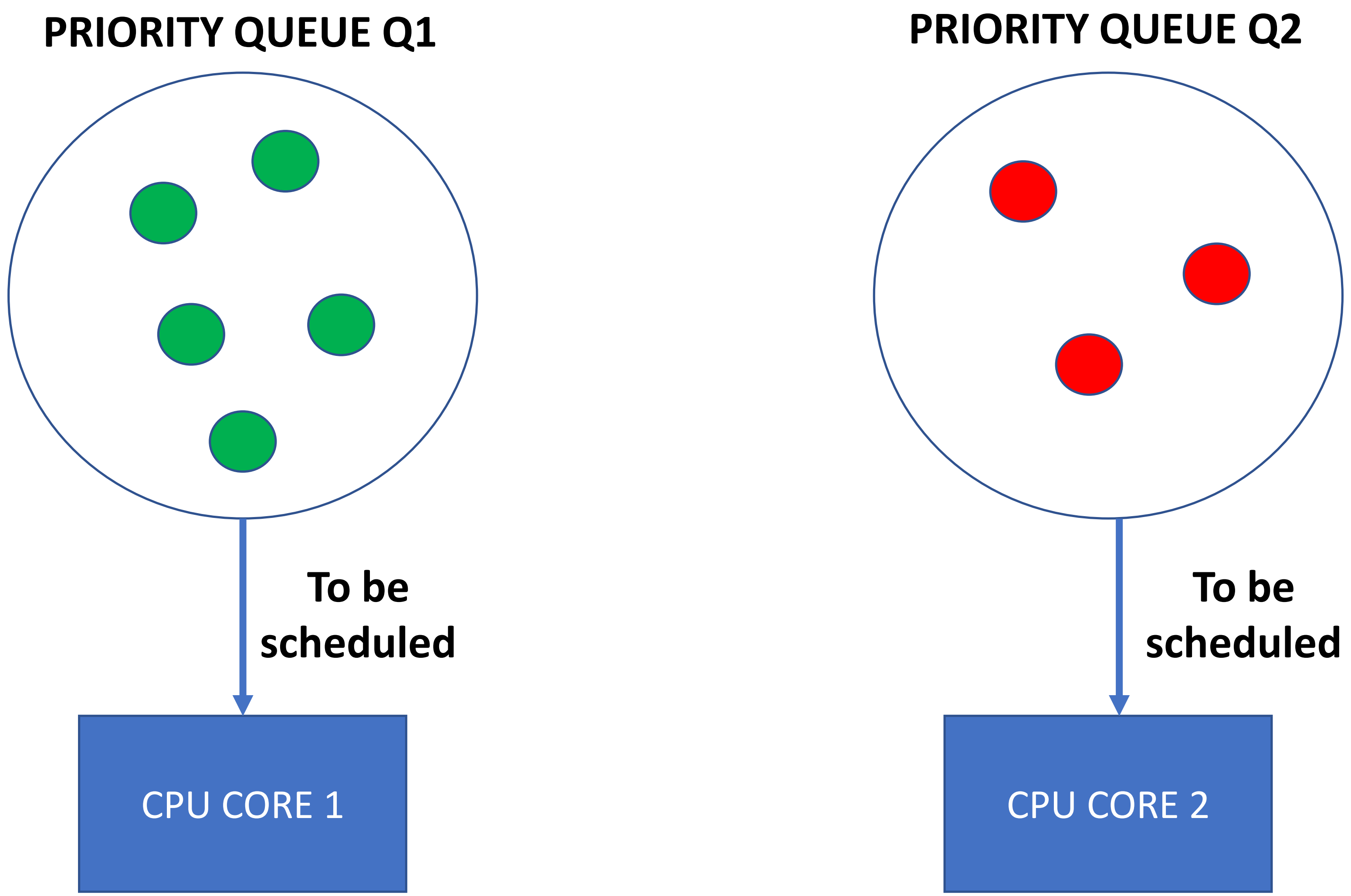
  Object: Set S of elements with “keys” (“priority”) that can be compared

  Operations: **Insert**(S,x), **Max**(S), **Extract_Max**(S)

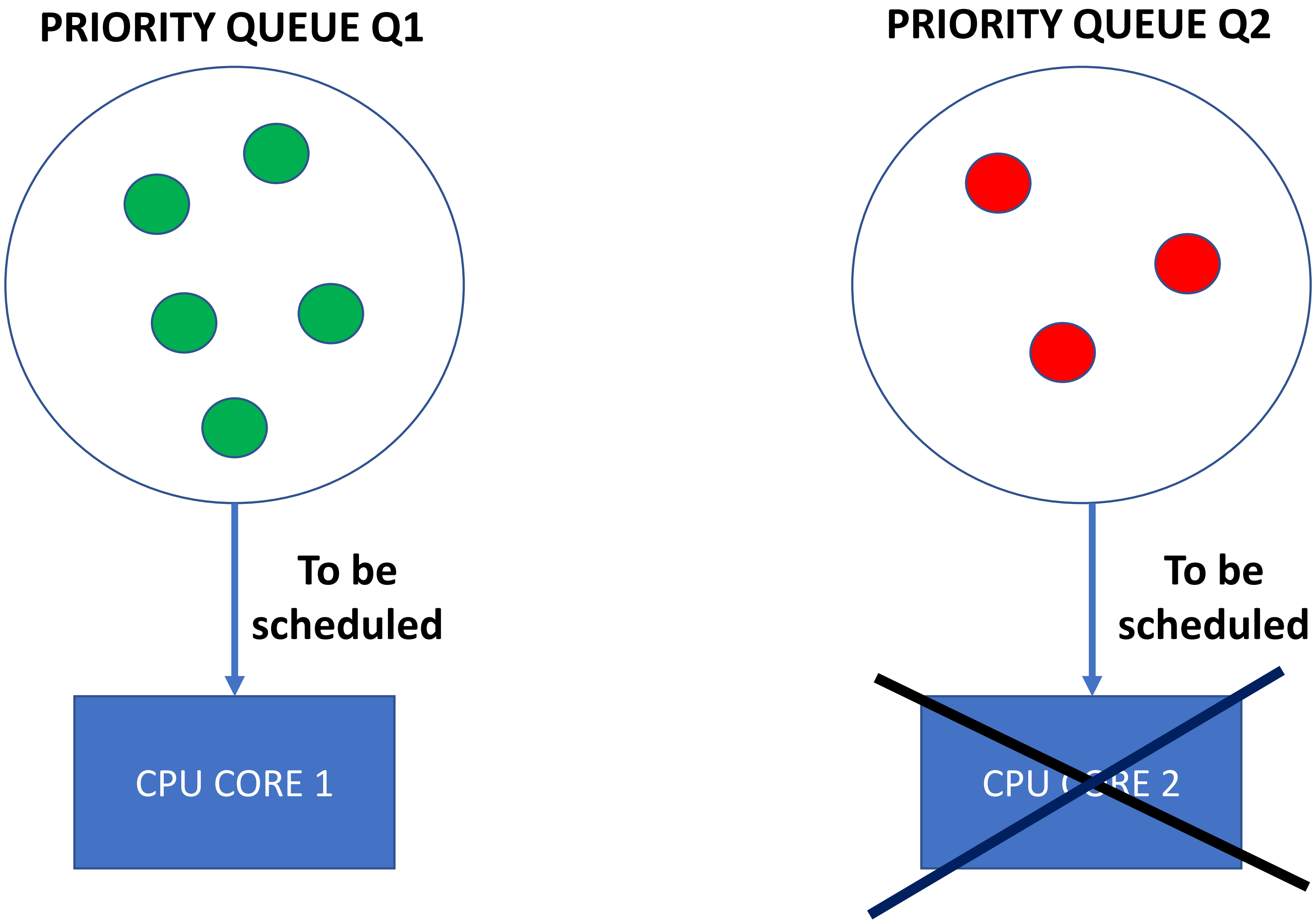
Application: Job Scheduling by OS



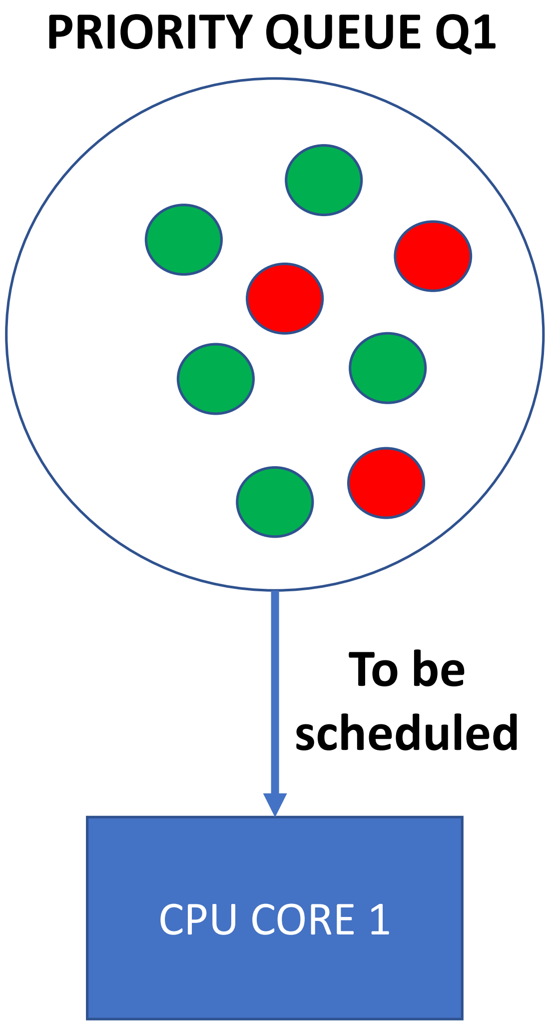
Consider the following scenario



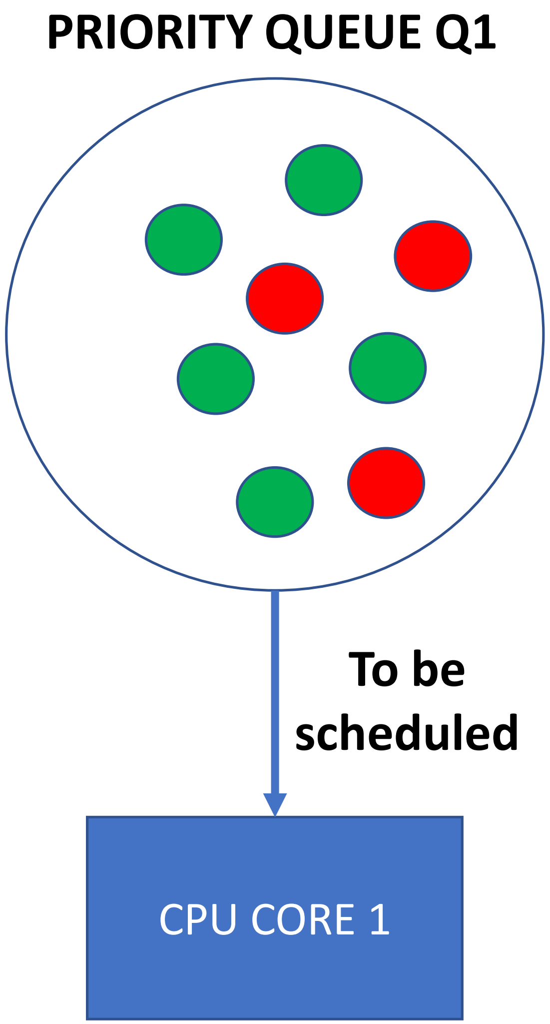
Consider the following scenario



Consider the following scenario



Consider the following scenario



Goal: Implement **Union**(Q1, Q2)

Abstract Data Types

Abstract Data Types		Insert	Min	Extract_Min	Union
Priority Queues		✓	✓	✓	X

Abstract Data Types

Abstract Data Types		Insert	Min	Extract_Min	Union
Priority Queues		✓	✓	✓	X
Mergeable Priority Queues					

Abstract Data Types

Abstract Data Types		Insert	Min	Extract_Min	Union
Priority Queues		✓	✓	✓	X
Mergeable Priority Queues		✓	✓	✓	✓

Data Structures

Abstract Data Types	Data Structures	Insert	Min	Extract_Min	Union
Priority Queues	Min-Heap	✓	✓	✓	X
Mergeable Priority Queues		✓	✓	✓	✓

Data Structures

Abstract Data Types	Data Structures	Insert	Min	Extract_Min	Union
Priority Queues	Min-Heap	Θ(log n)	✓	✓	X
Mergeable Priority Queues		✓	✓	✓	✓

Data Structures

Abstract Data Types	Data Structures	Insert	Min	Extract_Min	Union
Priority Queues	Min-Heap	$\Theta(\log n)$	$\Theta(1)$	✓	X
Mergeable Priority Queues		✓	✓	✓	✓

Data Structures

Abstract Data Types	Data Structures	Insert	Min	Extract_Min	Union
Priority Queues	Min-Heap	Θ(log n)	Θ(1)	Θ(log n)	X
Mergeable Priority Queues		✓	✓	✓	✓

Data Structures

Abstract Data Types	Data Structures	Insert	Min	Extract_Min	Union
Priority Queues	Min-Heap	$\Theta(\log n)$	$\Theta(1)$	$\Theta(\log n)$	X
Mergeable Priority Queues	Min Binomial Heap	✓	✓	✓	✓

Data Structures

Abstract Data Types	Data Structures	Insert	Min	Extract_Min	Union
Priority Queues	Min-Heap	$\Theta(\log n)$	$\Theta(1)$	$\Theta(\log n)$	X
Mergeable Priority Queues	Min Binomial Heap	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$

Min Binomial Heaps

Visualizing Binomial Heaps

Elements are stored in a sequence of Binomial Trees.

Binomial Trees

B_k tree: defined recursively

Binomial Trees

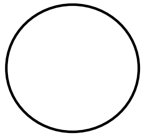
B_k tree: defined recursively

k = 0

Binomial Trees

B_k tree: defined recursively

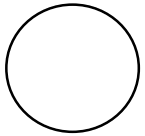
k = 0 B_0 :



Binomial Trees

B_k tree: defined recursively

$k = 0$ B_0 :

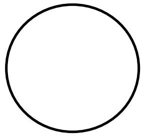


$k > = 1$

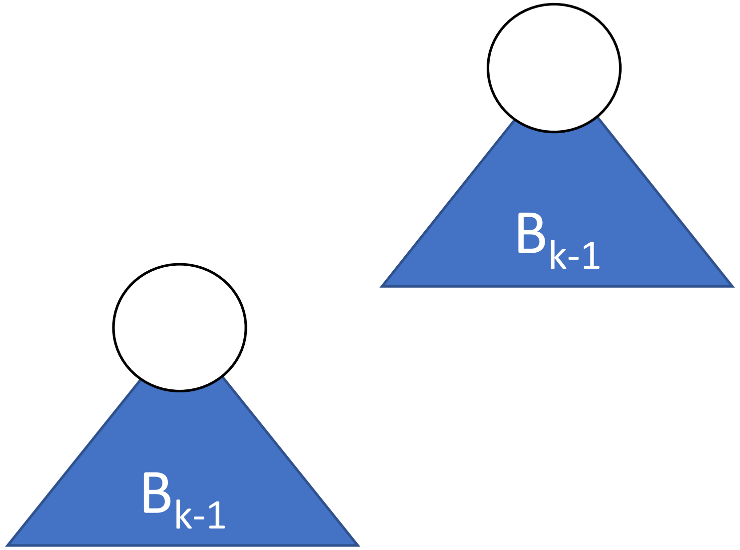
Binomial Trees

B_k tree: defined recursively

k = 0 B_0 :



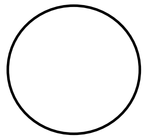
k > = 1



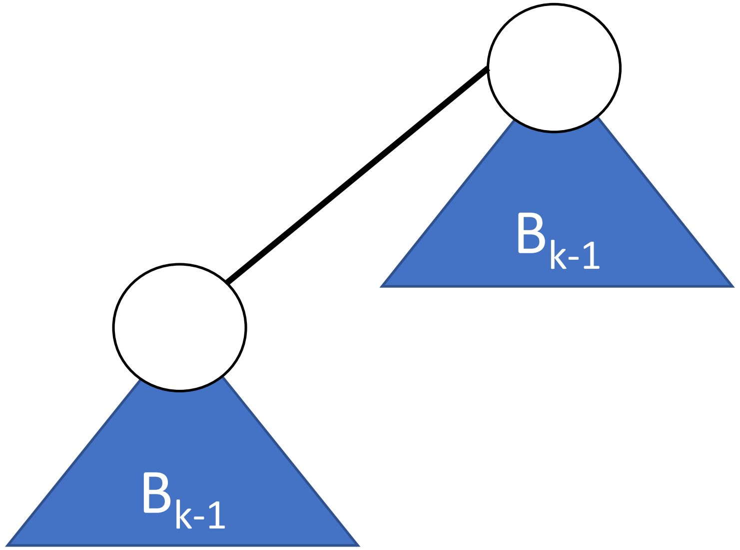
Binomial Trees

B_k tree: defined recursively

k = 0 B_0 :



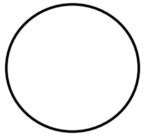
k > = 1 B_k :



Binomial Trees (for k = 0, 1, 2, 3)

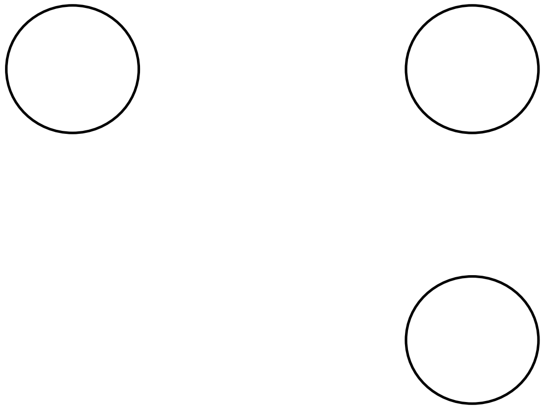
Binomial Trees (for k = 0, 1, 2, 3)

B_0



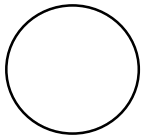
Binomial Trees (for k = 0, 1, 2, 3)

B_0

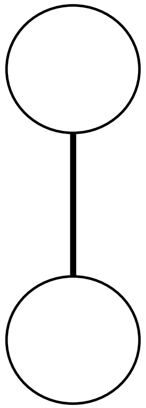


Binomial Trees (for k = 0, 1, 2, 3)

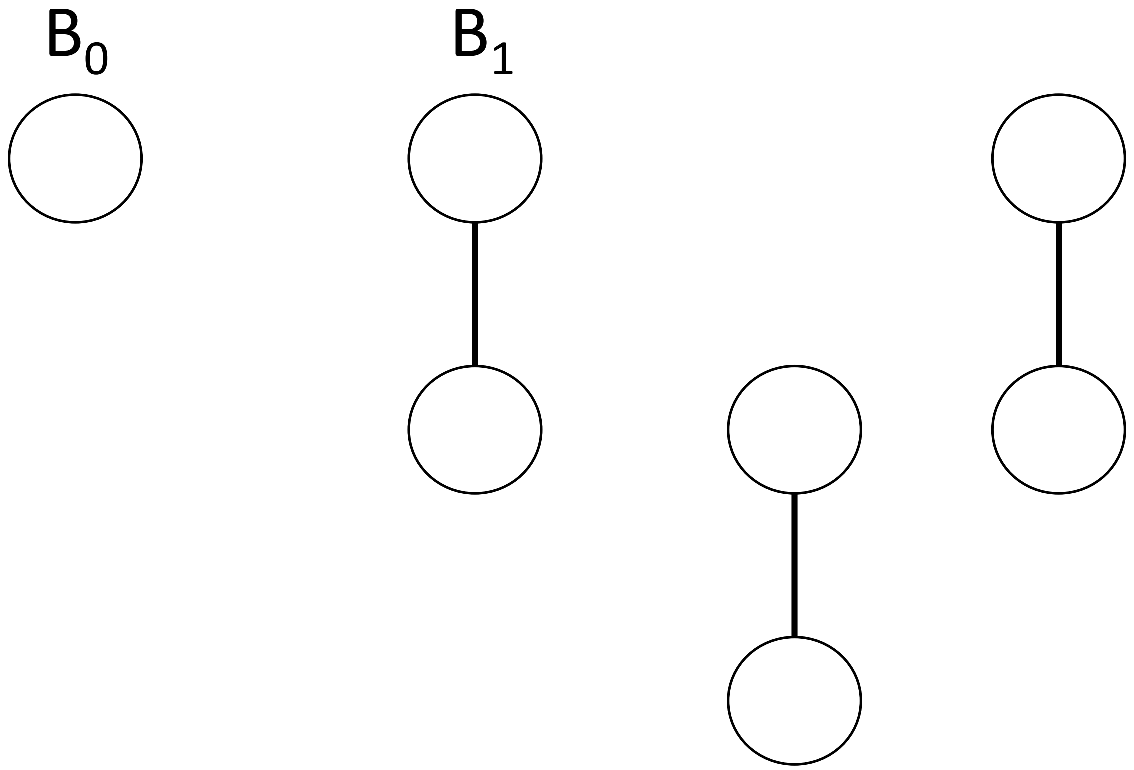
B_0



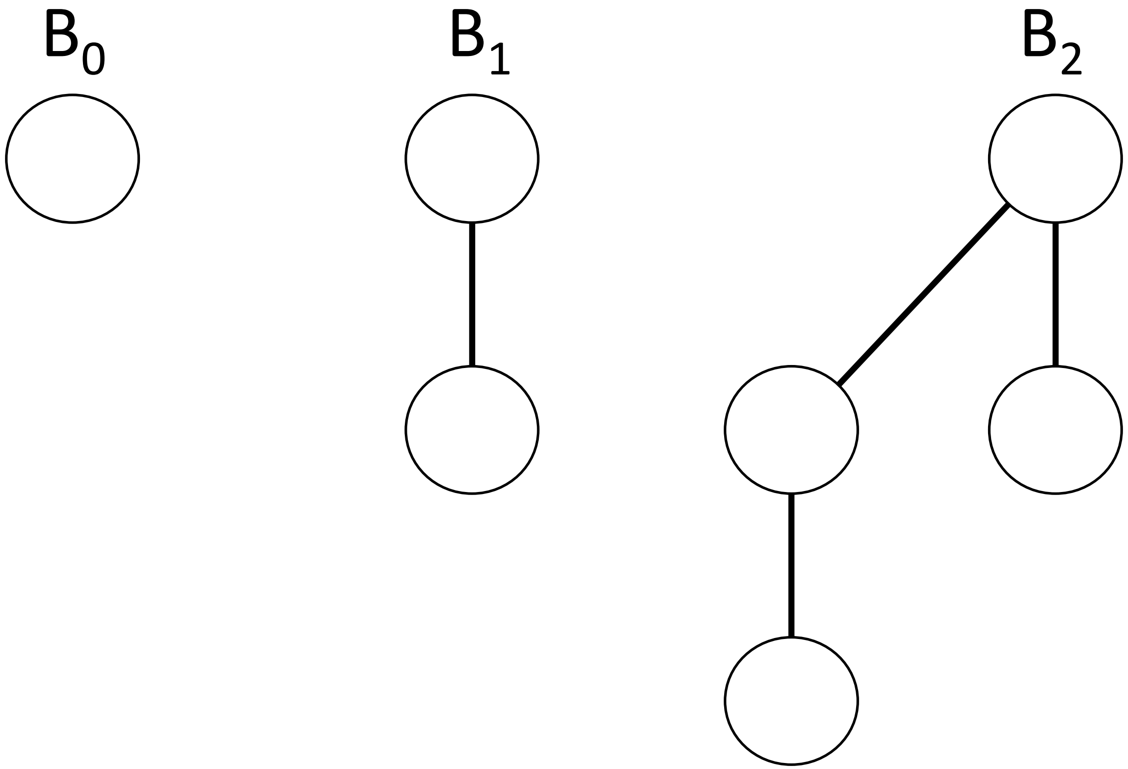
B_1



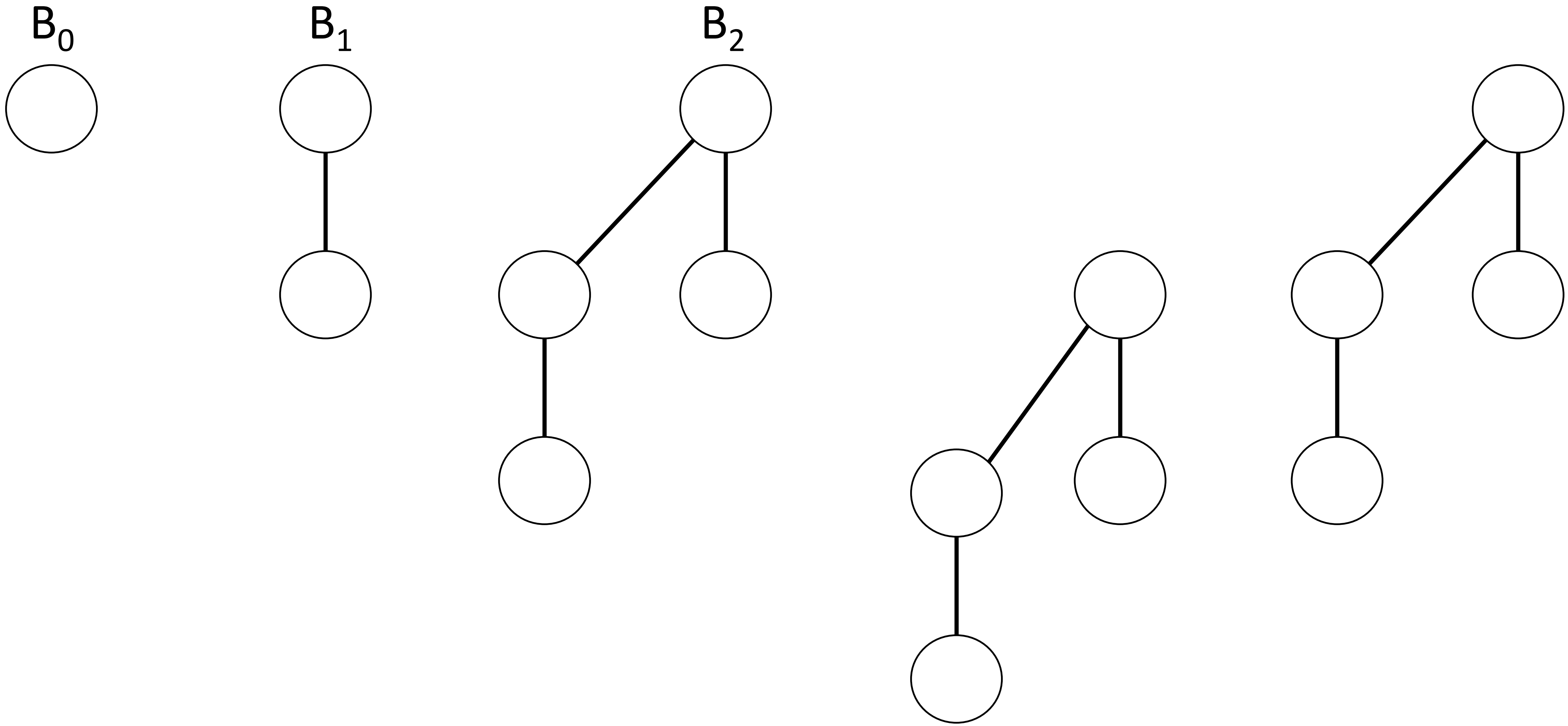
Binomial Trees (for k = 0, 1, 2, 3)



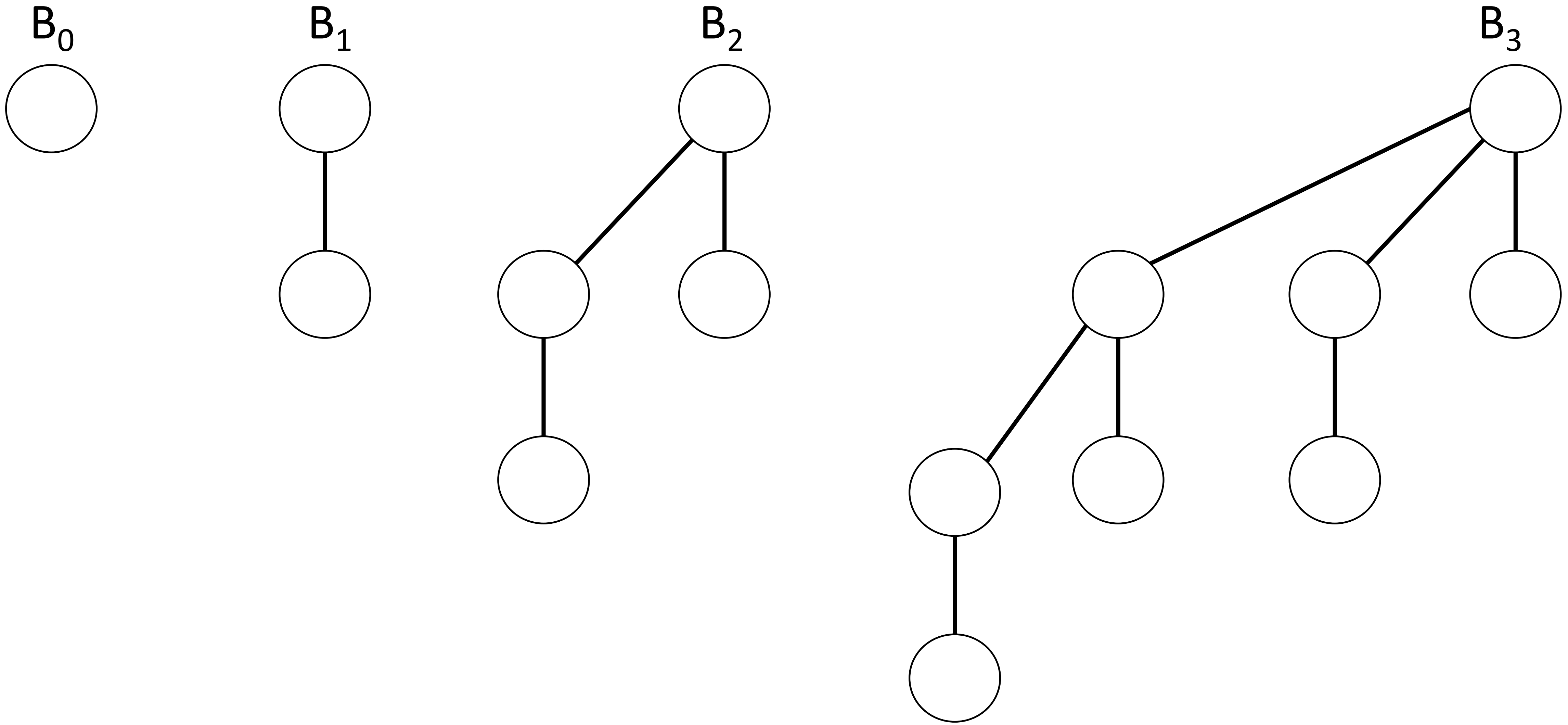
Binomial Trees (for k = 0, 1, 2, 3)



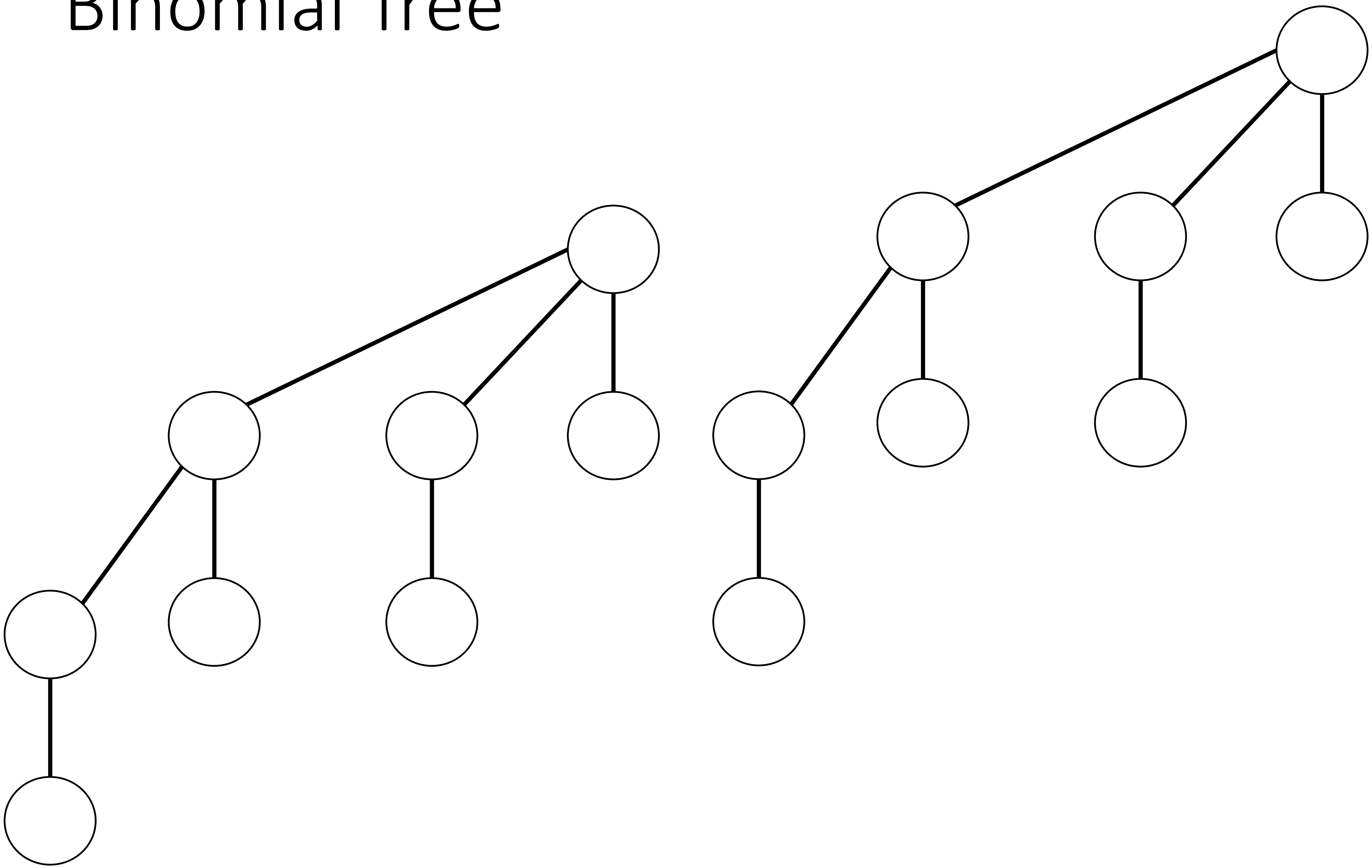
Binomial Trees (for k = 0, 1, 2, 3)



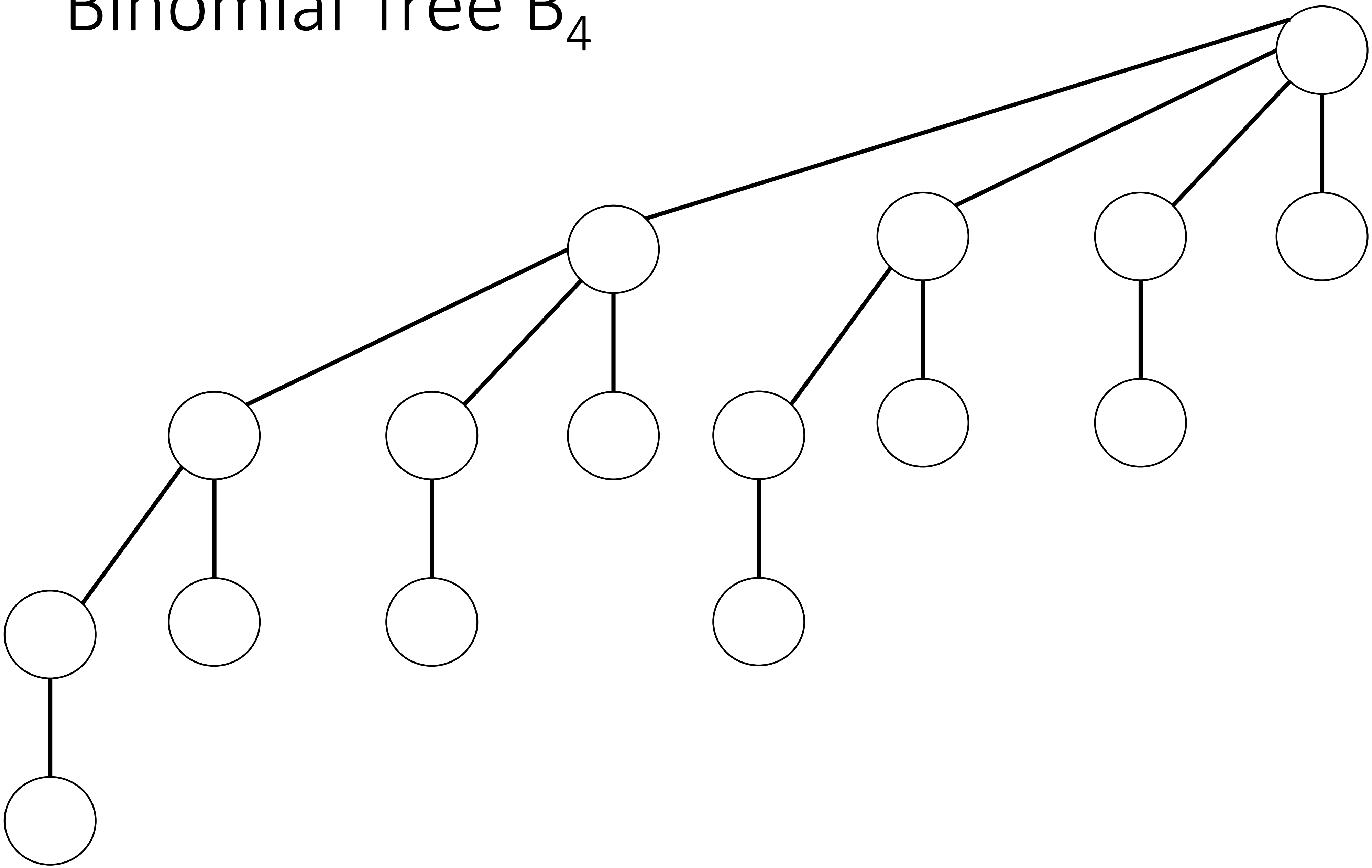
Binomial Trees (for k = 0, 1, 2, 3)



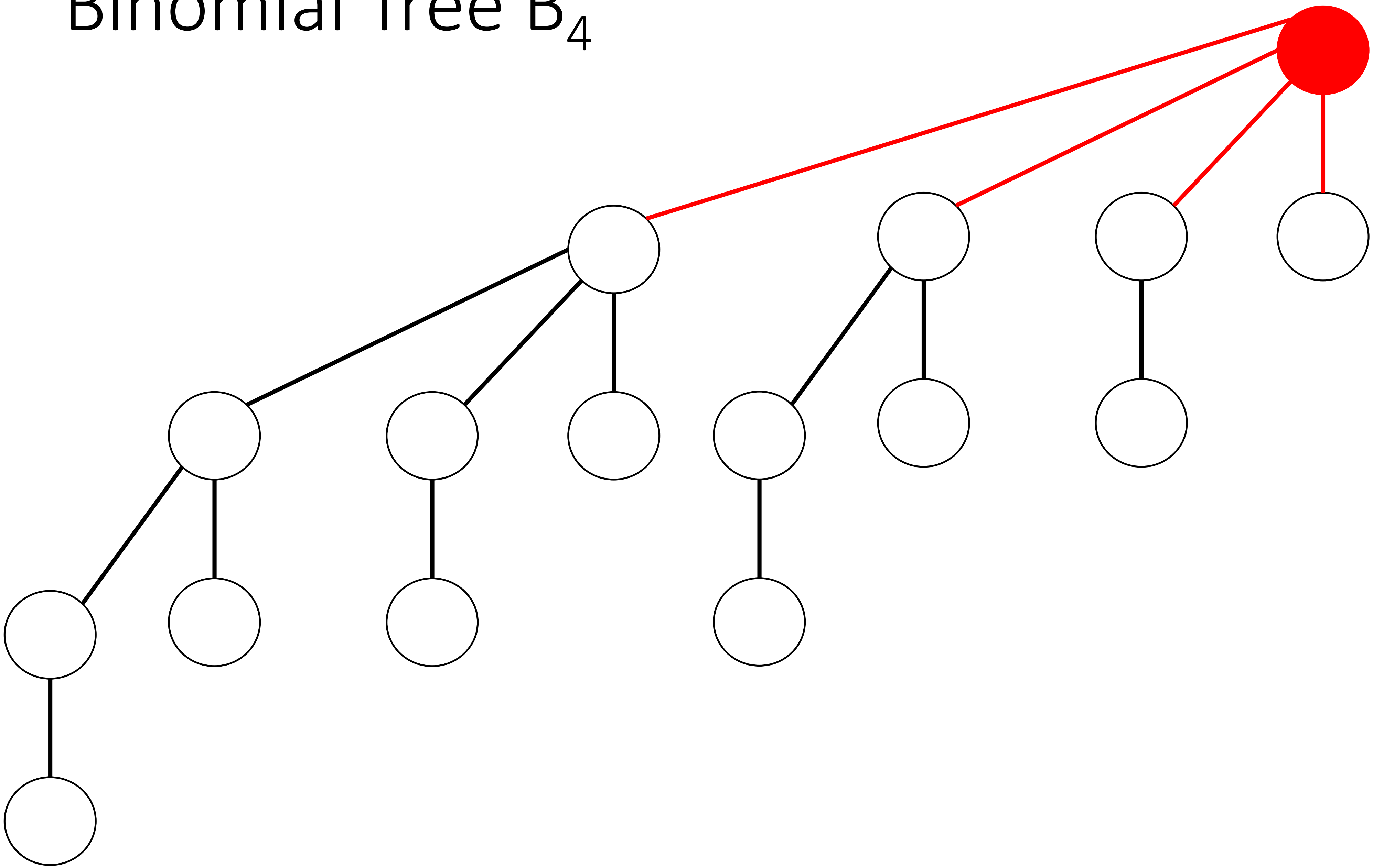
Binomial Tree



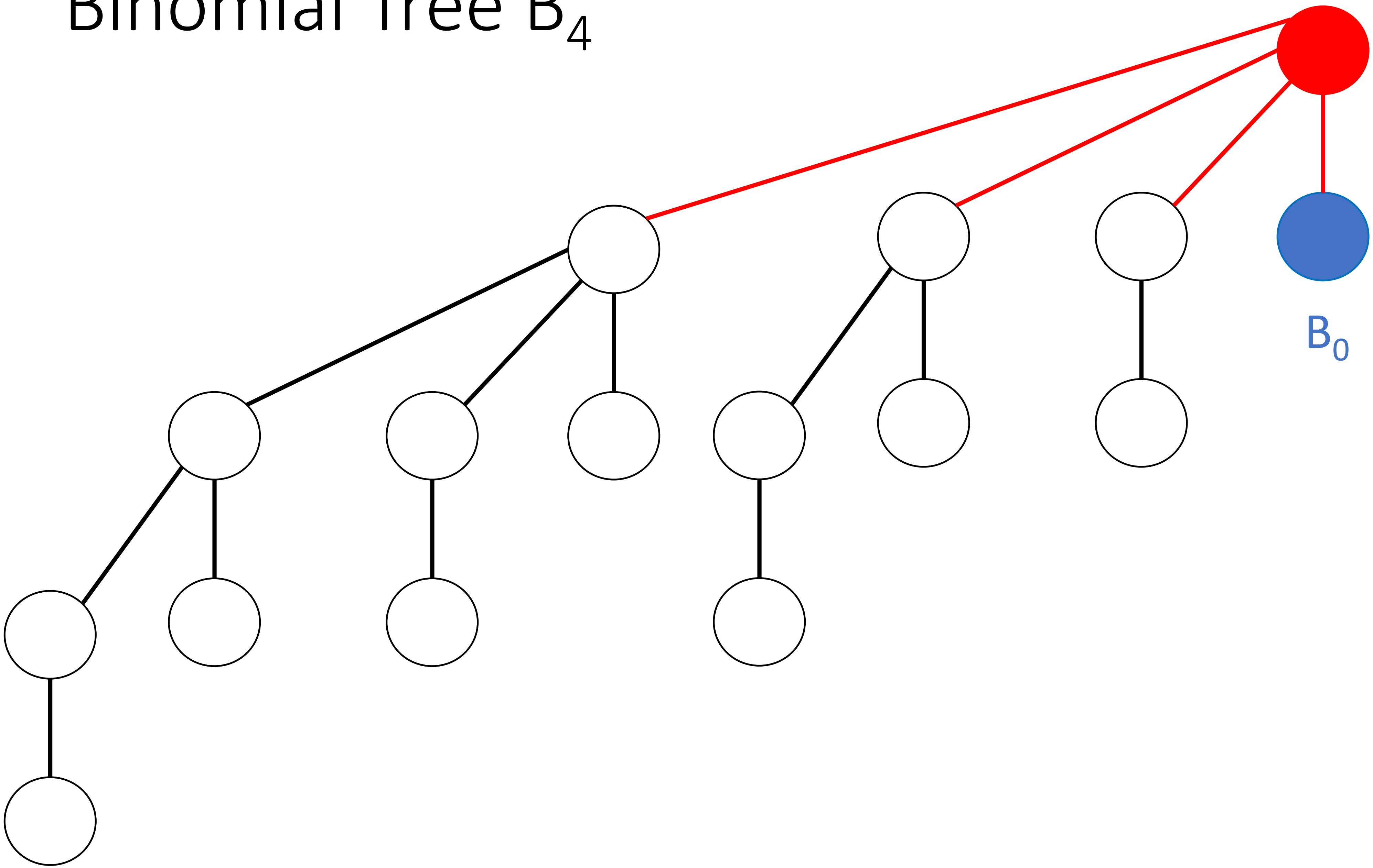
Binomial Tree B_4



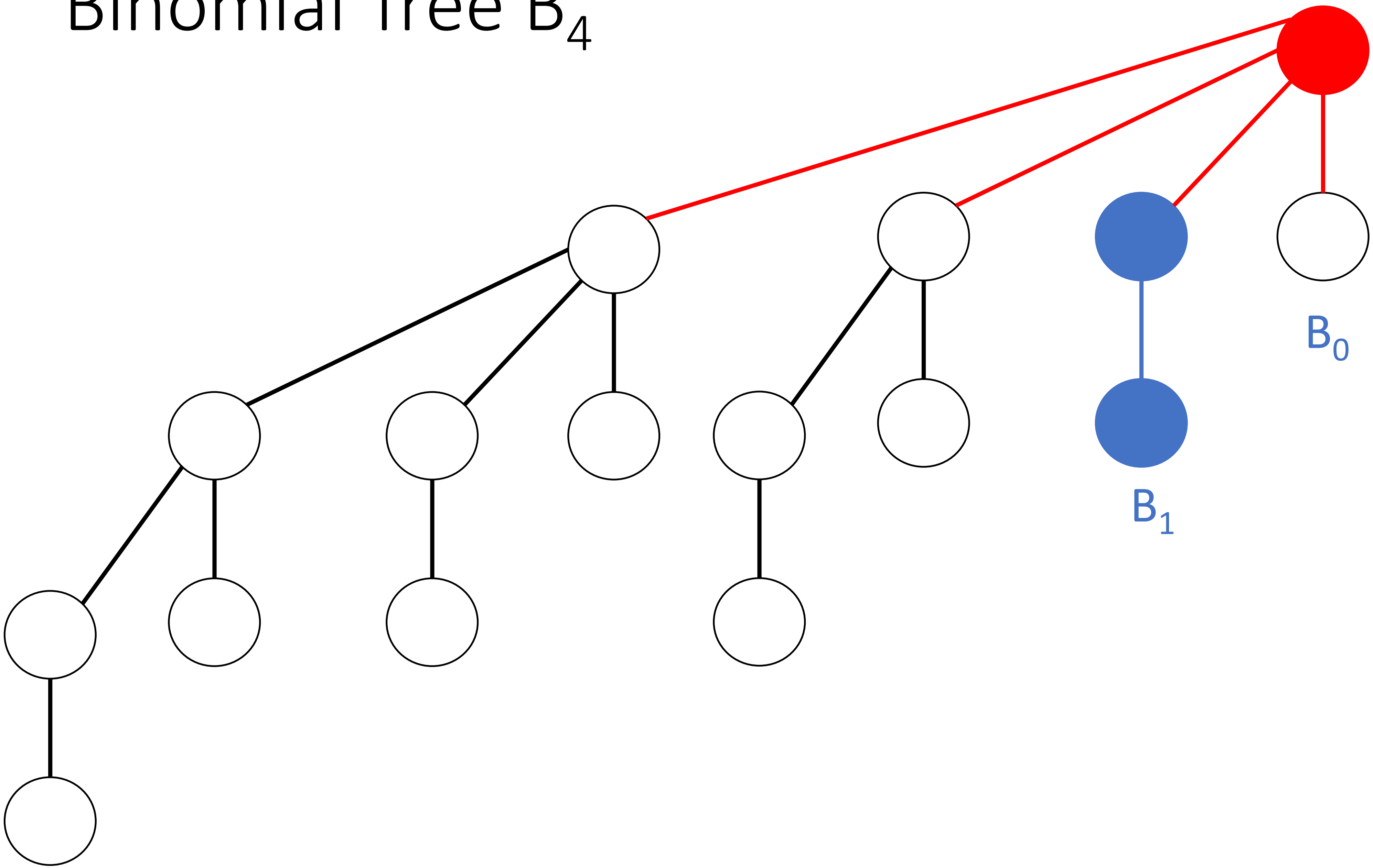
Binomial Tree B_4



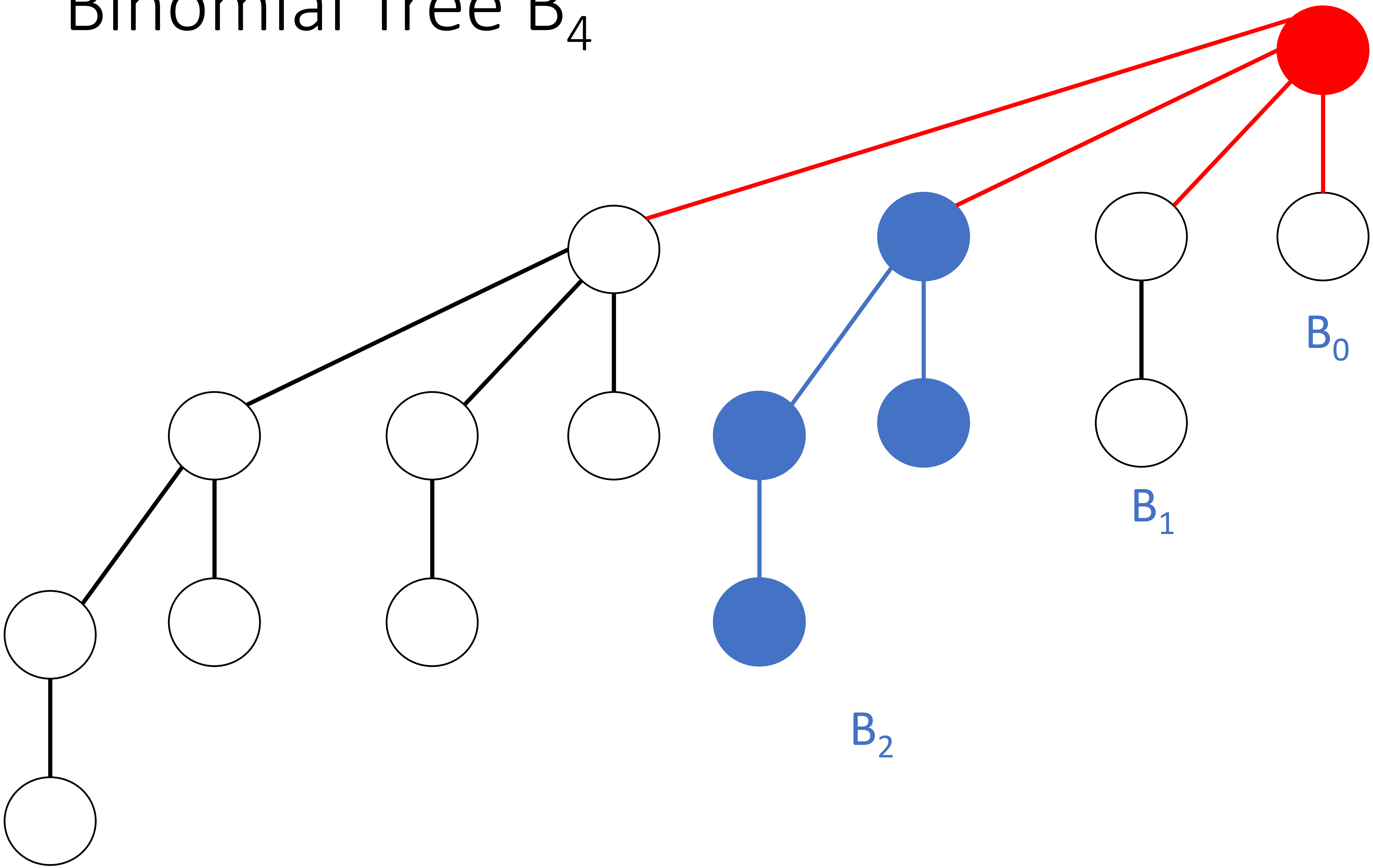
Binomial Tree B_4



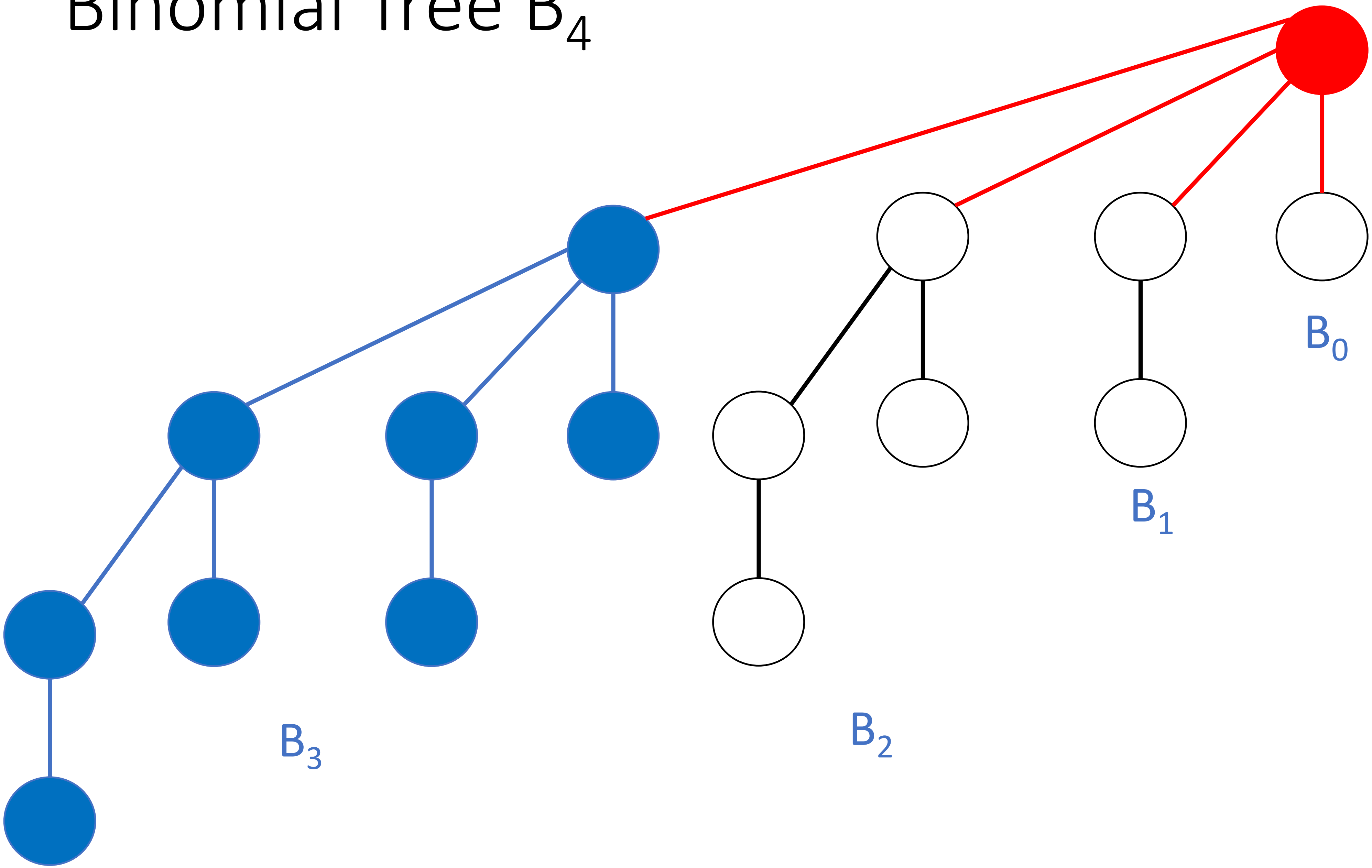
Binomial Tree B_4



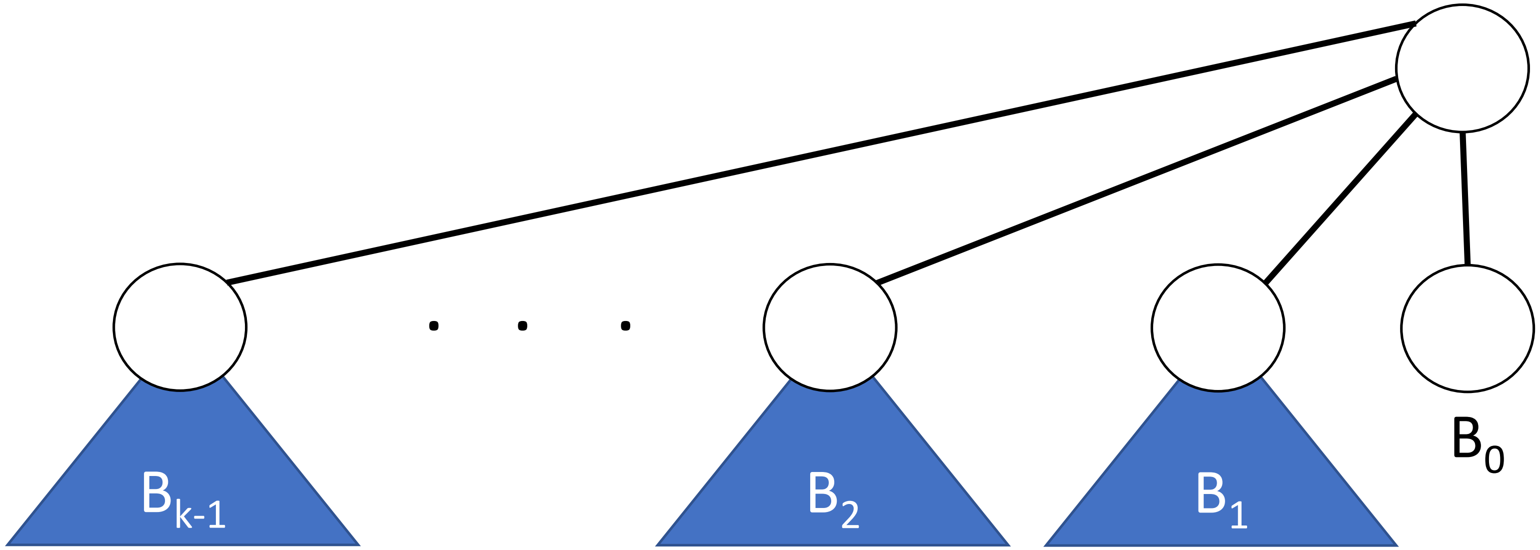
Binomial Tree B_4



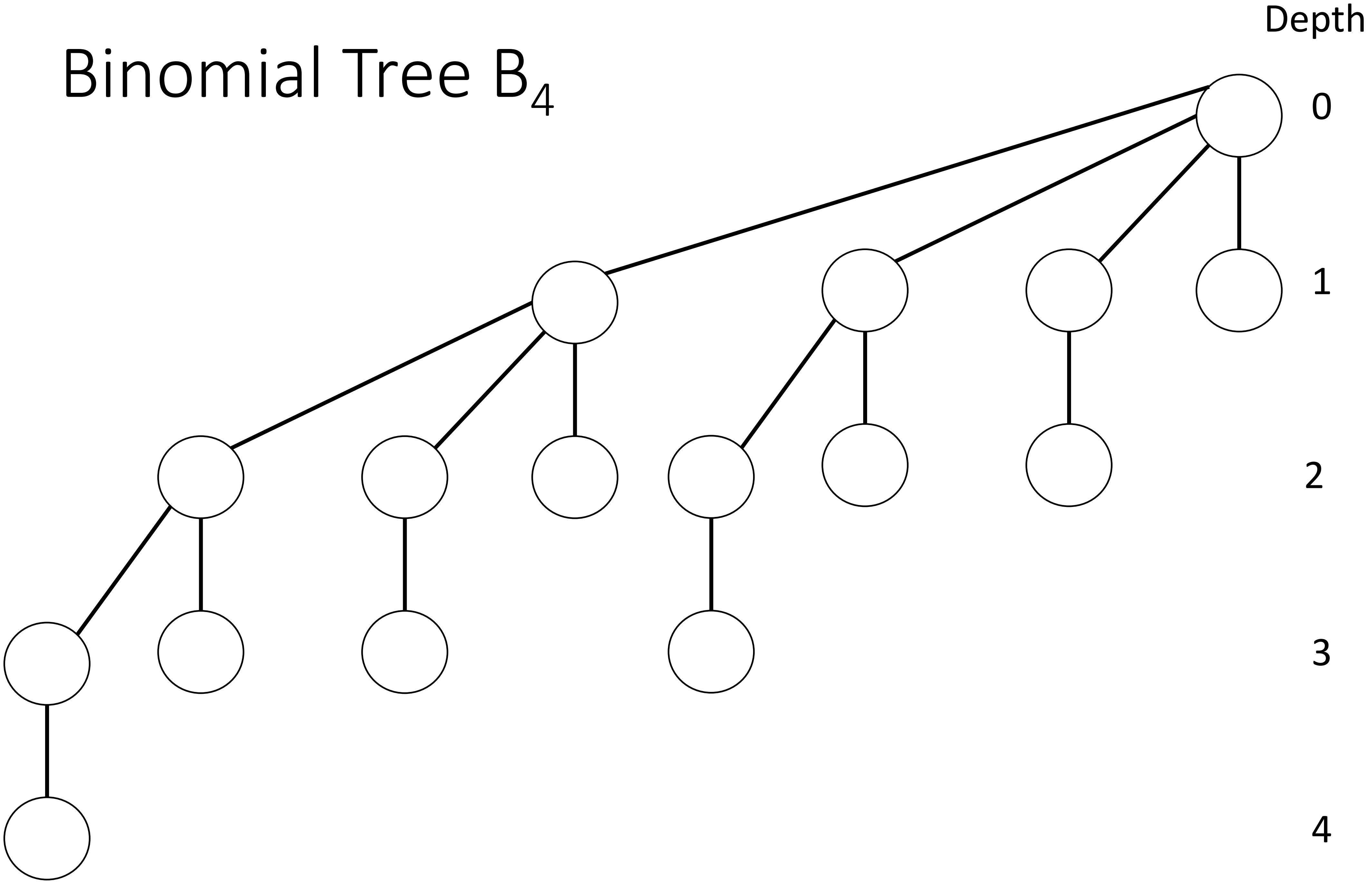
Binomial Tree B_4



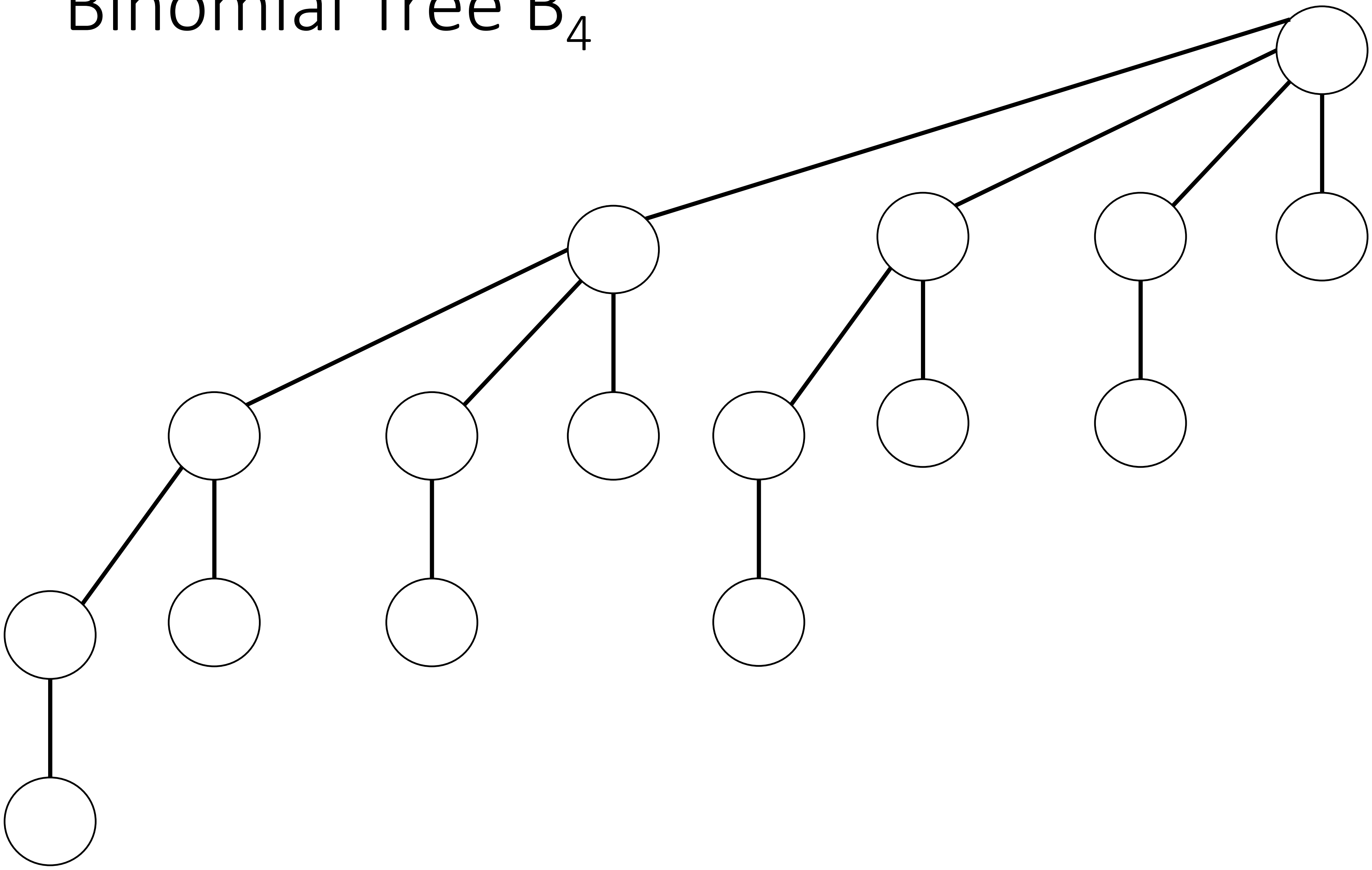
Binomial Tree B_k



Binomial Tree B_4

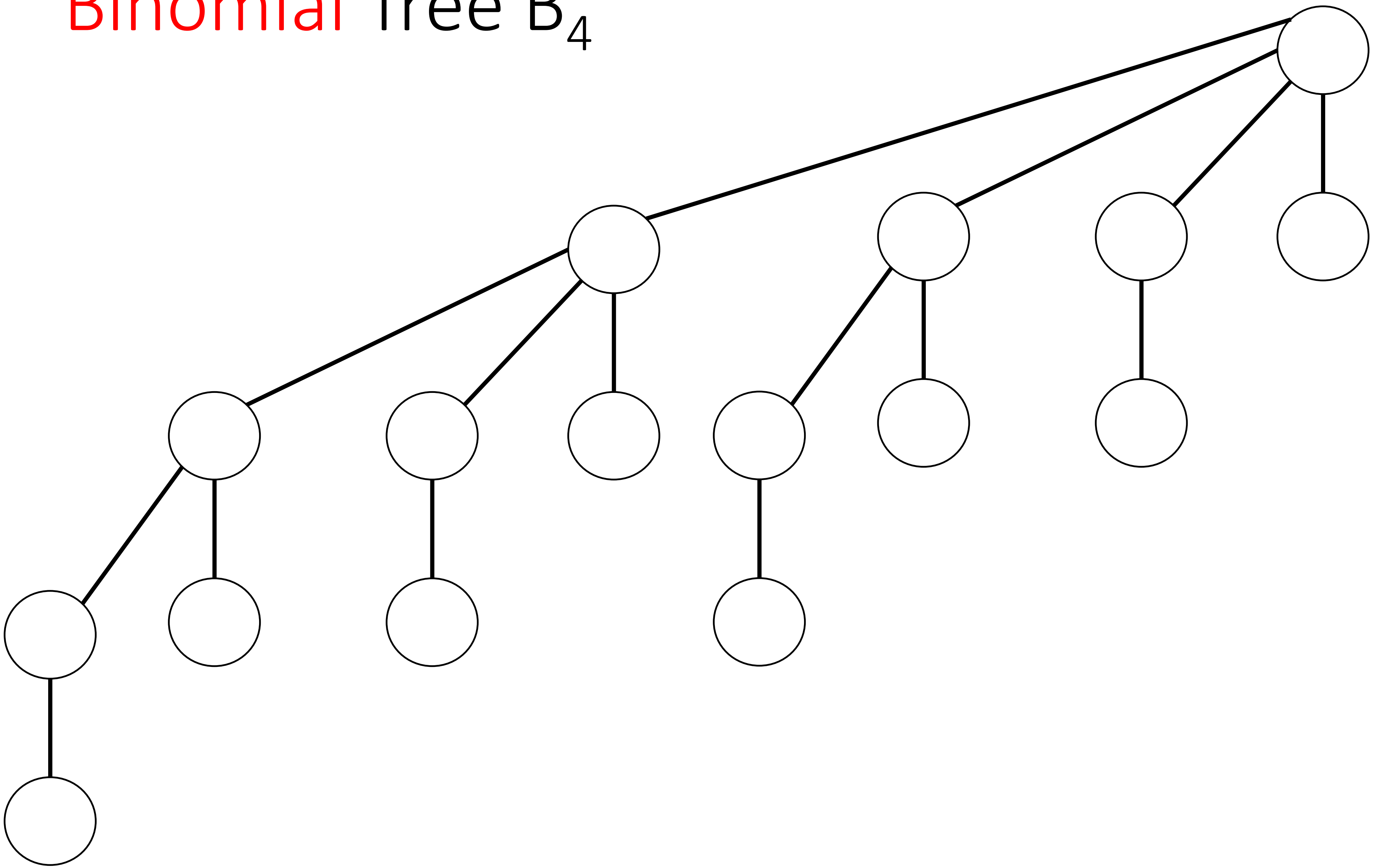


Binomial Tree B_4



Depth	# of nodes
0	1
1	4
2	6
3	4
4	1

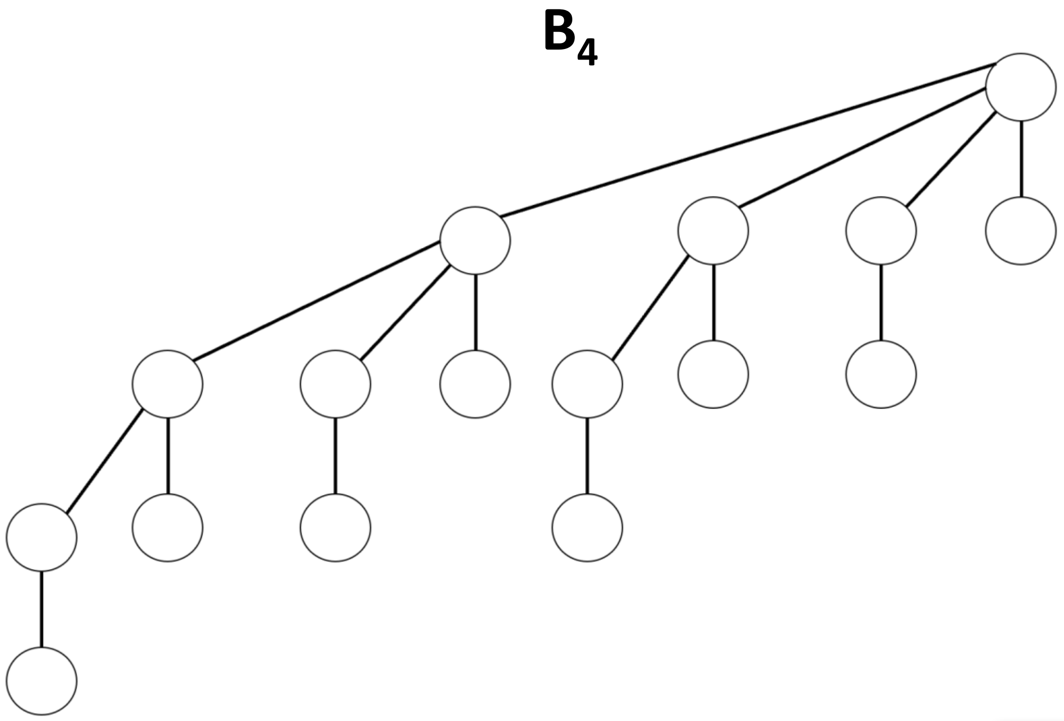
Binomial Tree B_4



Depth	# of nodes
0	$\binom{4}{0} = 1$
1	$\binom{4}{1} = 4$
2	$\binom{4}{2} = 6$
3	$\binom{4}{3} = 4$
4	$\binom{4}{4} = 1$

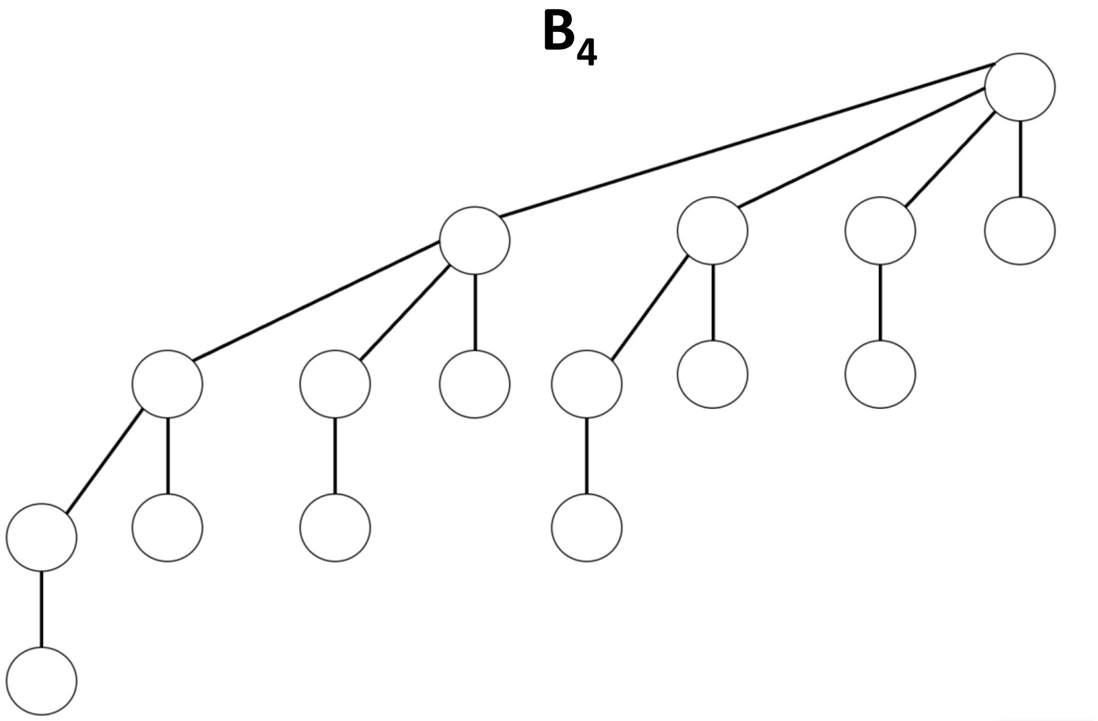
Binomial Tree properties

- B_k tree has:



Binomial Tree properties

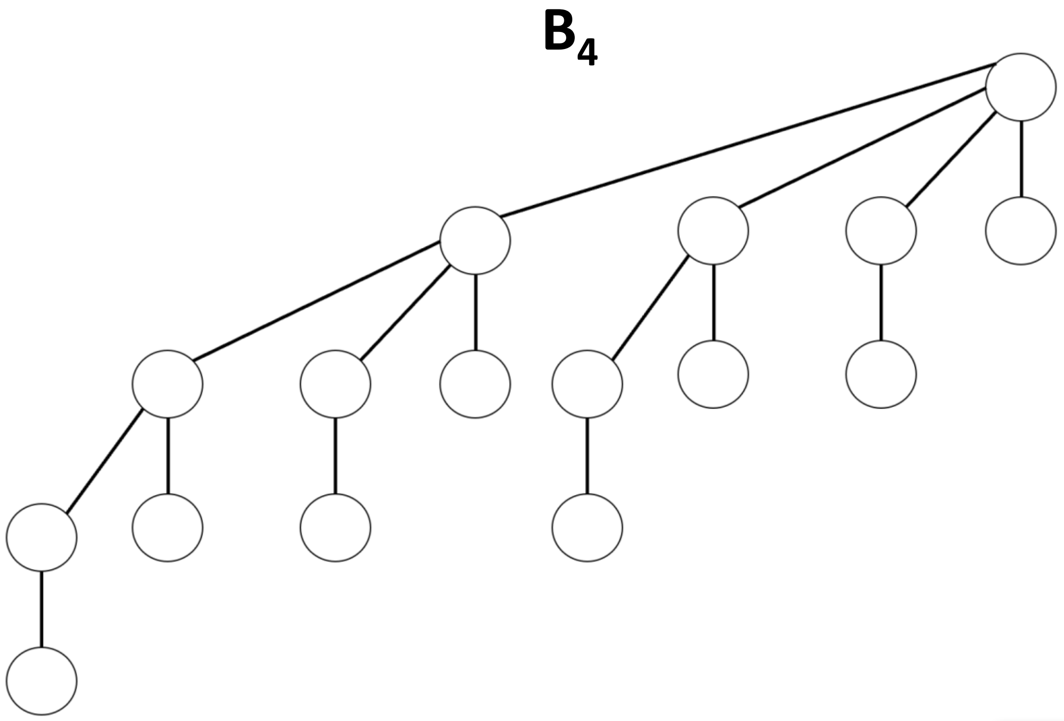
- B_k tree has:
  - Height k



Height = 4

Binomial Tree properties

- B_k tree has:
  - Height k
  - 2^k nodes

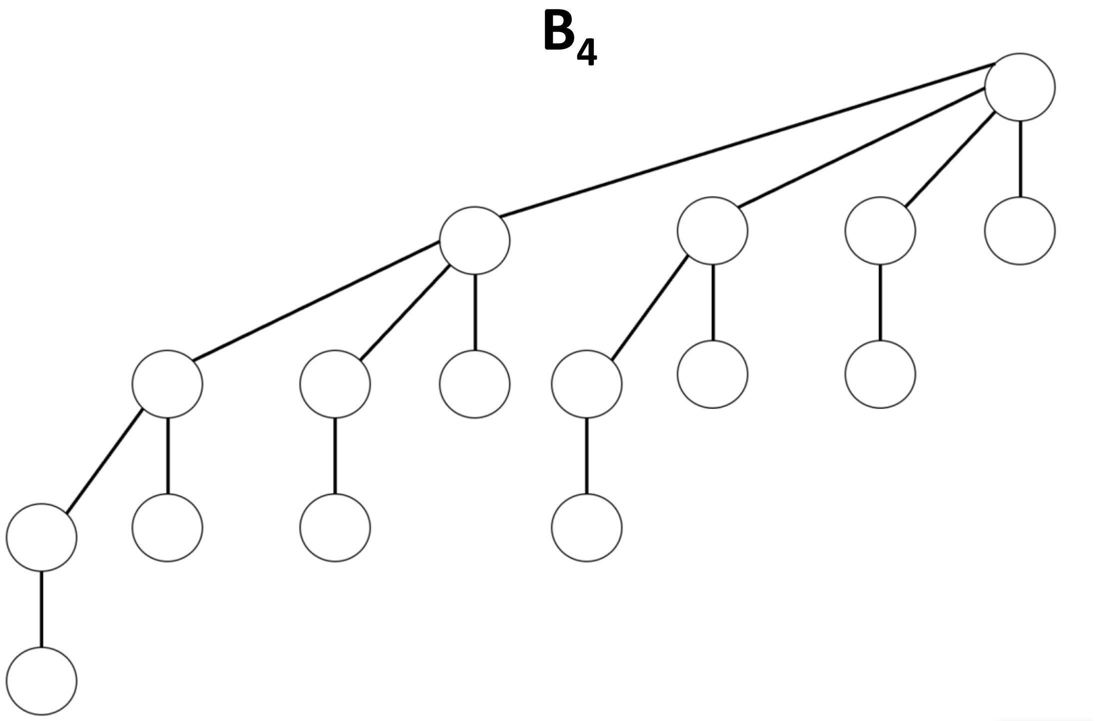


Height = 4

2^4 = 16 nodes

Binomial Tree properties

- B_k tree has:
  - Height k
  - 2^k nodes
  - $\binom{k}{d}$ nodes at depth d



Height = 4

2^4 = 16 nodes

$\binom{4}{2}$ = 6 nodes at level 2

Binomial Forest of size n (denoted F_n)

A sequence of B_k trees with *strictly decreasing* k's and a total of n nodes.

Example: Binomial Forest F_7 of n = 7 nodes

Example: Binomial Forest F_7 of n = 7 nodes

$n = 7 = < 1\ 1\ 1 >_2 = 2^2 + 2^1 + 2^0$

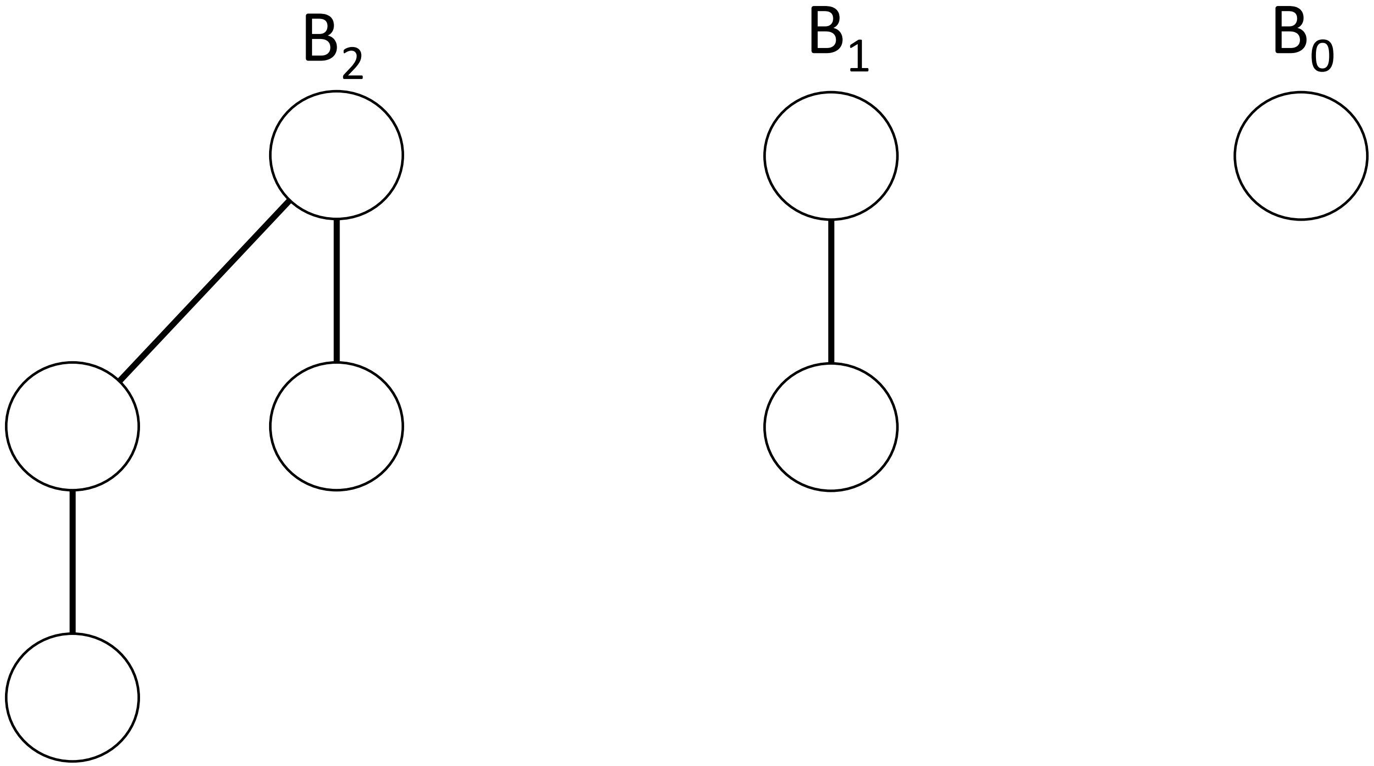
Example: Binomial Forest F_7 of n = 7 nodes

$n = 7 = < 1\ 1\ 1 >_2 = 2^2 + 2^1 + 2^0$ $F_7 = < B_2\ B_1\ B_0 >$

Example: Binomial Forest F_7 of n = 7 nodes

$n = 7 = < 1\ 1\ 1 >_2 = 2^2 + 2^1 + 2^0$ $F_7 = < B_2\ B_1\ B_0 >$

F_7 :



Example: Binomial Forest F_9 of n = 9 nodes

Example: Binomial Forest F_9 of n = 9 nodes

$n = 9 = < 1\ 0\ 0\ 1 >_2 = 2^3 + 2^0$

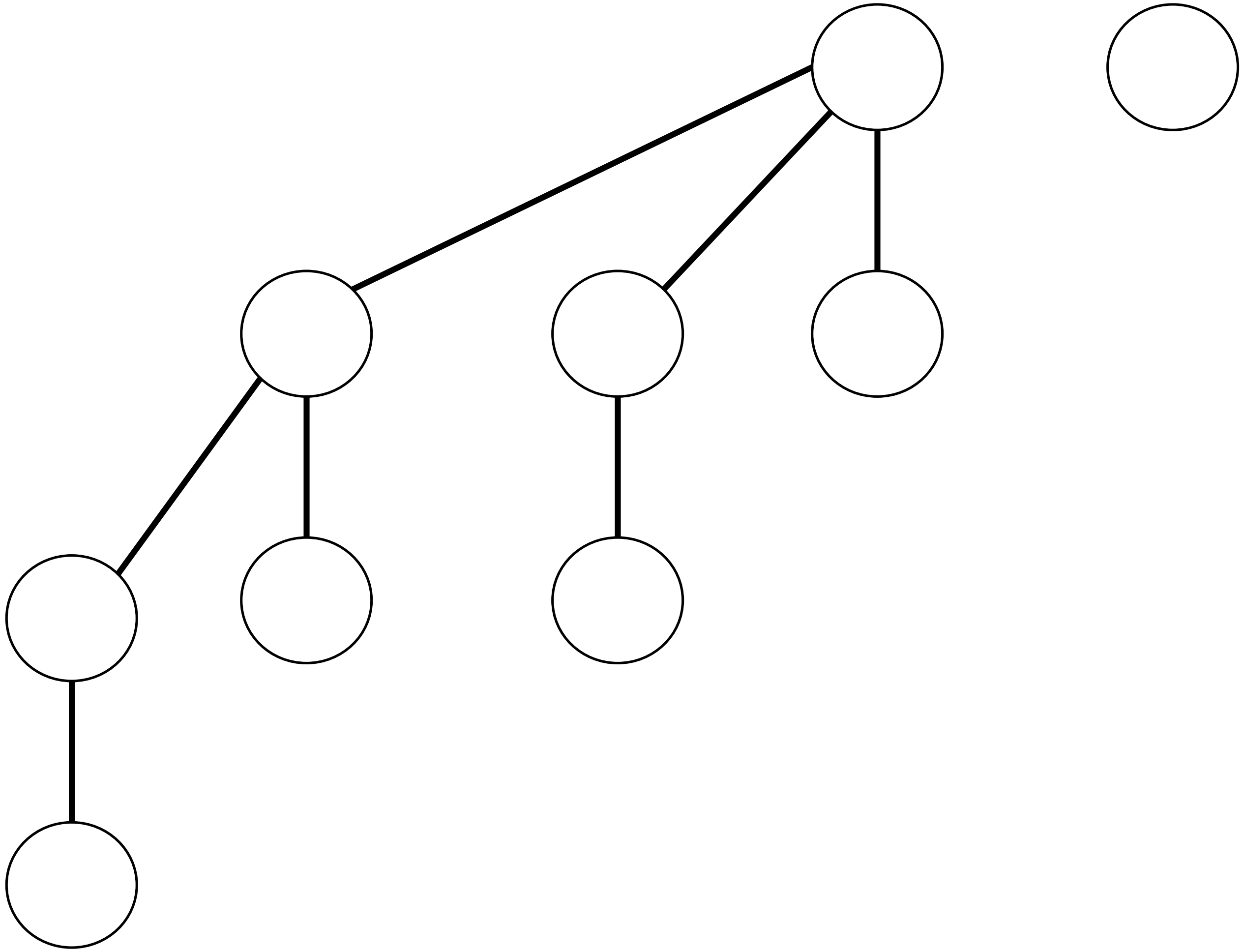
Example: Binomial Forest F_9 of $n = 9$ nodes

$n = 9 = < 1\ 0\ 0\ 1 >_2 = 2^3 + 2^0$ $\qquad$ $F_9 = < B_3\ B_0 >$

Example: Binomial Forest F_9 of n = 9 nodes

$n = 9 = < 1\ 0\ 0\ 1 >_2 = 2^3 + 2^0$ $\qquad$ $F_9 = < B_3\ B_0 >$

F_9 : $\qquad$ B_3 $\qquad$ B_0



Binomial Forest F_n

Binomial Forest F_n

- F_n has n nodes.

Binomial Forest F_n

- F_n has n nodes.
- $n = \langle b_t, b_{t-1}, ..., b_0 \rangle_2$.

Binomial Forest F_n

- F_n has n nodes.
- $n = \langle b_t, b_{t-1}, \ldots., b_0 \rangle_2$. Note that $t = \lfloor \log_2 n \rfloor$

Binomial Forest F_n

- F_n has n nodes.
- $n = \langle b_t, b_{t-1}, \ldots., b_0 \rangle_2$. Note that $t = \lfloor \log_2 n \rfloor$
- F_n : <all trees B_i such that bit $b_i = 1$>

Binomial Forest F_n

- F_n has n nodes.
- $n = <b_t, b_{t-1}, \ldots., b_0>_2$. Note that $t = \lfloor \log_2 n \rfloor$
- F_n : <all trees B_i such that bit $b_i = 1$>

- Let α(n) = # of 1's in binary representation of n

Binomial Forest F_n

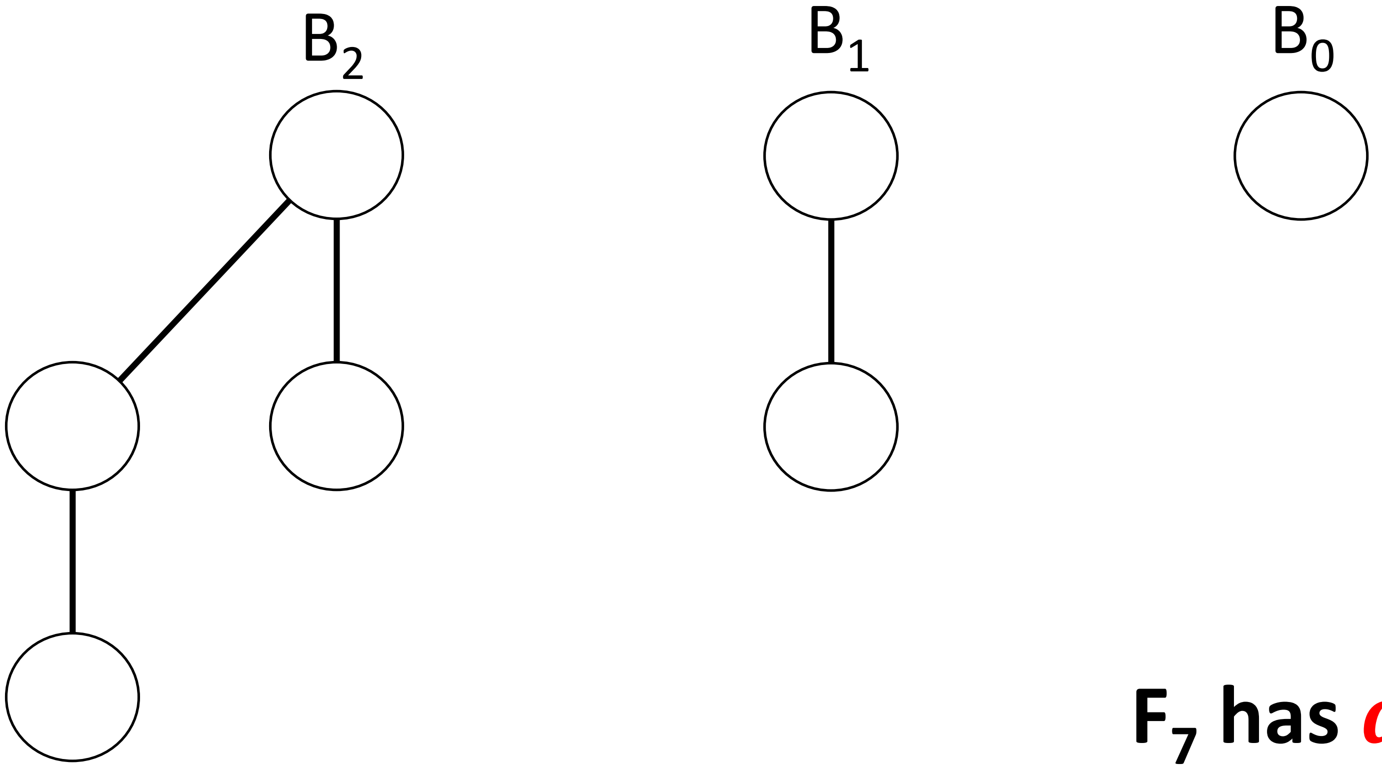
- F_n has n nodes.
- $n = \langle b_t, b_{t-1}, \ldots, b_0 \rangle_2$.   Note that $t = \lfloor \log_2 n \rfloor$
- F_n : <all trees B_i such that bit $b_i = 1$>

- Let α(n) = # of 1's in binary representation of n
    - F_n has α(n) trees

Example: Binomial Forest F_7 of n = 7 nodes

$n = 7 = \langle 1\ 1\ 1 \rangle_2$

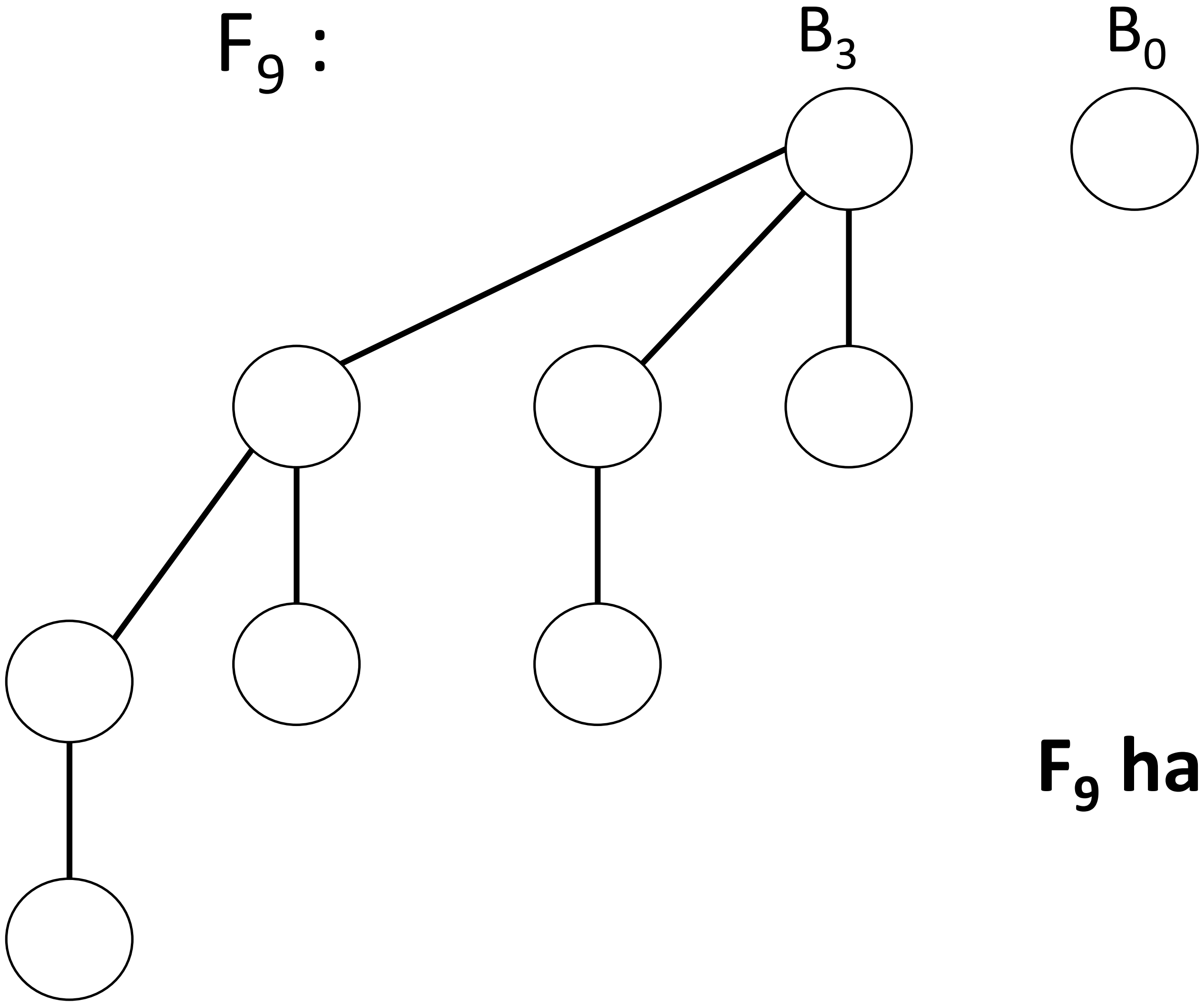
F_7 :



F_7 has $\alpha(7) = 3$ trees

Example: Binomial Forest F_9 of n = 9 nodes

$n = 9 = \langle 1\ 0\ 0\ 1 \rangle_2$



F_9 has $\alpha(9) = 2$ trees

Binomial Forest F_n

- F_n has n nodes.
- $n = <b_t, b_{t-1}, \ldots., b_0>_2$.   Note that $t = \lfloor \log_2 n \rfloor$
- F_n : <all trees B_i such that bit $b_i = 1$>

- Let α(n) = # of 1’s in binary representation of n
    - F_n has α(n) trees

Binomial Forest F_n

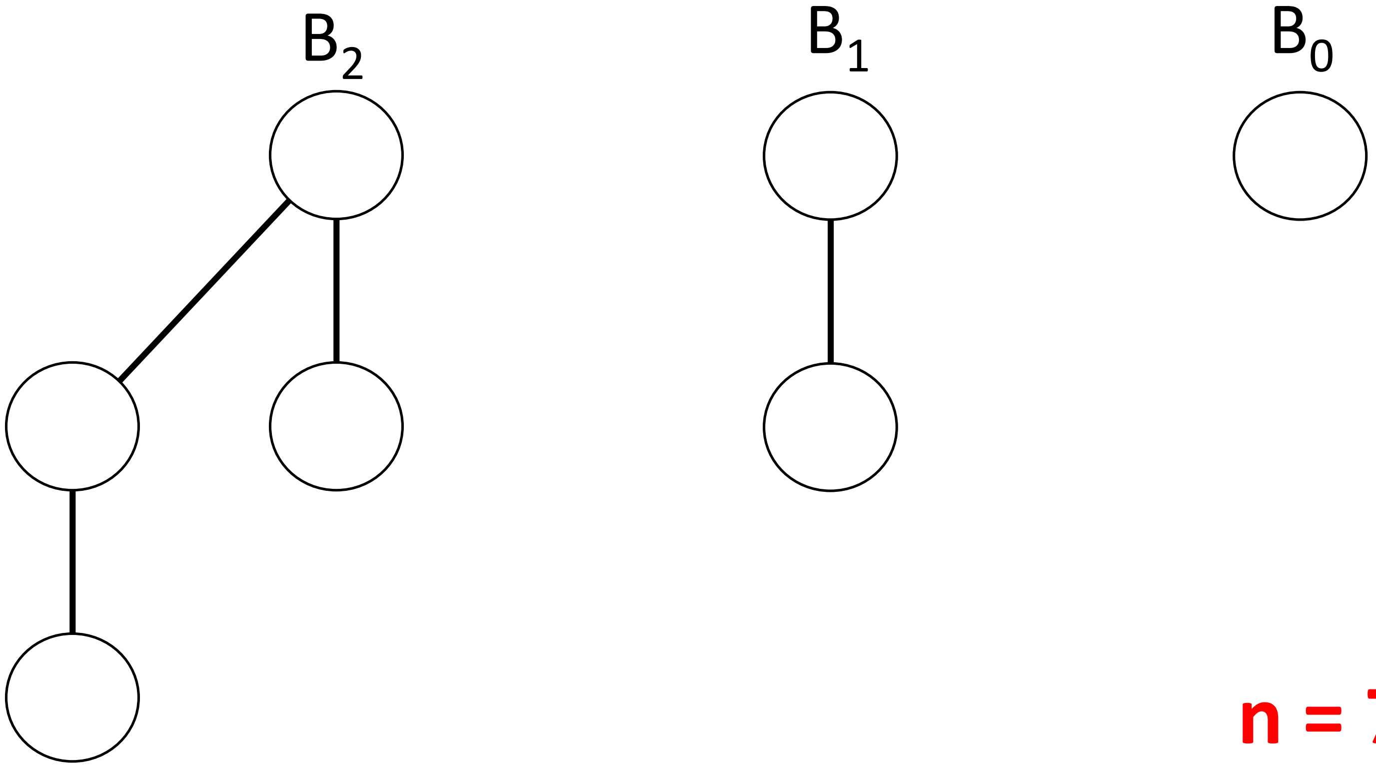
- F_n has n nodes.
- n = <b_t, b_{t-1}, …., b_0>$_2$.   Note that $t = \lfloor \log_2 n \rfloor$
- F_n : <all trees B_i such that bit b_i = 1>

- Let α(n) = # of 1's in binary representation of n
  - F_n has α(n) trees
  - F_n has n - α(n) edges

Example: Binomial Forest F_7 of n = 7 nodes

$n = 7 = \langle 1\ 1\ 1 \rangle_2$

F_7 :

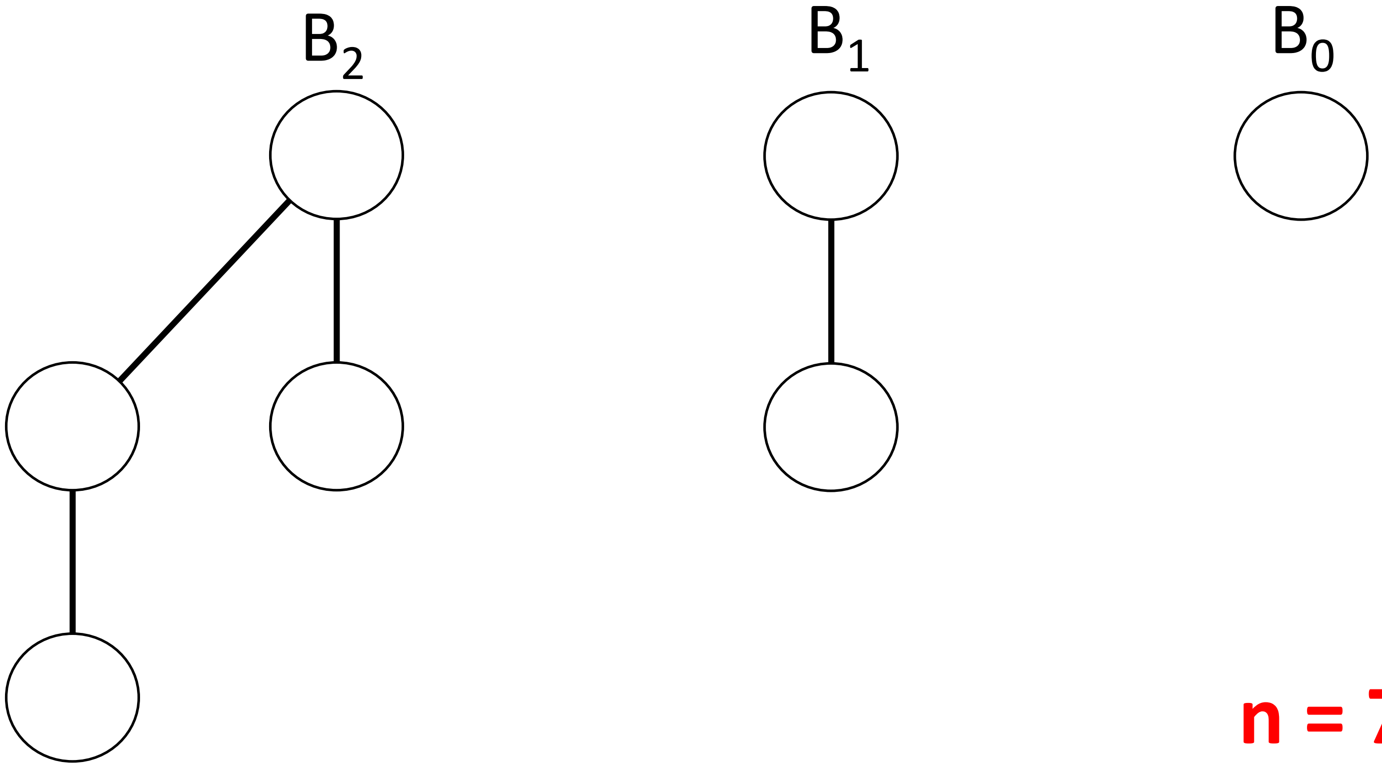


n = 7, $\alpha(7) = 3$

Example: Binomial Forest F_7 of n = 7 nodes

$n = 7 = < 1\ 1\ 1 >_2$

F_7 :



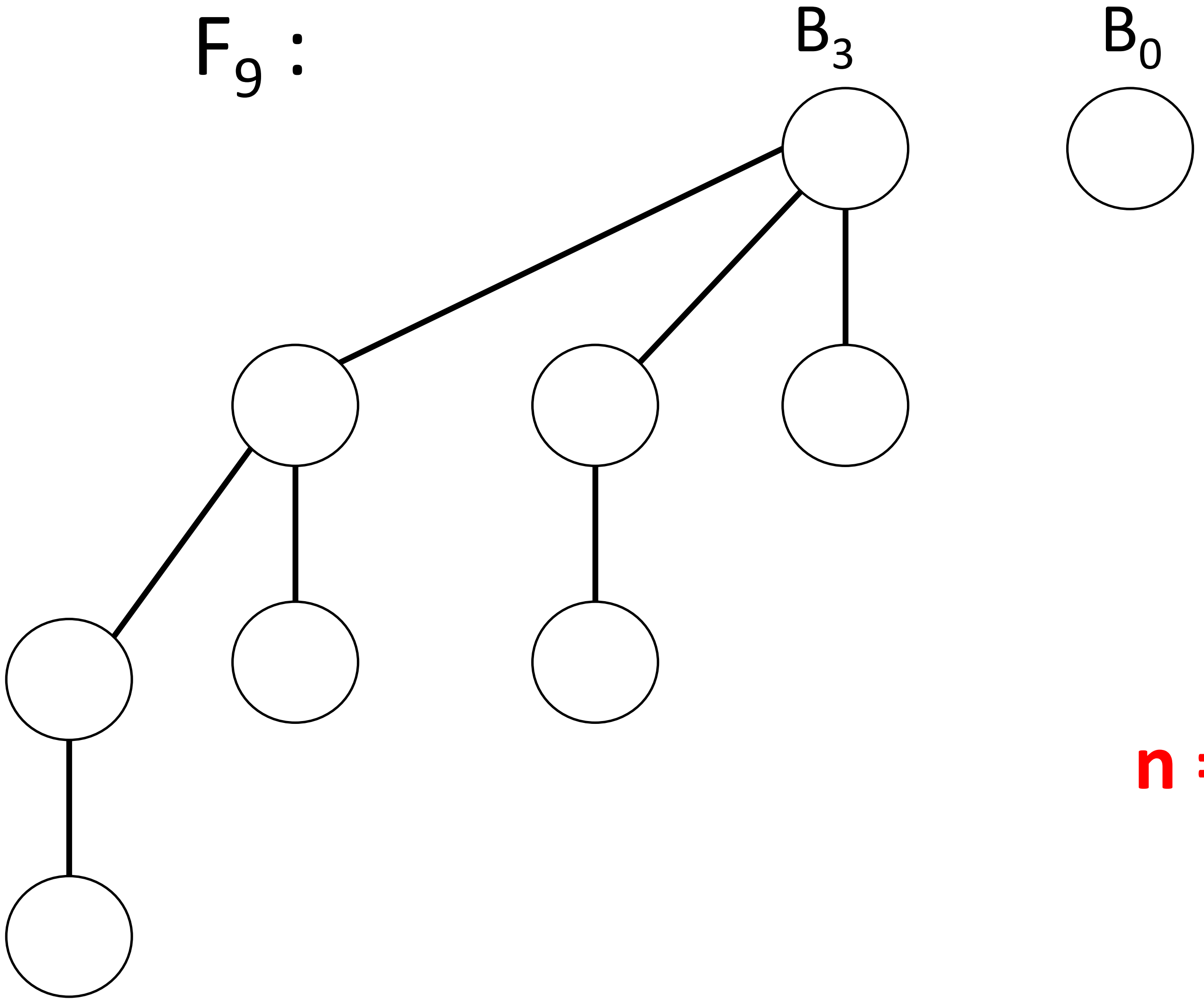
$n = 7, \alpha(7) = 3$

F_7 has $7-\alpha(7) = 4$ edges

Example: Binomial Forest F_9 of n = 9 nodes

$n = 9 = \langle 1\ 0\ 0\ 1 \rangle_2$

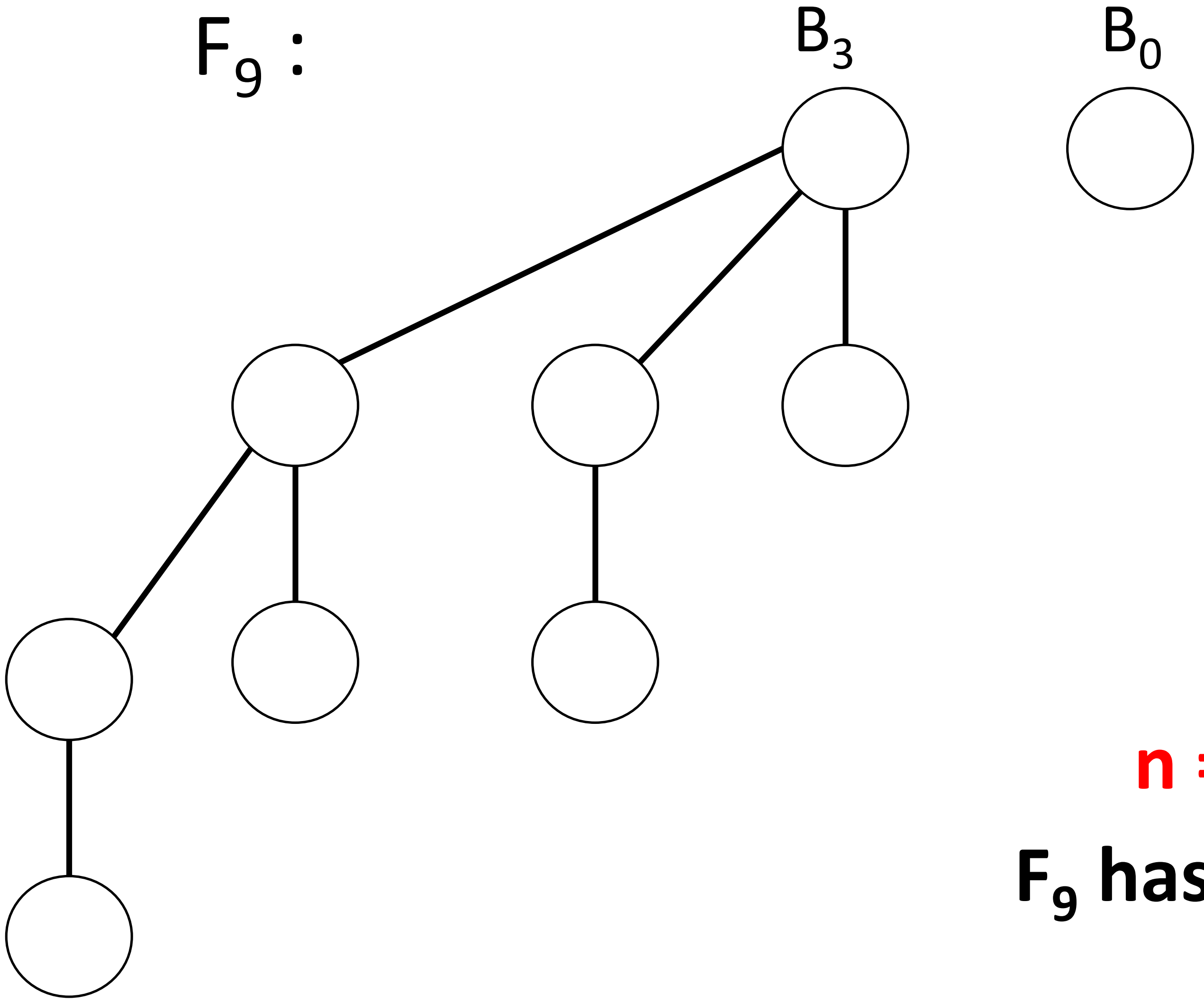
F_9 :



$n = 9$, $\alpha(9) = 2$

Example: Binomial Forest F_9 of n = 9 nodes

$n = 9 = \langle 1\ 0\ 0\ 1 \rangle_2$



$n = 9$, $\alpha(9) = 2$

F_9 has $9-\alpha(9) = 7$ edges

Binomial Forest F_n

- F_n has n nodes.
- $n = \langle b_t, b_{t-1}, \ldots., b_0 \rangle_2$.   Note that $t = \lfloor \log_2 n \rfloor$
- F_n : <all trees B_i such that bit $b_i = 1$>

- Let α(n) = # of 1's in binary representation of n
  - F_n has α(n) trees
  - F_n has n - α(n) edges

A Min Binomial Heap of n elements is a Binomial Forest F_n such that

1. Each node of F_n stores one element

A Min Binomial Heap of n elements is a Binomial Forest F_n such that

1. Each node of F_n stores one element

2. Each B_k tree of F_n is Min-Heap ordered

A Min Binomial Heap of n elements is a Binomial Forest F_n such that

1. Each node of F_n stores one element

2. Each B_k tree of F_n is Min-Heap ordered,
   i.e. each tree satisfies the Min-Heap property

Example of a Min Binomial heap

S = {10, 13, 1, 3, 8, 18, 7}   n = 7 elements

Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ $n = 7$ elements

Put S in $F_7 = < B_2\ B_1\ B_0 >$

Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ $n = 7$ elements

Put S in $F_7 = < B_2\ B_1\ B_0 >$

Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ $n = 7$ elements

Put S in $F_7 = < B_2\ B_1\ B_0 >$

B_0

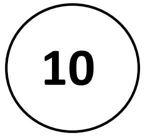
10

Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ n = 7 elements

Put S in $F_7 = < B_2\ B_1\ B_0 >$

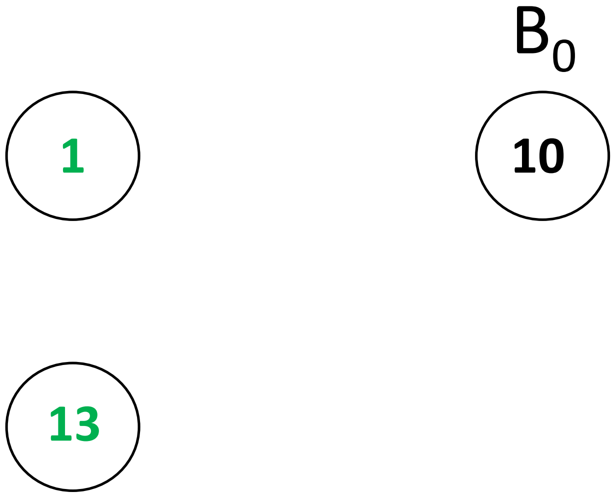
B_0



Example of a Min Binomial heap

S = {10, 13, 1, 3, 8, 18, 7}   n = 7 elements

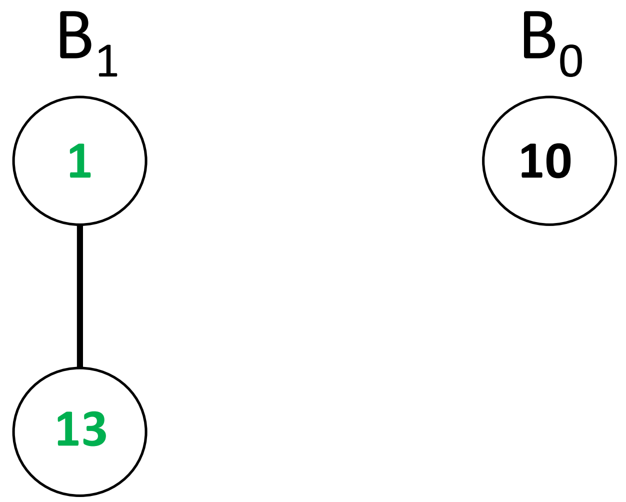
Put S in F_7 = < B_2 B_1 B_0 >



Example of a Min Binomial heap

S = {10, 13, 1, 3, 8, 18, 7}   n = 7 elements

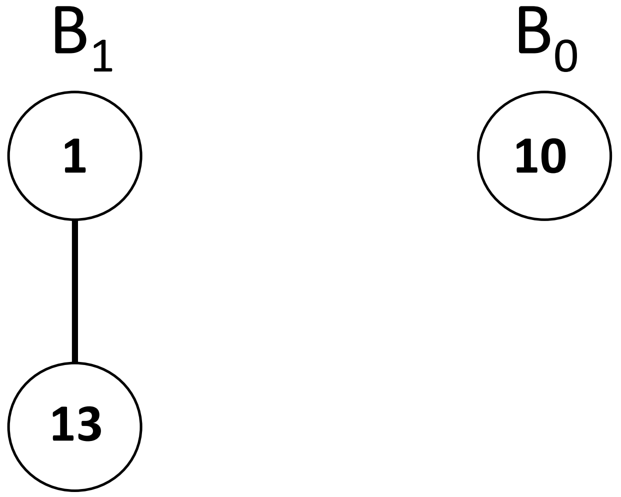
Put S in $F_7 = < B_2\ B_1\ B_0 >$



Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ $n = 7$ elements

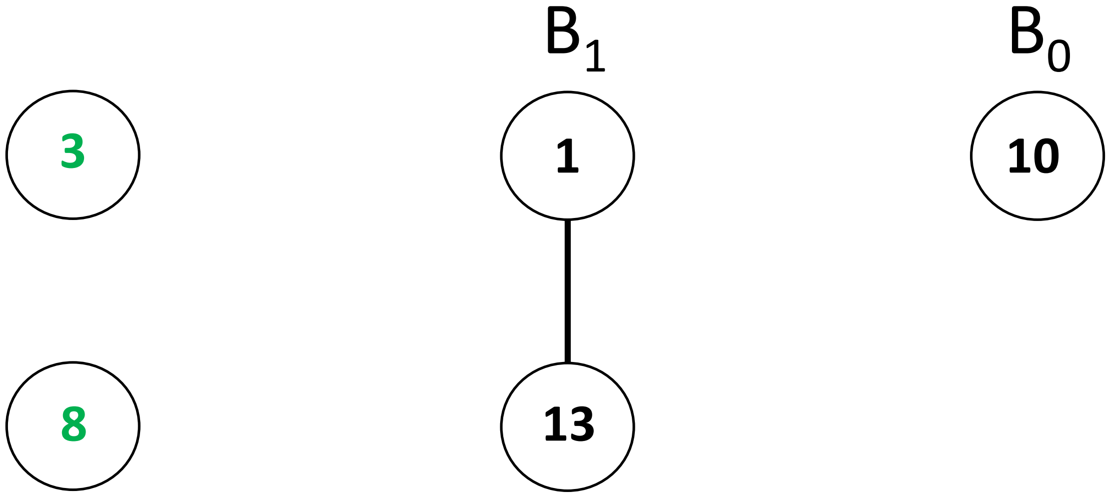
Put S in $F_7 = < B_2\ B_1\ B_0 >$



Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ $n = 7$ elements

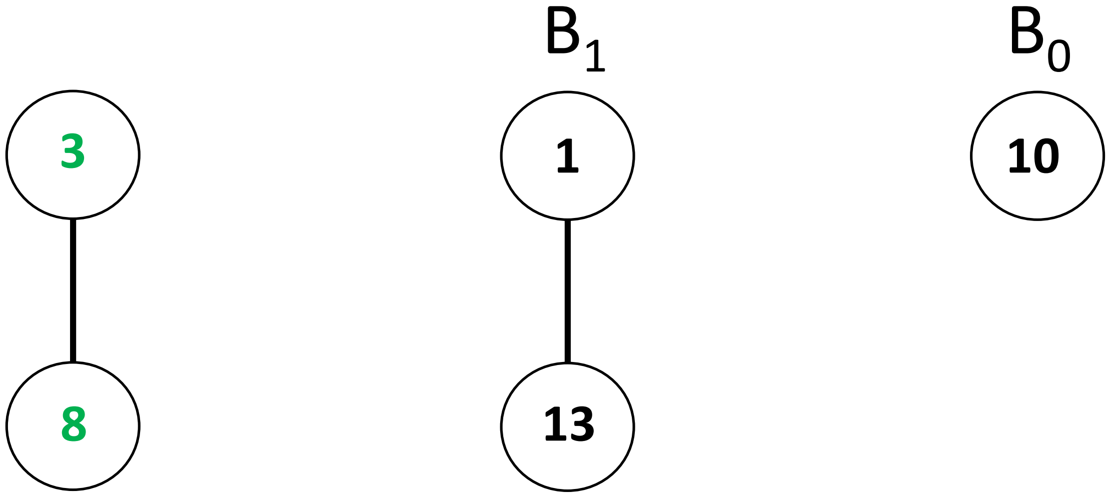
Put S in $F_7 = < B_2\ B_1\ B_0 >$



Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ $n = 7$ elements

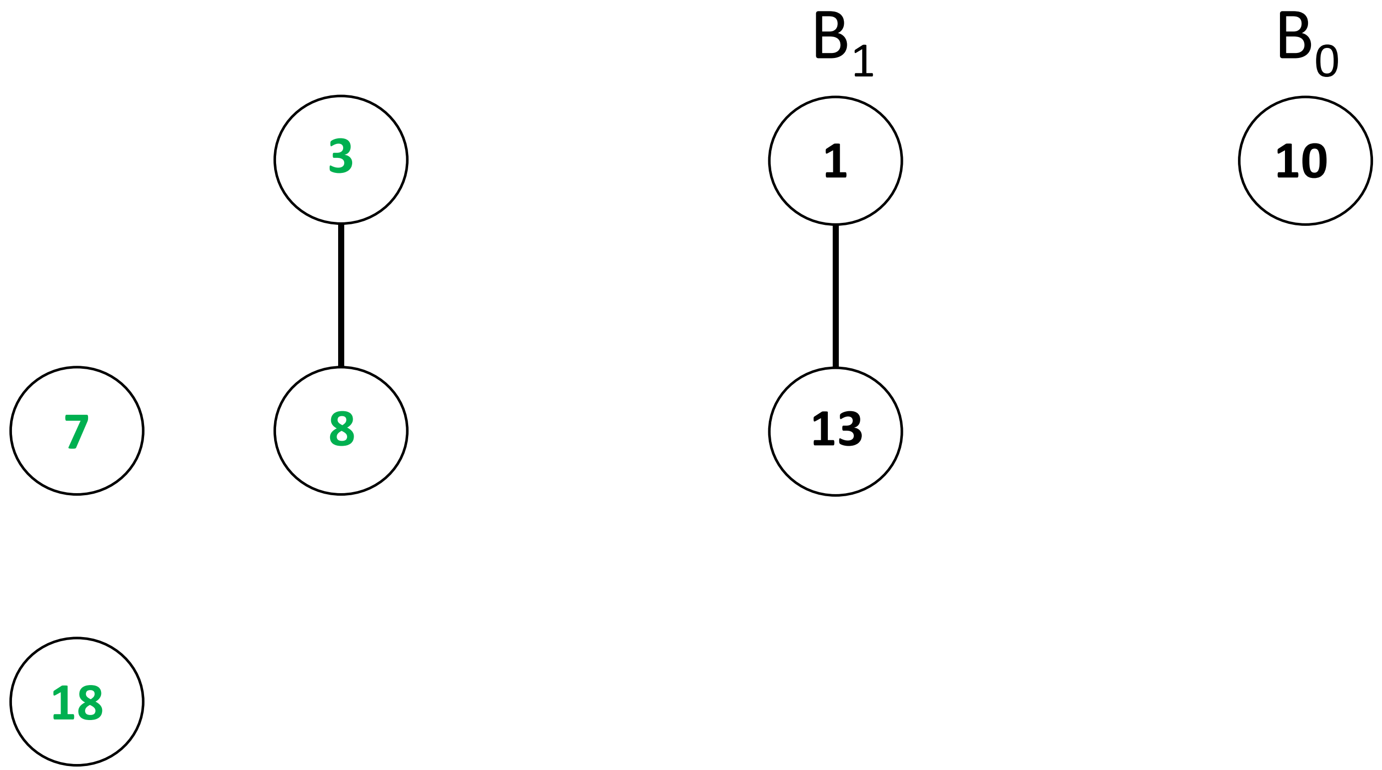
Put S in $F_7 = \langle B_2\ B_1\ B_0 \rangle$



Example of a Min Binomial heap

S = {10, 13, 1, 3, 8, 18, 7}   n = 7 elements

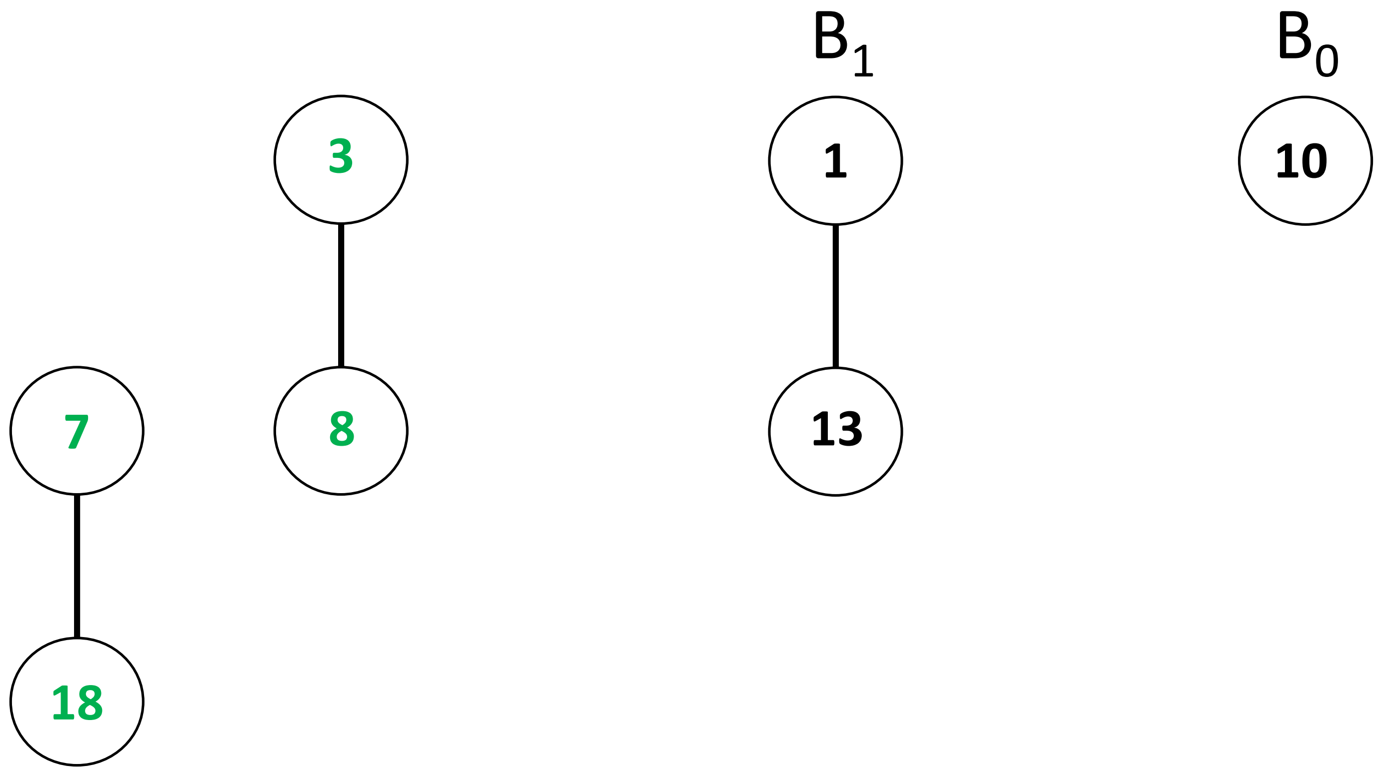
Put S in F_7 = < B_2 B_1 B_0 >



Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ n = 7 elements

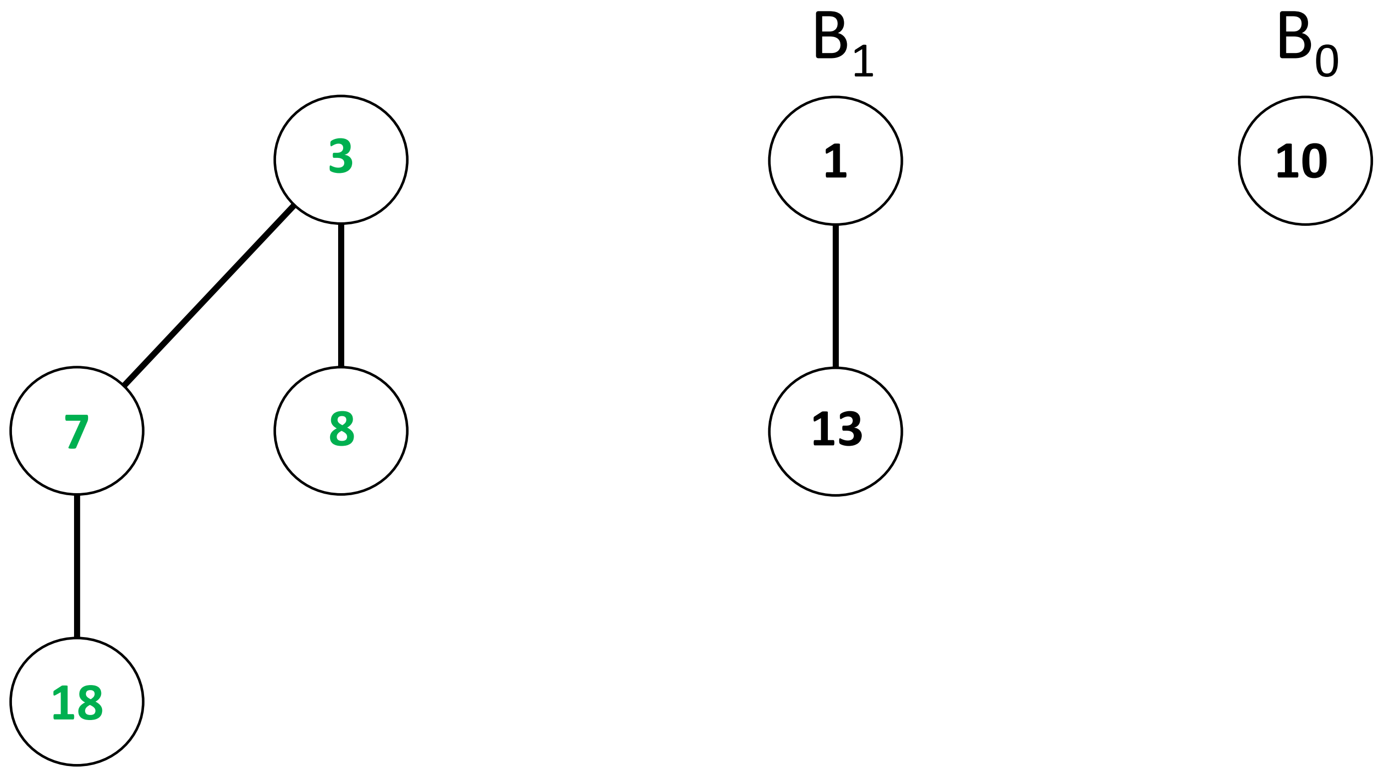
Put S in $F_7 = < B_2\ B_1\ B_0 >$



Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ $n = 7$ elements

Put S in $F_7 = < B_2\ B_1\ B_0 >$

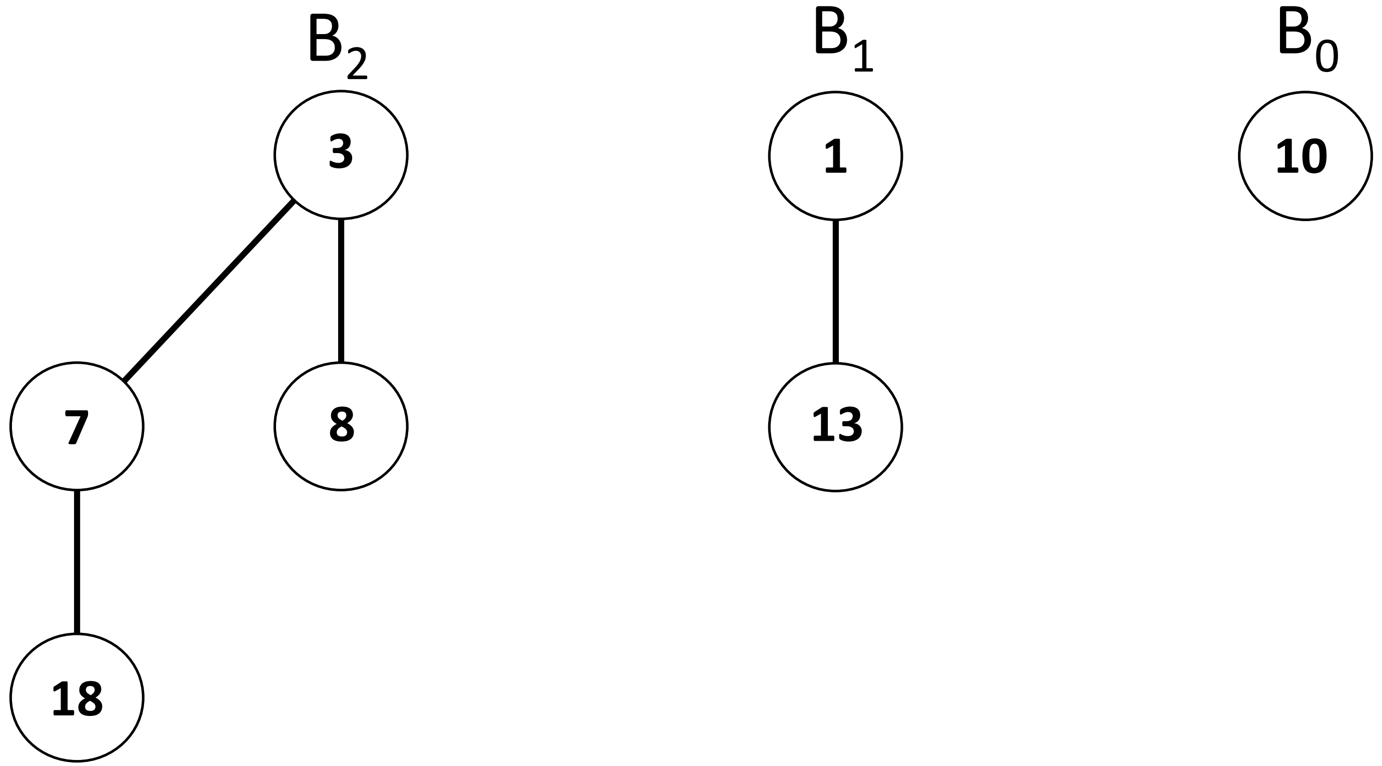


Example of a Min Binomial heap

S = {10, 13, 1, 3, 8, 18, 7}   n = 7 elements

Put S in F_7 = < B_2 B_1 B_0 >

F_7 :

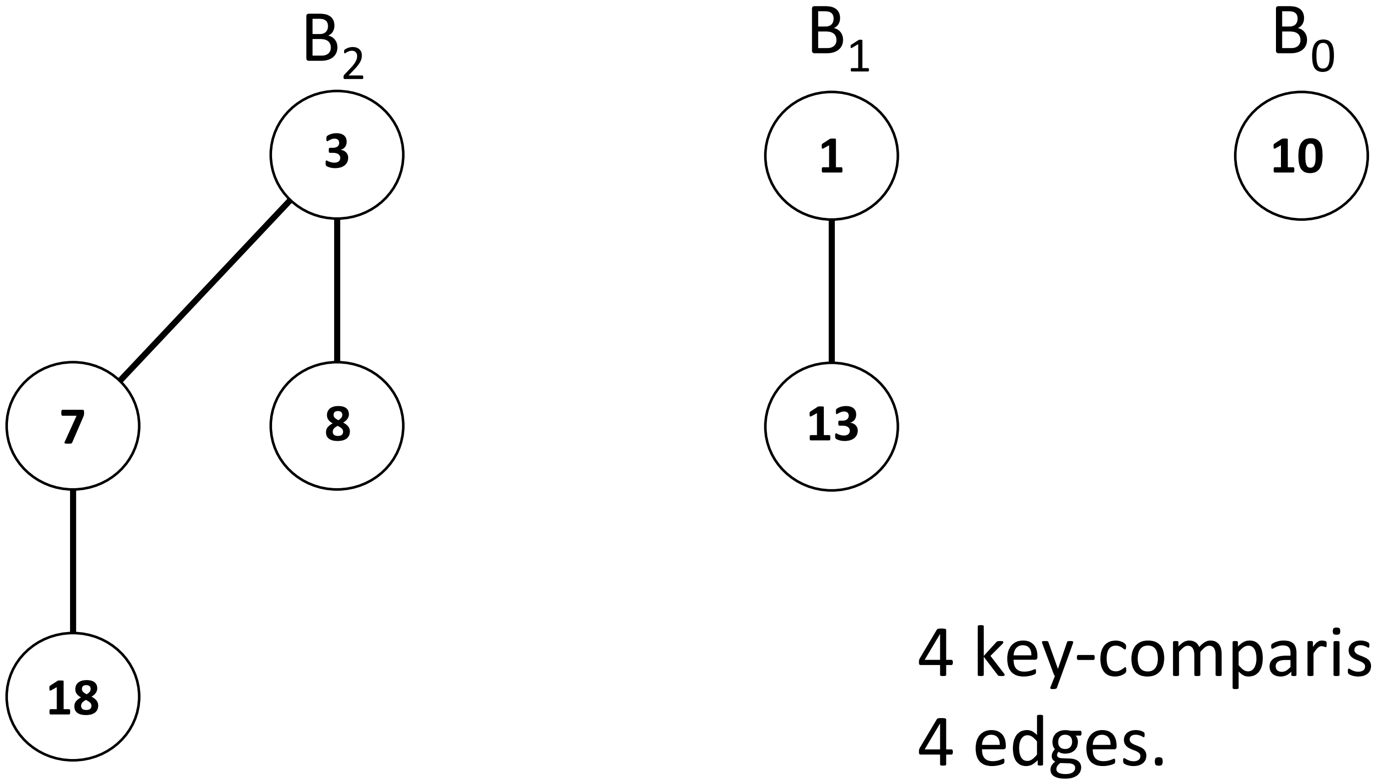


Example of a Min Binomial heap

$S = \{10, 13, 1, 3, 8, 18, 7\}$ $n = 7$ elements

Put S in $F_7 = \langle B_2\ B_1\ B_0 \rangle$

F_7 :



4 key-comparisons.
4 edges.

FACTS:

- One key-comparison per Binomial Heap edge.

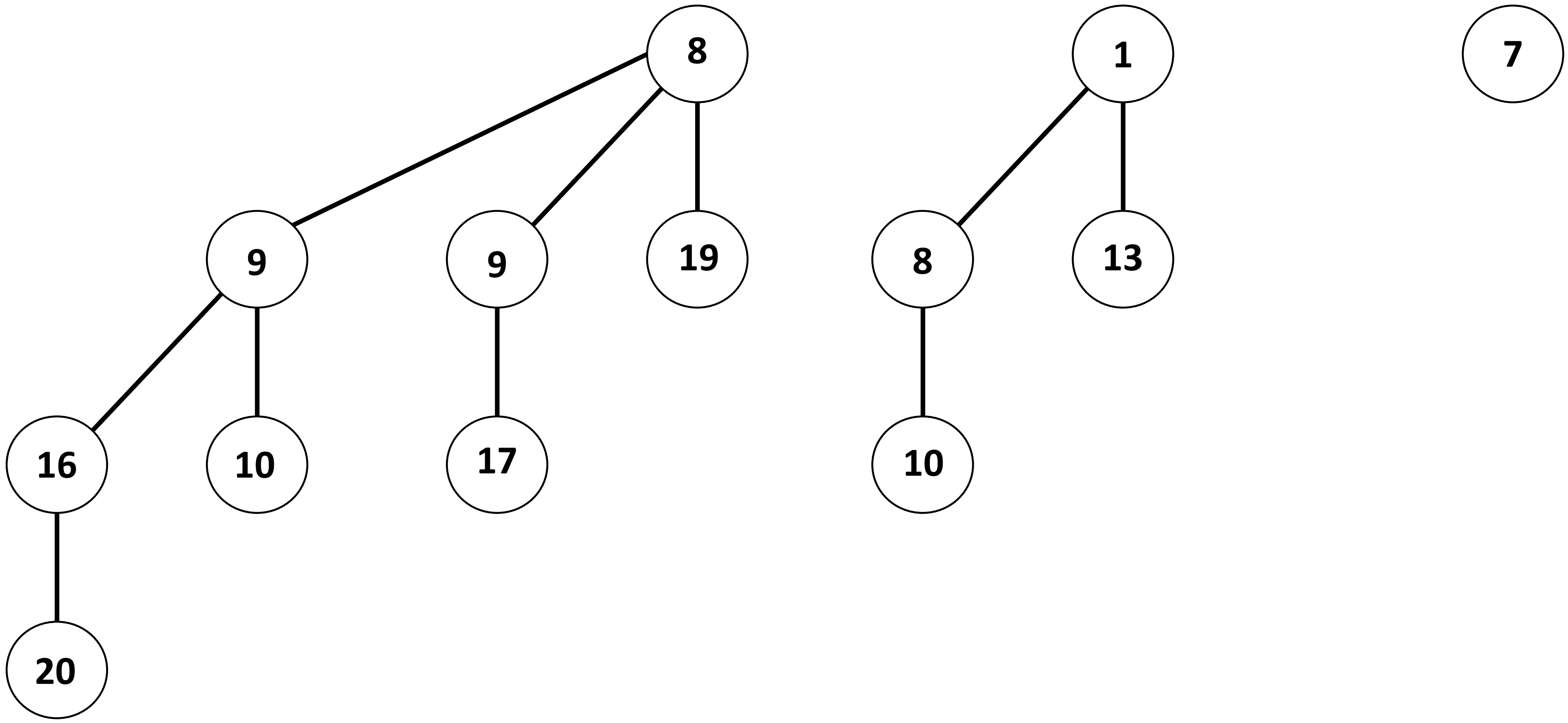
FACTS:

- One key-comparison per Binomial Heap edge.

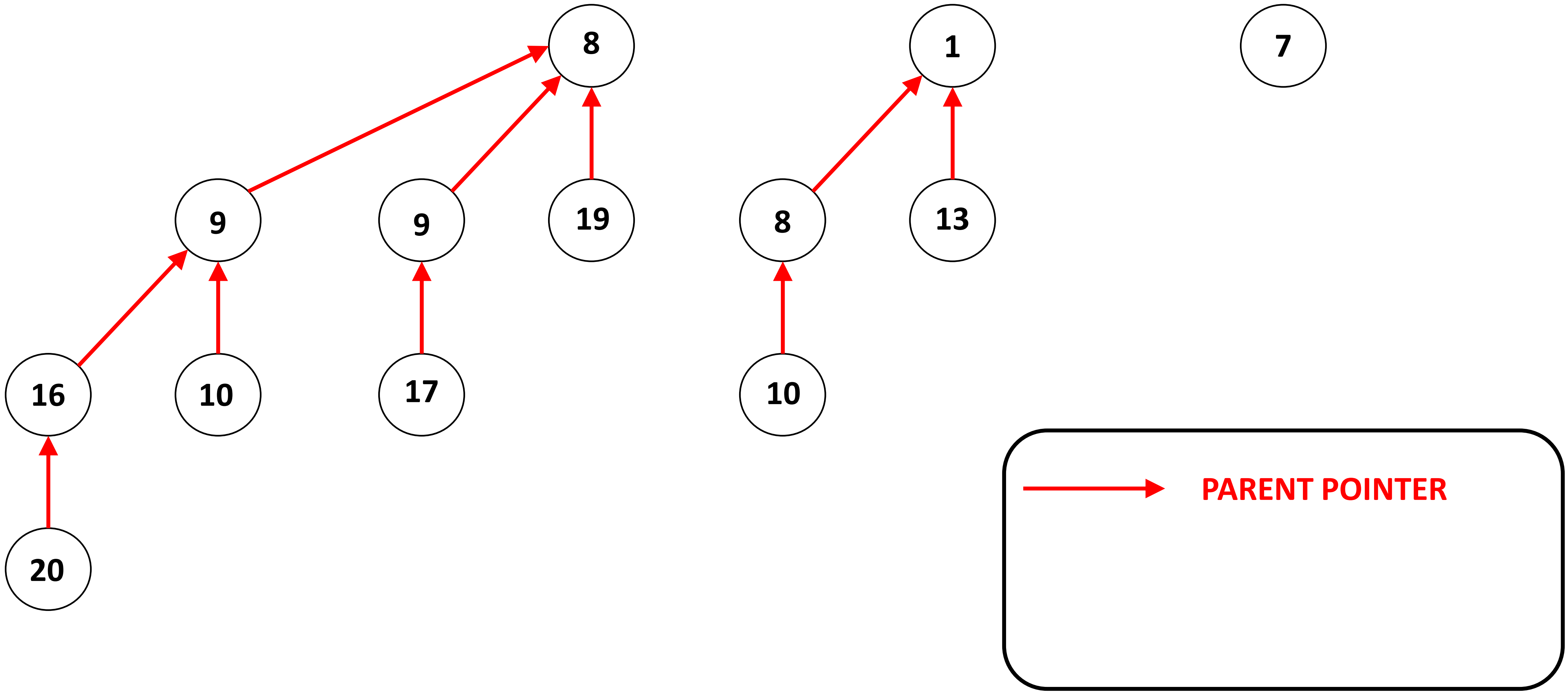
- A Binomial Heap for n elements can be built in O(n) key-comparisons.

Storing Binomial Heaps in Memory

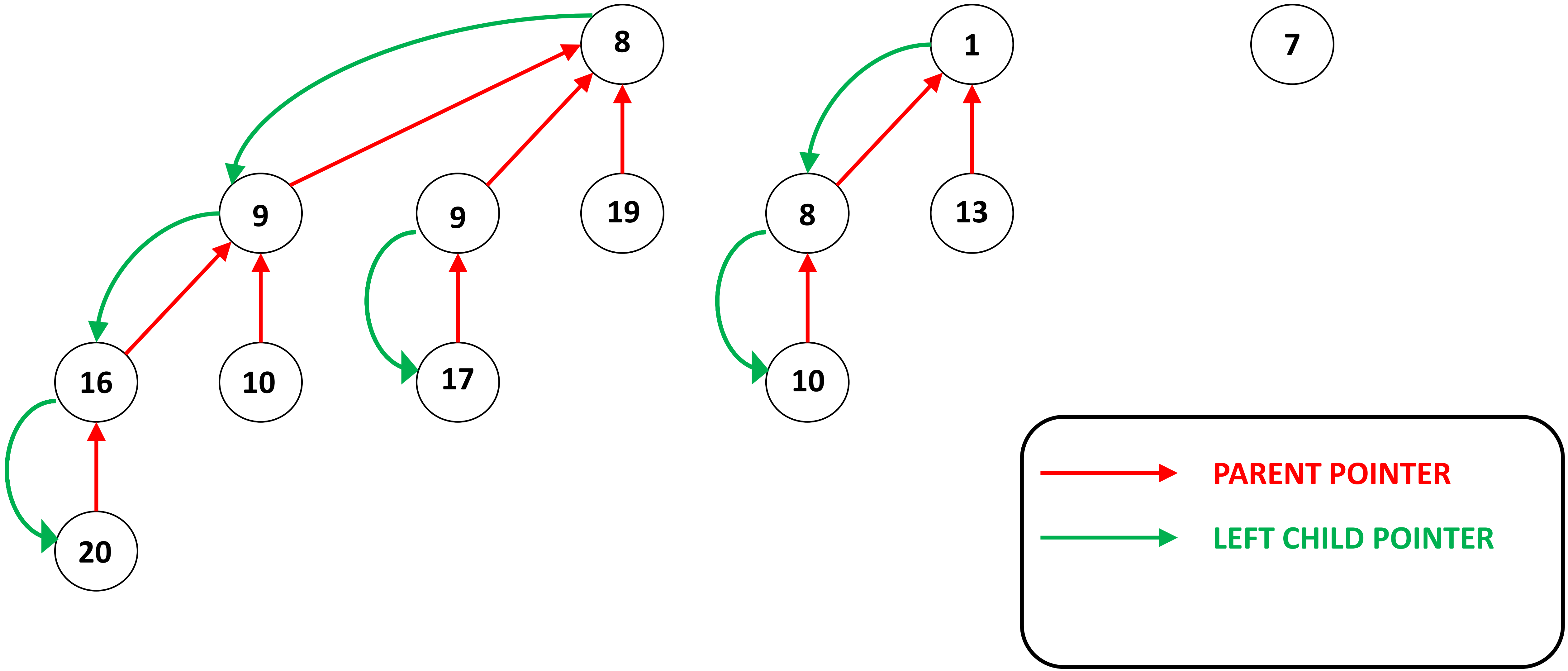
Binomial Heap (Visually)



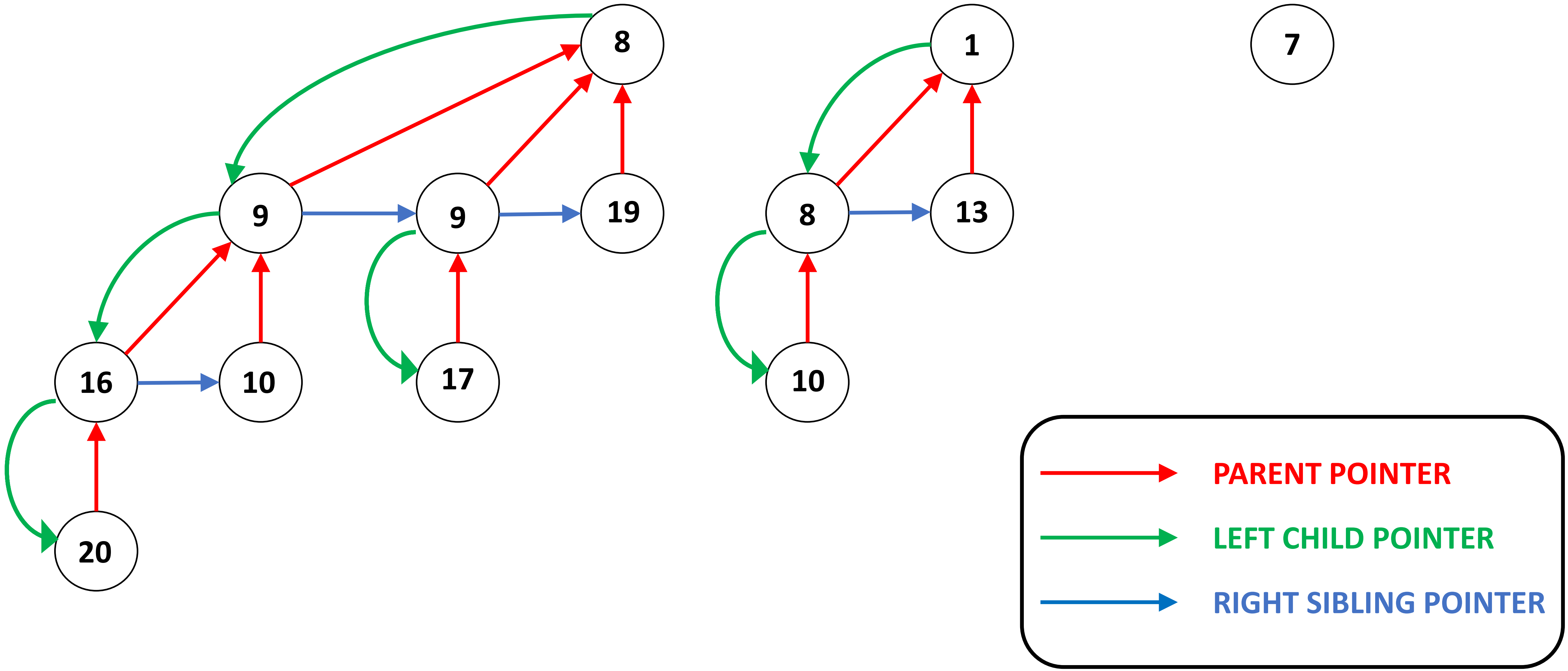
Binomial Heap (in Memory)



Binomial Heap (in Memory)

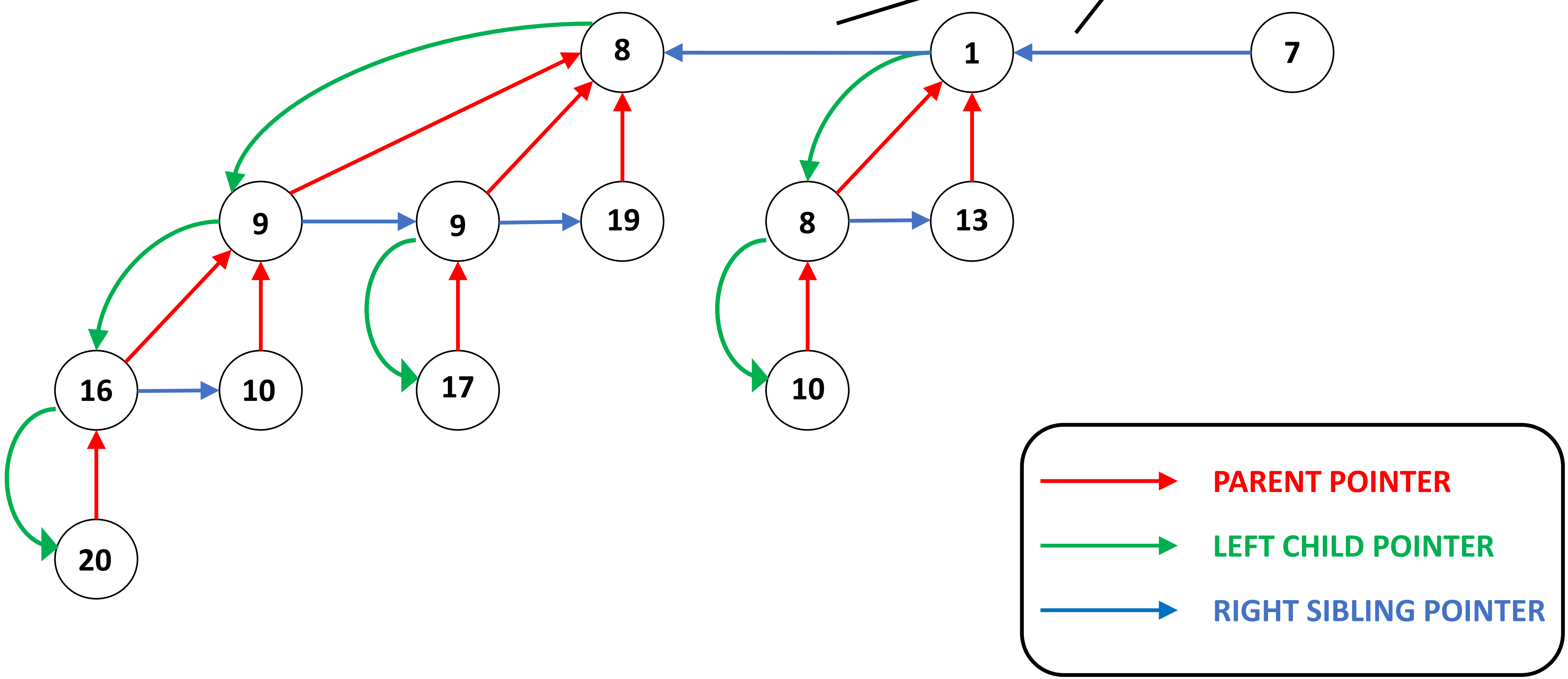


Binomial Heap (in Memory)

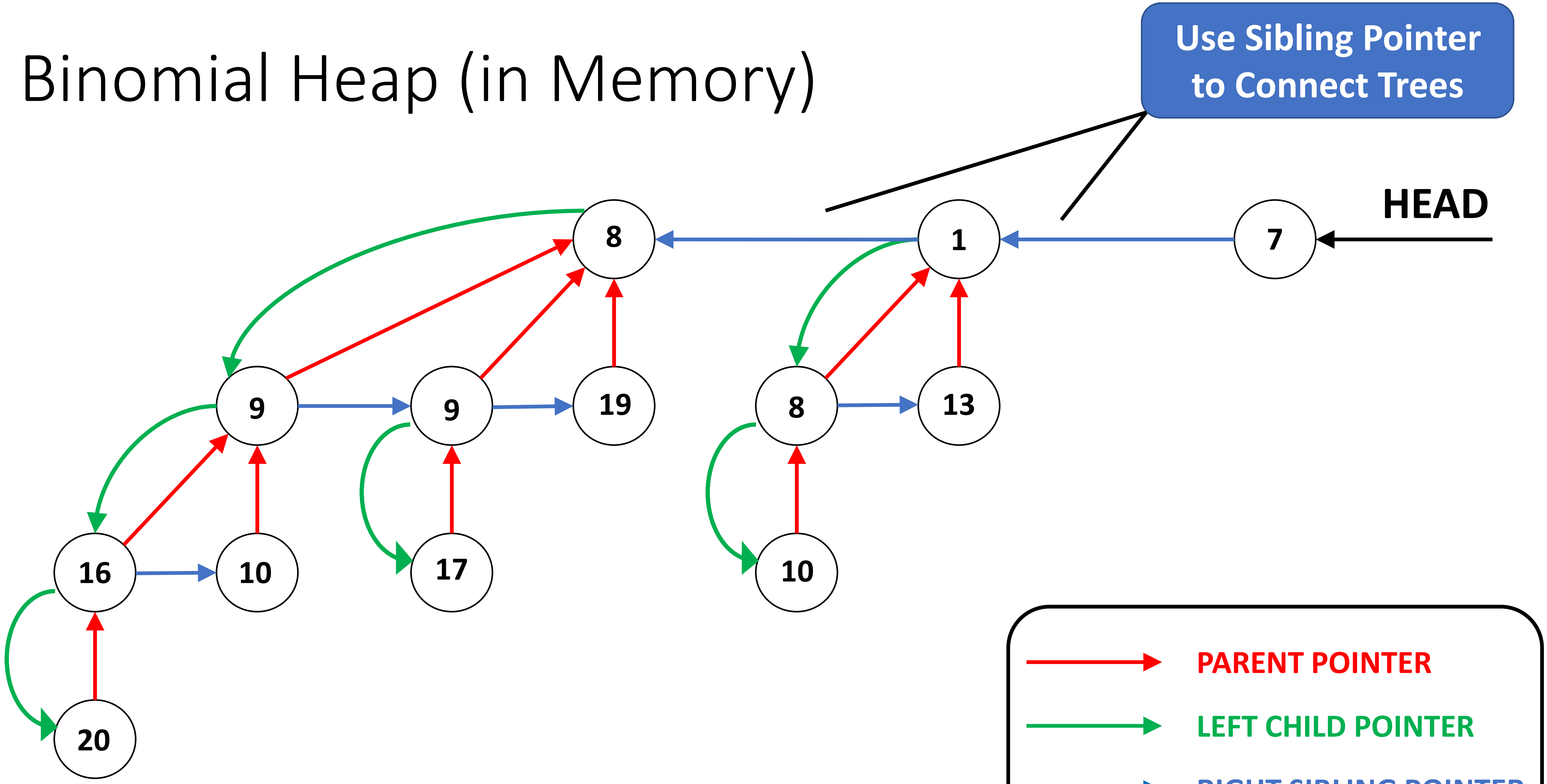


Binomial Heap (in Memory)

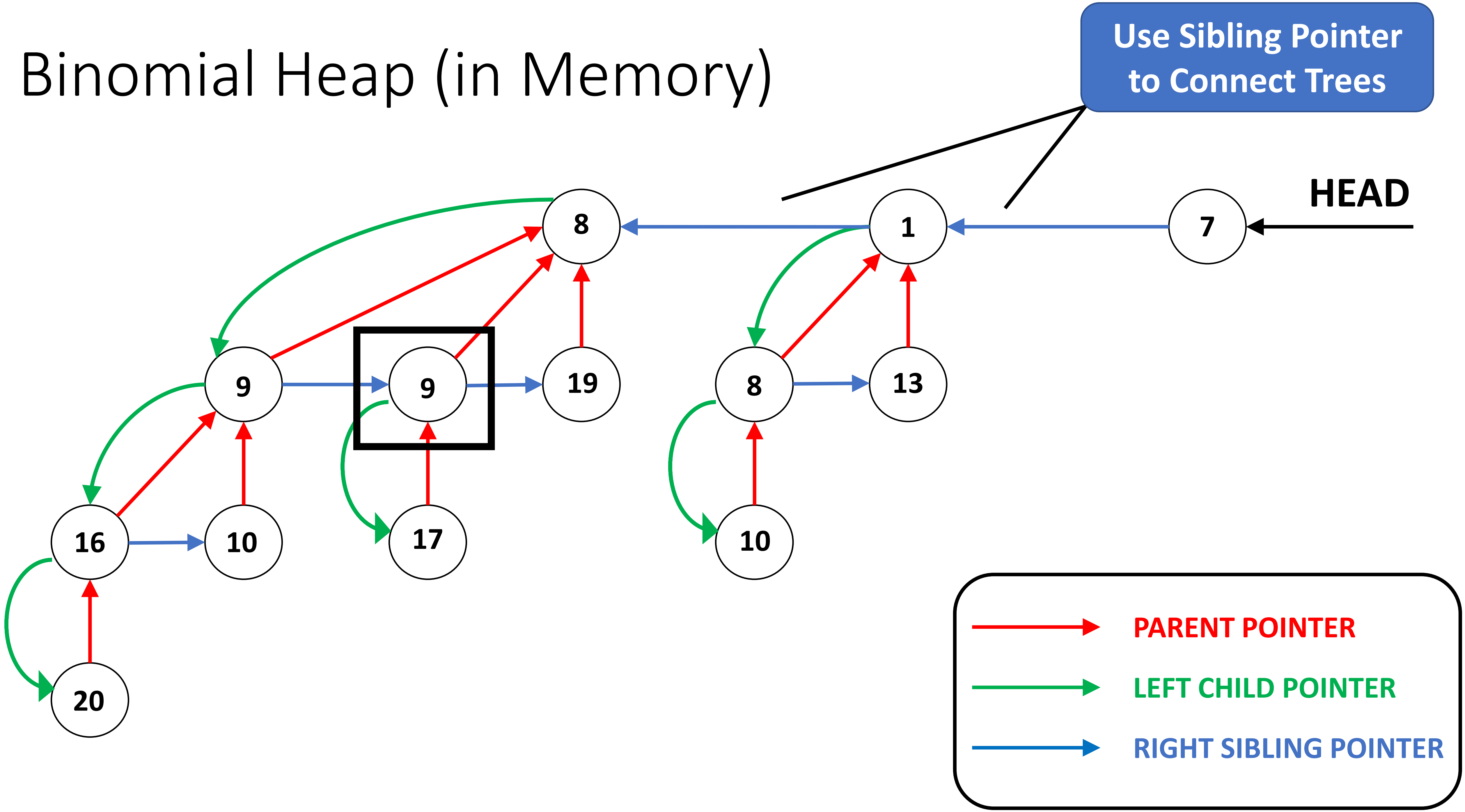
Use Sibling Pointer to Connect Trees



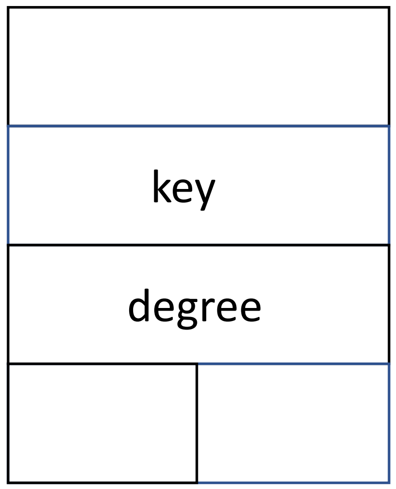
Binomial Heap (in Memory)



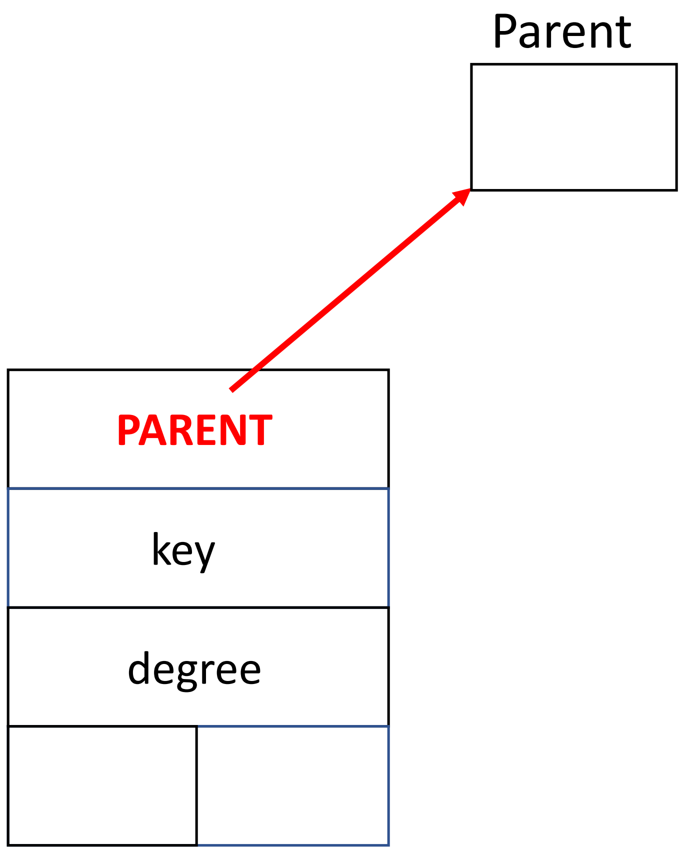
Binomial Heap (in Memory)



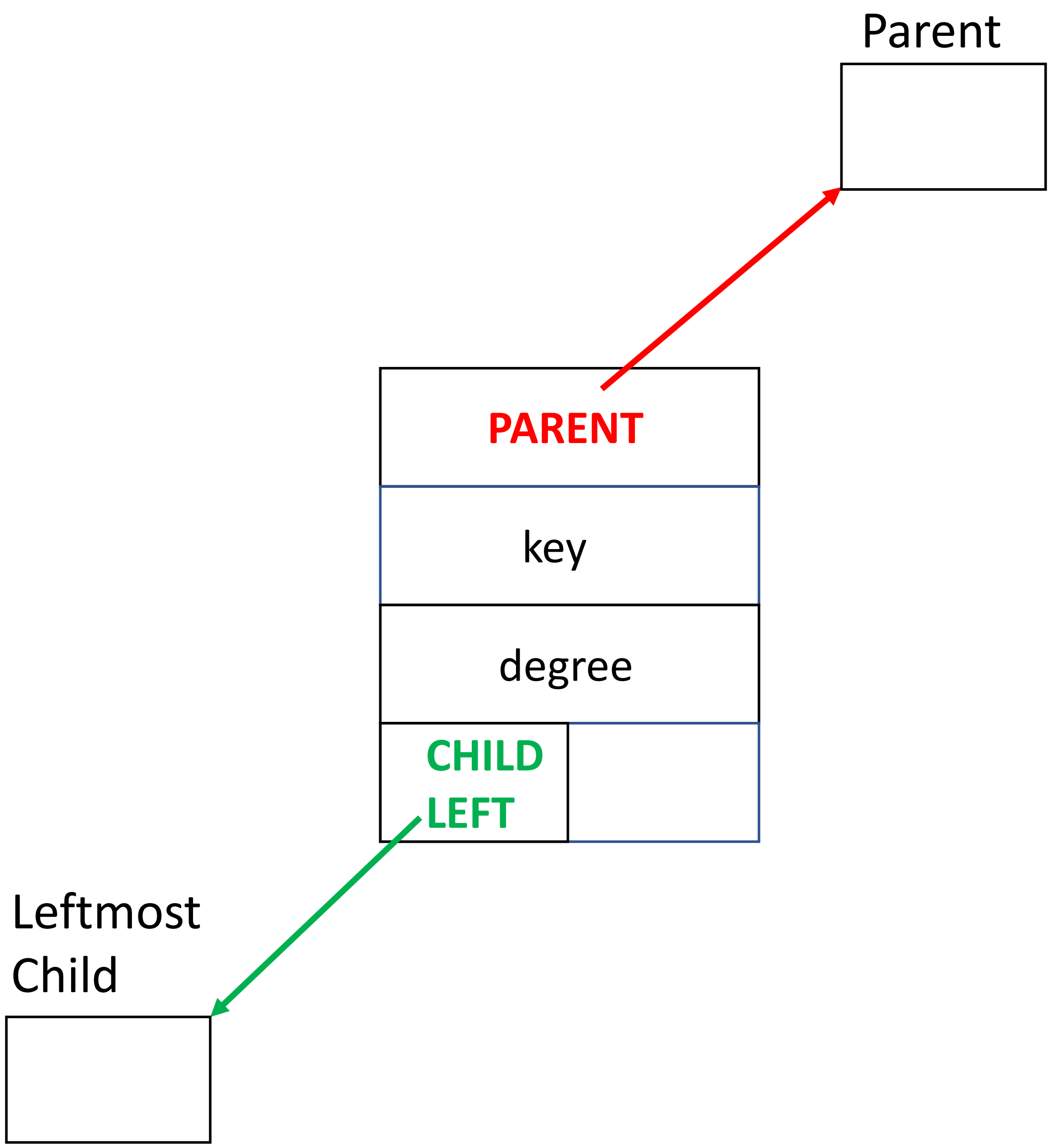
Binomial Heap (in Memory)



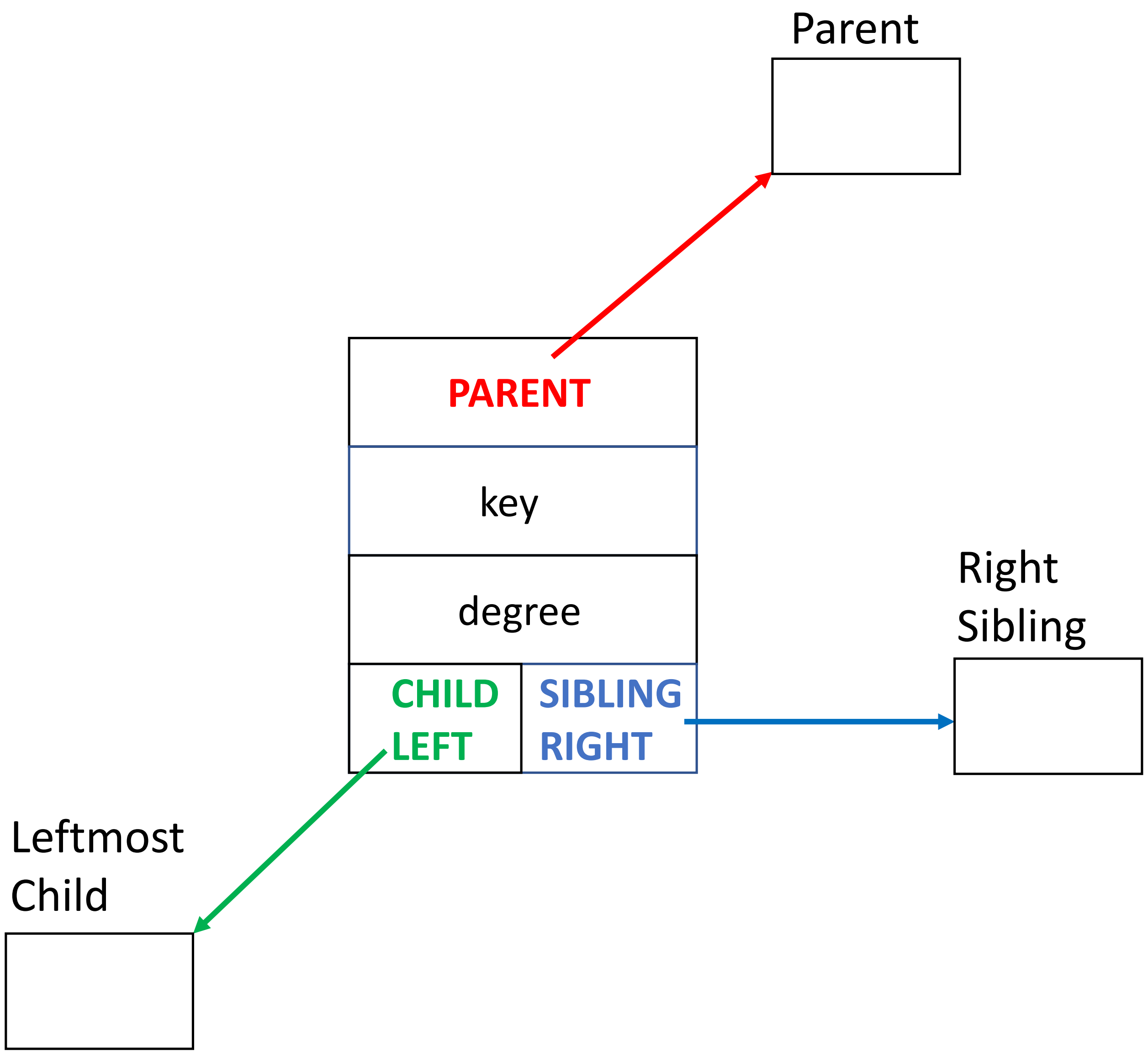
Binomial Heap (in Memory)



Binomial Heap (in Memory)

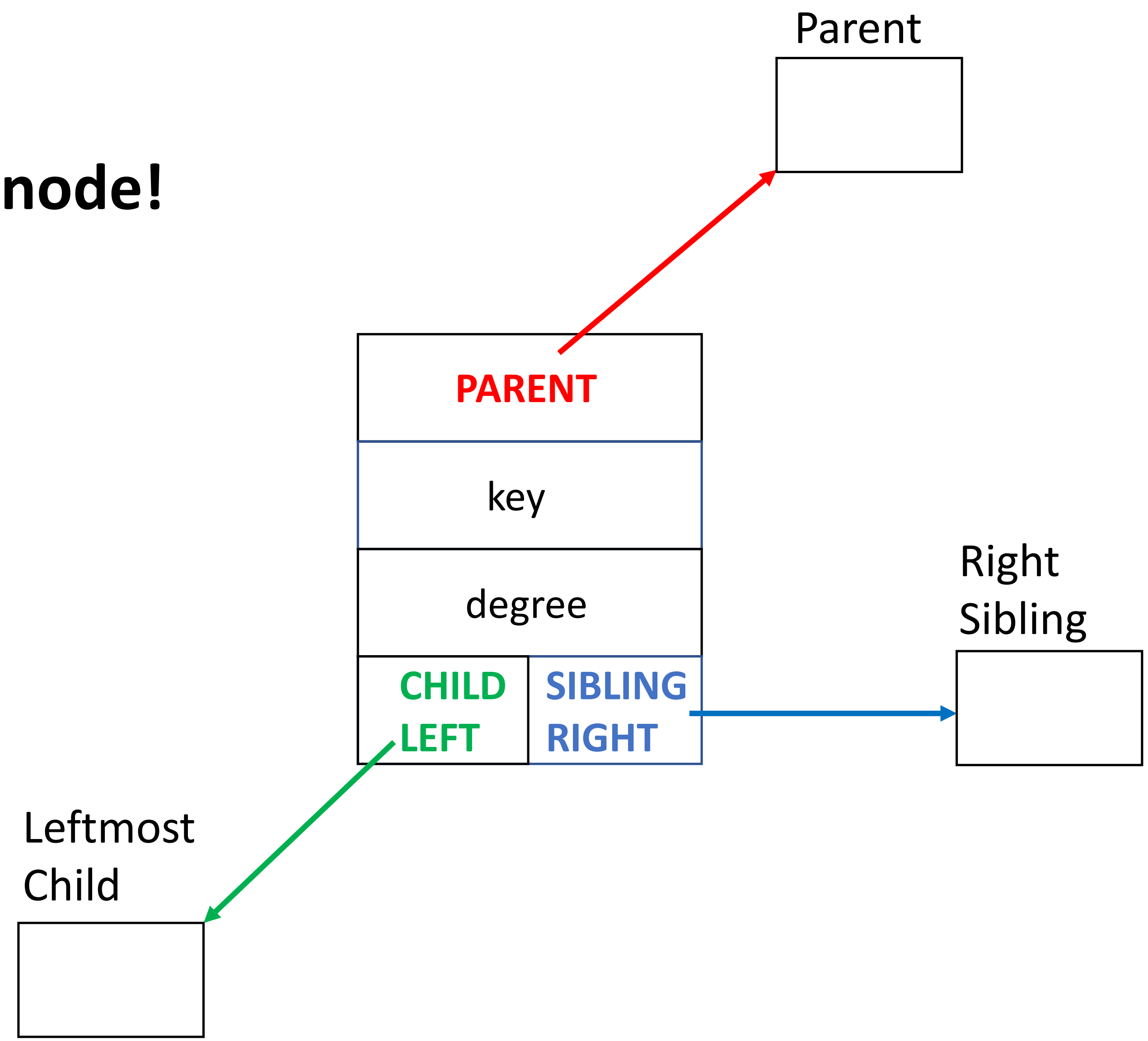


Binomial Heap (in Memory)

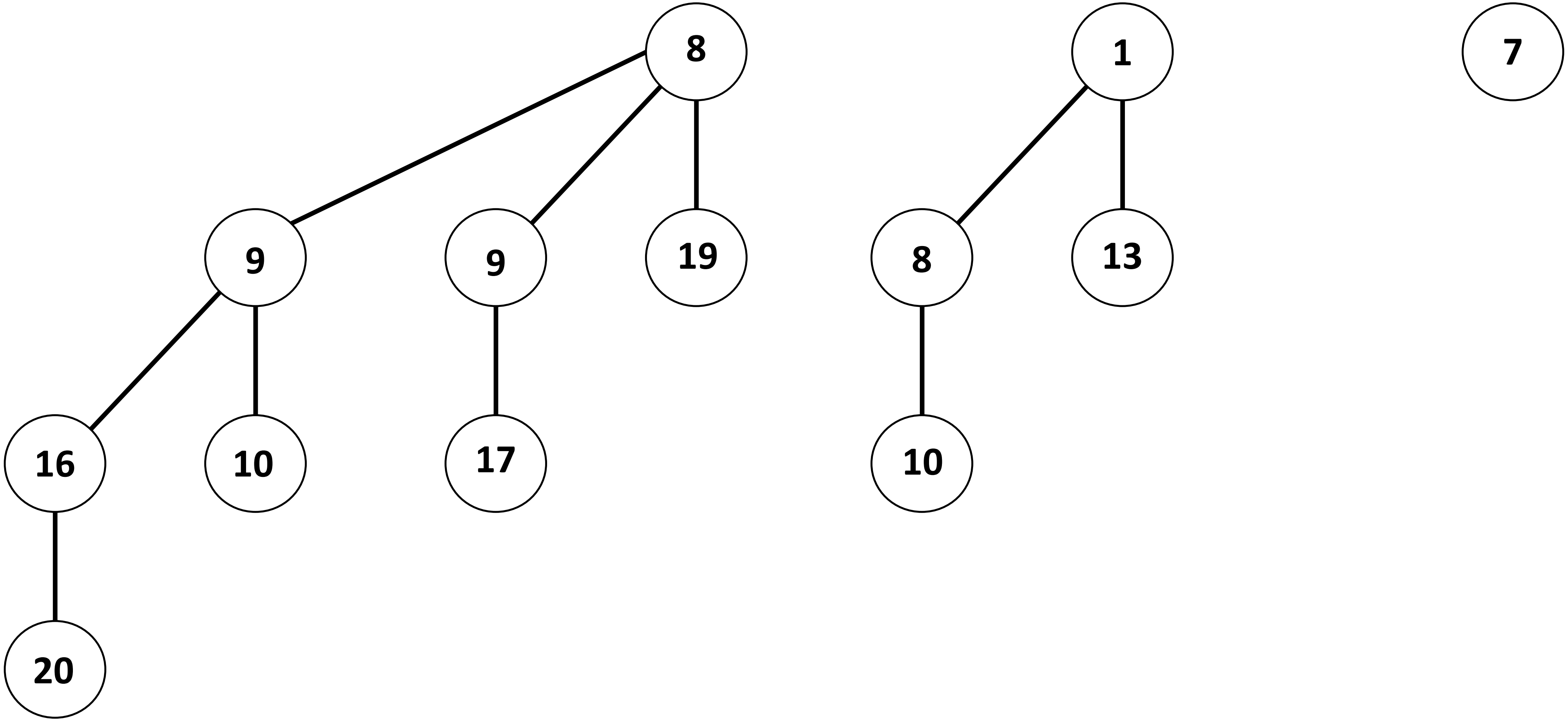


Binomial Heap (in Memory)

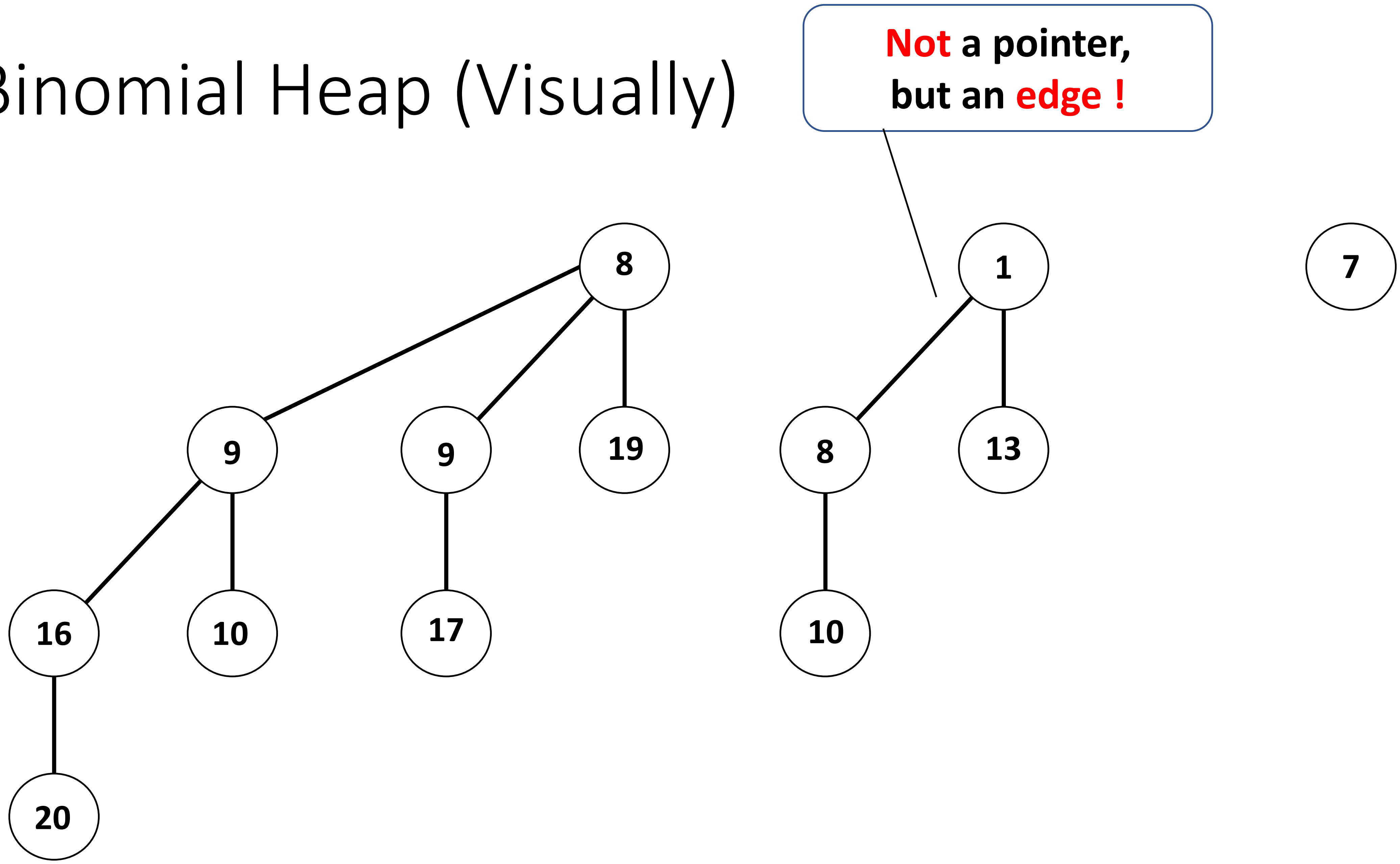
3 pointers per node!



Binomial Heap (Visually)



Binomial Heap (Visually)



Binomial Heap Operations

Must implement the following operations:

- **Union(T, Q)**
- **Insert(T, x)**
- **Min(T)**
- **Extract_Min(T)**

Must implement the following operations:

- **Union(T, Q)**
- **Insert(T, x)**
- **Min(T)**
- **Extract_Min(T)**

High Level Ideas

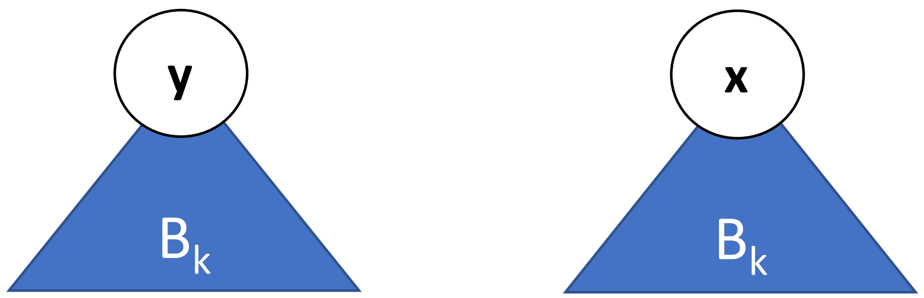
High Level Ideas

Lemma 1: Can merge two min heap-ordered B_k trees into a single min heap-ordered B_{k+1} tree with just one key comparison.

High Level Ideas

Lemma 1: Can merge two min heap-ordered B_k trees into a single min heap-ordered B_{k+1} tree with just one key comparison.

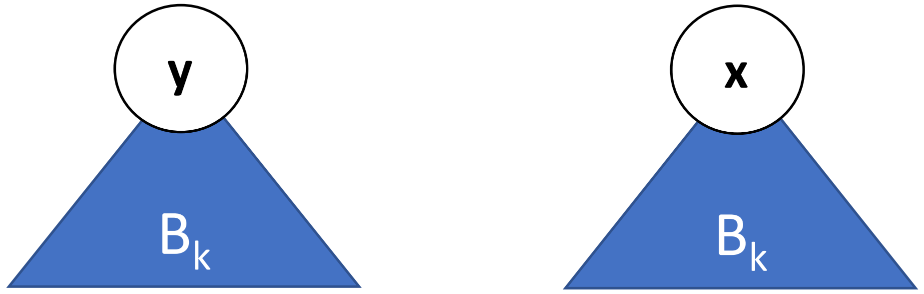
Proof: To merge



High Level Ideas

Lemma 1: Can merge two min heap-ordered B_k trees into a single min heap-ordered B_{k+1} tree with just one key comparison.

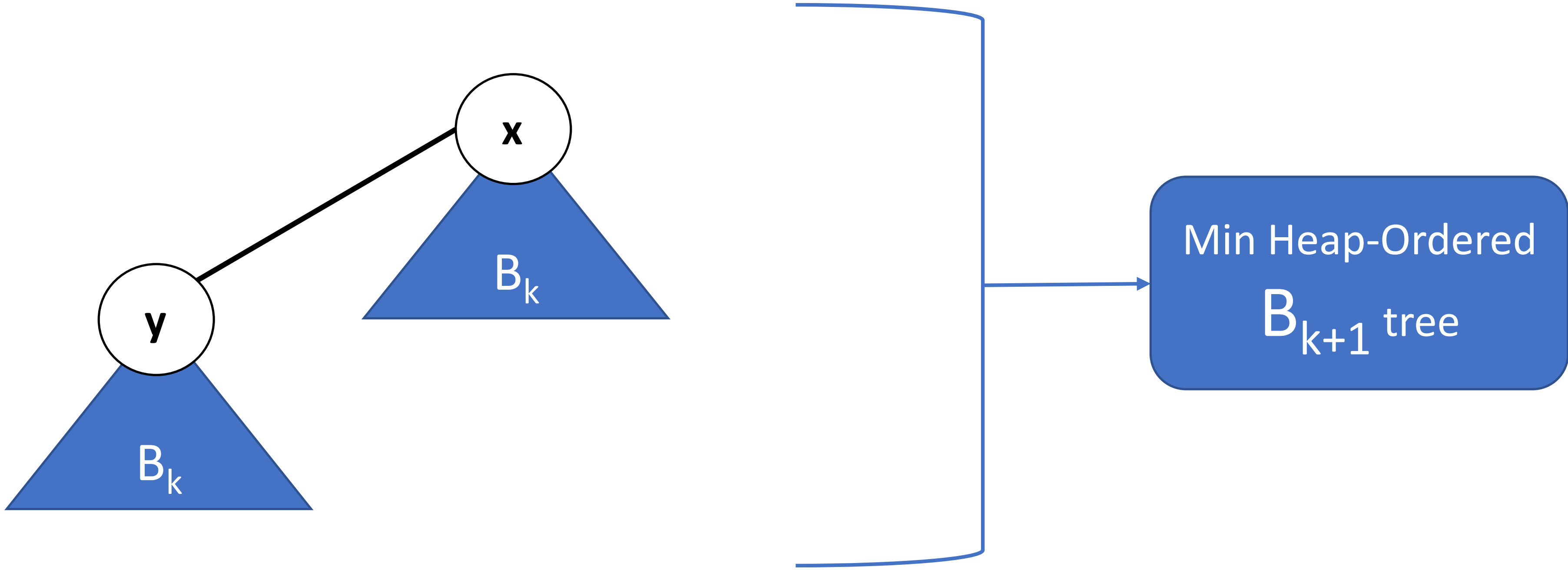
Proof: To merge, if x <= y



High Level Ideas

Lemma 1: Can merge two min heap-ordered B_k trees into a single min heap-ordered B_{k+1} tree with just one key comparison.

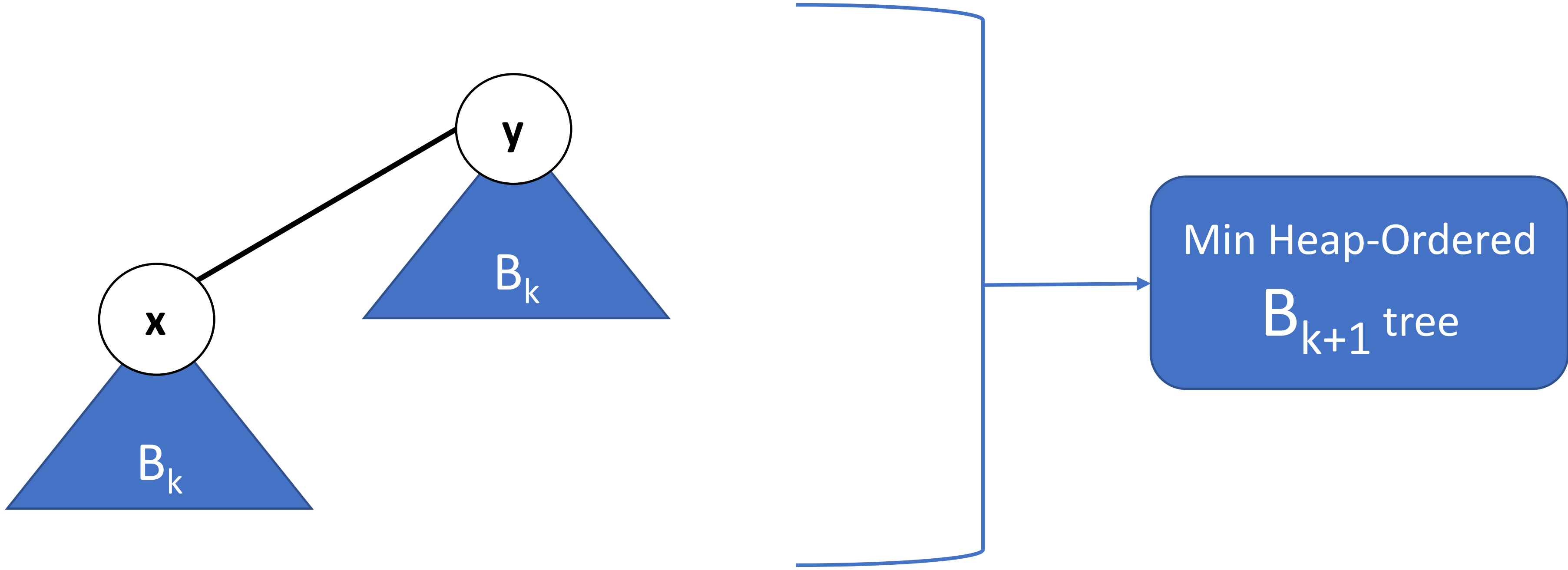
Proof: To merge, if x <= y



High Level Ideas

Lemma 1: Can merge two min heap-ordered B_k trees into a single min heap-ordered B_{k+1} tree with just one key comparison.

Proof: To merge, if y < x

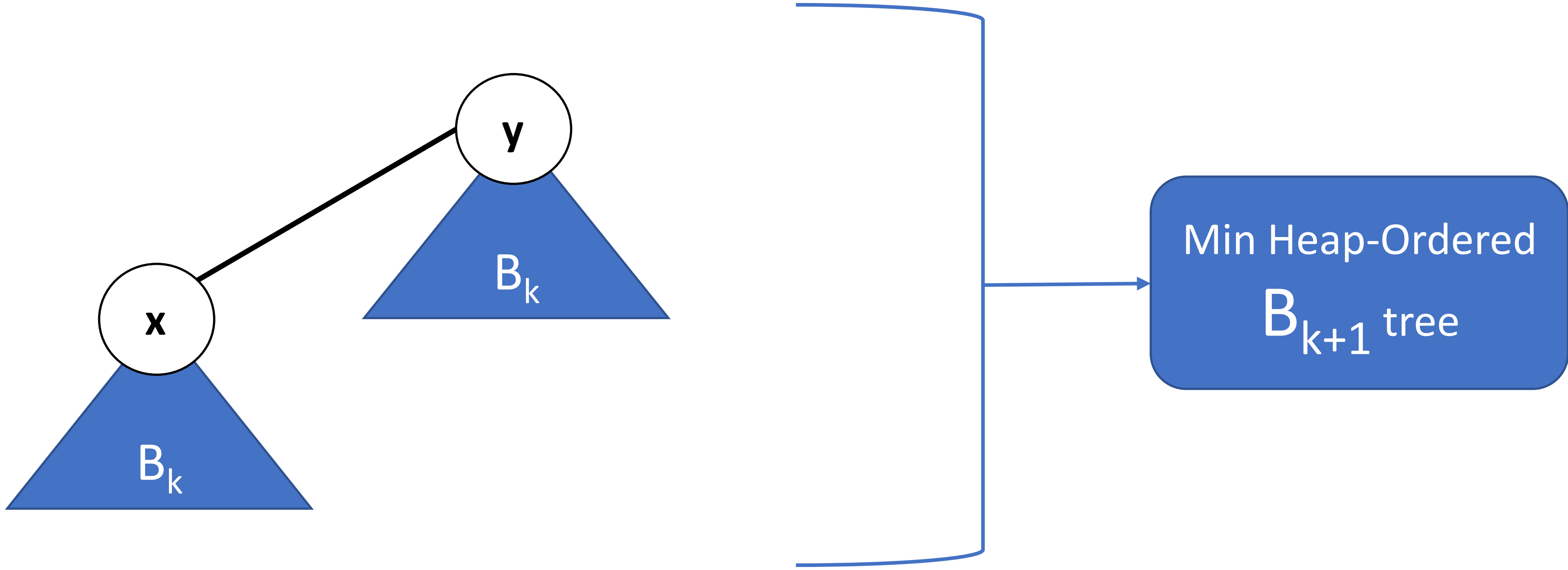


High Level Ideas

Cannot merge two max-heaps So easily !

Lemma 1: Can merge two min heap-ordered B_k trees into a single min heap-ordered B_{k+1} tree with just one key comparison.

Proof: To merge, if y < x



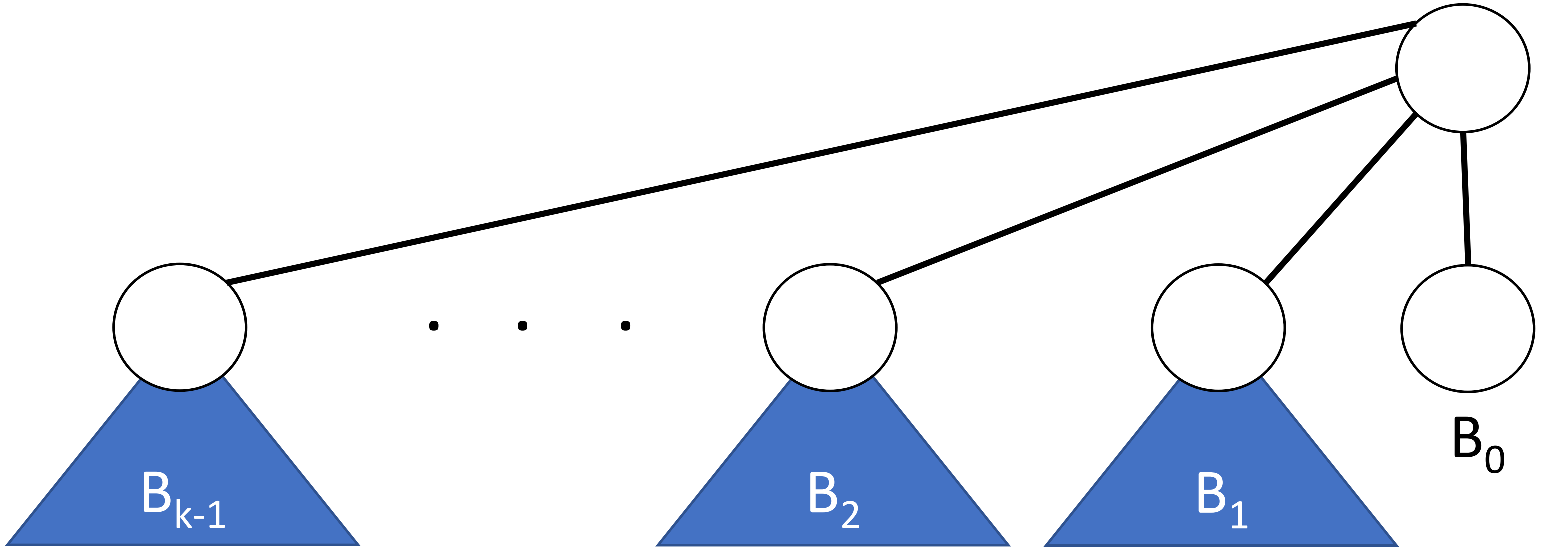
High Level Ideas

Lemma 2: Deleting the root of a min heap-ordered B_k tree gives a min binomial heap.

High Level Ideas

Lemma 2: Deleting the root of a min heap-ordered B_k tree gives a min binomial heap.

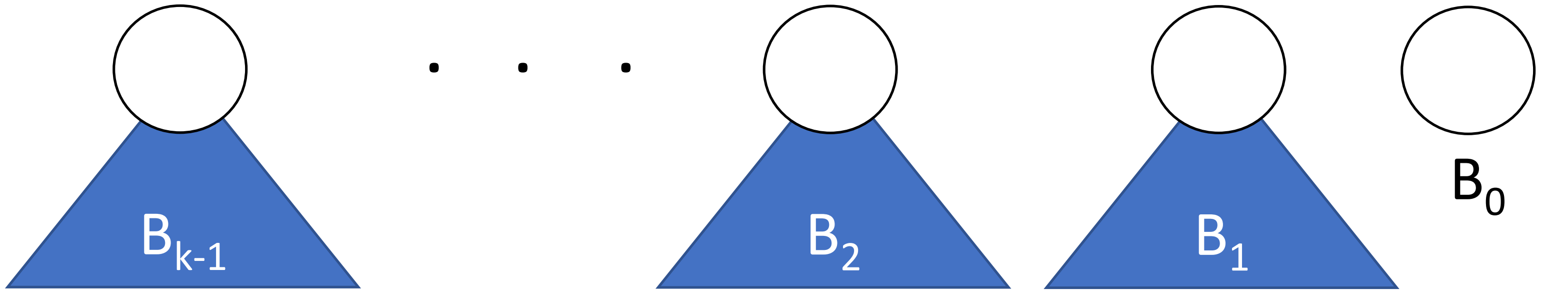
Proof: Deleting the root of min heap-ordered B_k tree.



High Level Ideas

Lemma 2: Deleting the root of a min heap-ordered B_k tree gives a min binomial heap.

Proof: Deleting the root of min heap-ordered B_k tree.



Union(T, Q)

Union(T, Q)

T is a Binomial Heap of size $n = 3 = \langle \; 1\;1 \rangle_2$

1	1	B_1	B_0	:	T

Union(T, Q)

T is a Binomial Heap of size n = 3 = $< \ \ 1\ 1>_2$

Q is a Binomial Heap of size n = 7 = $< 1\ 1\ 1>_2$

	1	1		B_1	B_0	:	T
1	1	1	B_2	B_1	B_0	:	Q

Union(T, Q)

T is a Binomial Heap of size n = 3 = <   1 1>$_2$

Q is a Binomial Heap of size n = 7 = < 1 1 1>$_2$

	1	1	+		B_1	B_0	: T	+
1	1	1		B_2	B_1	B_0	: Q	

Union(T, Q)

T is a Binomial Heap of size $n = 3 = <\ \ 1\ 1>_2$

Q is a Binomial Heap of size $n = 7 = <1\ 1\ 1>_2$

	1					B_1		**Carry**
	1	1	**+**			B_1	B_0	: T **+**
1	1	1			B_2	B_1	B_0	: Q
		0					X	**Result**

Union(T, Q)

T is a Binomial Heap of size $n = 3 = <\ \ 1\ 1>_2$

Q is a Binomial Heap of size $n = 7 = < 1\ 1\ 1>_2$

1	1			B_2	B_1		**Carry**	
	1	1			B_1	B_0	: T	
			+					**+**
1	1	1		B_2	B_1	B_0	: Q	
	1	0			B_1	X	**Result**	

Union(T, Q)

T is a Binomial Heap of size $n = 3 = <\ \ 1\ 1>_2$

Q is a Binomial Heap of size $n = 7 = < 1\ 1\ 1>_2$

1	1	1			B_3	B_2	B_1		**Carry**
		1	1				B_1	B_0	: T
				+					**+**
	1	1	1			B_2	B_1	B_0	: Q
1	0	1	0		B_3	X	B_1	X	**Result**

Union(T, Q)

T is a Binomial Heap of size $n = 3 = < \quad 1\ 1>_2$

Q is a Binomial Heap of size $n = 7 = < 1\ 1\ 1>_2$

1	1	1			B_3	B_2	B_1		**Carry**
		1	1				B_1	B_0	: T
				+					+
	1	1	1			B_2	B_1	B_0	: Q
1	0	1	0		B_3	X	B_1	X	**Result**

How many new edges were added?

Union(T, Q)

T is a Binomial Heap of size n = 3 = < 1 1>$_2$

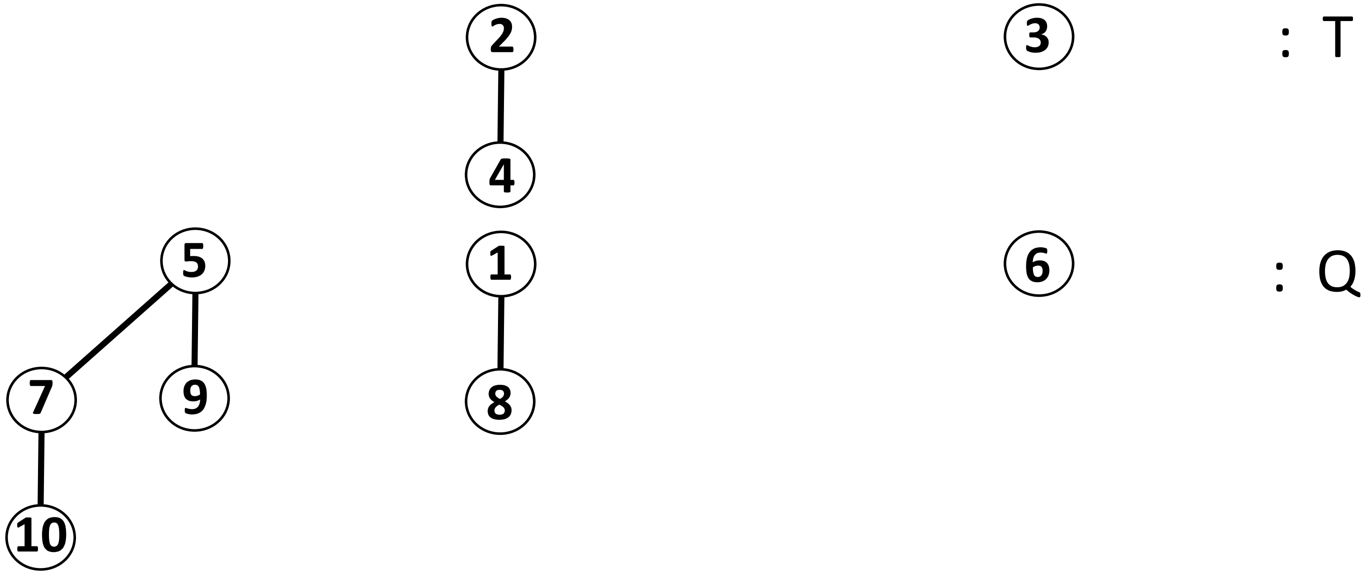
Q is a Binomial Heap of size n = 7 = < 1 1 1>$_2$

1	1	1			B_3	B_2	B_1		**Carry**
		1	1				B_1	B_0	: T
				+					**+**
	1	1	1			B_2	B_1	B_0	: Q
1	0	1	0		B_3	X	B_1	X	**Result**

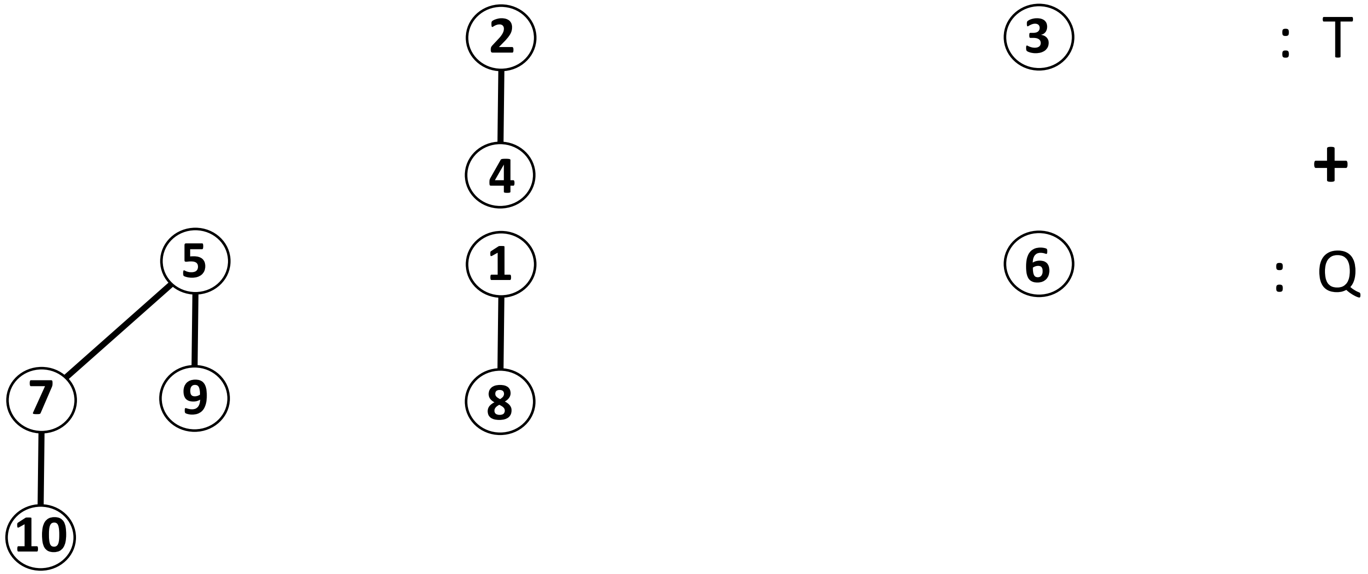
How many new edges were added?

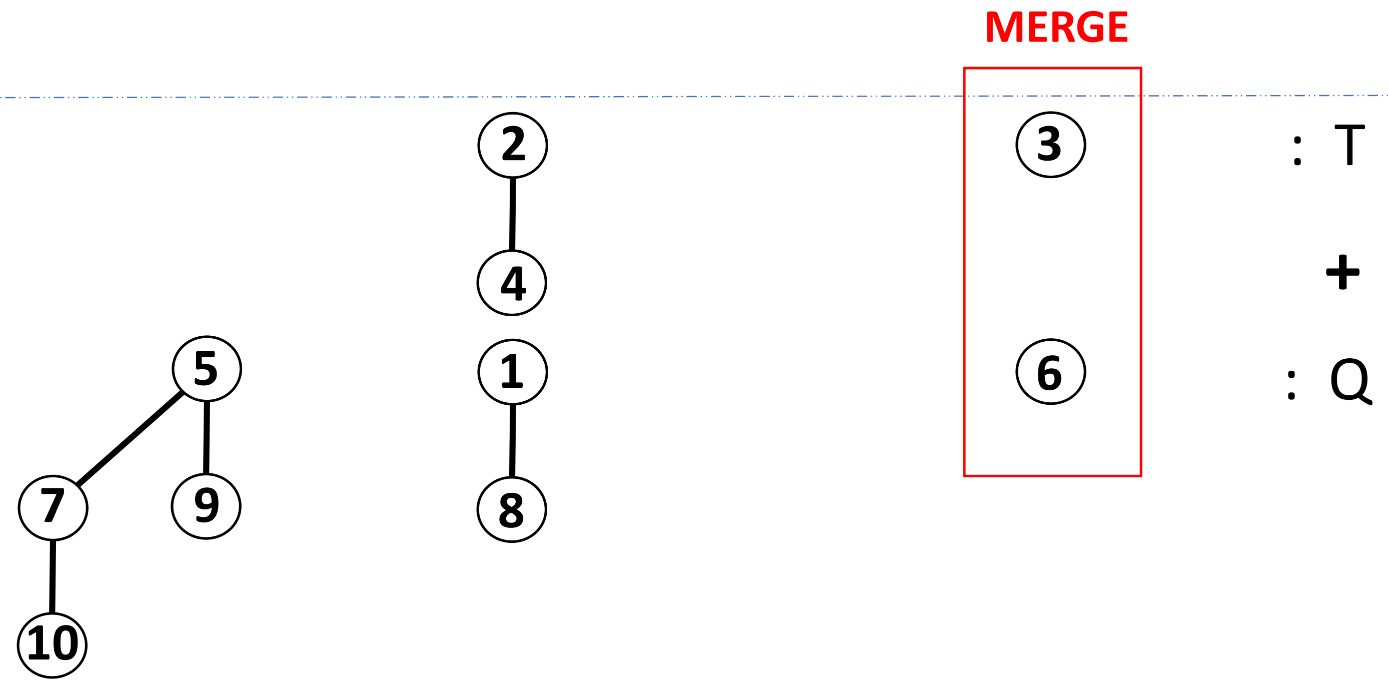
3

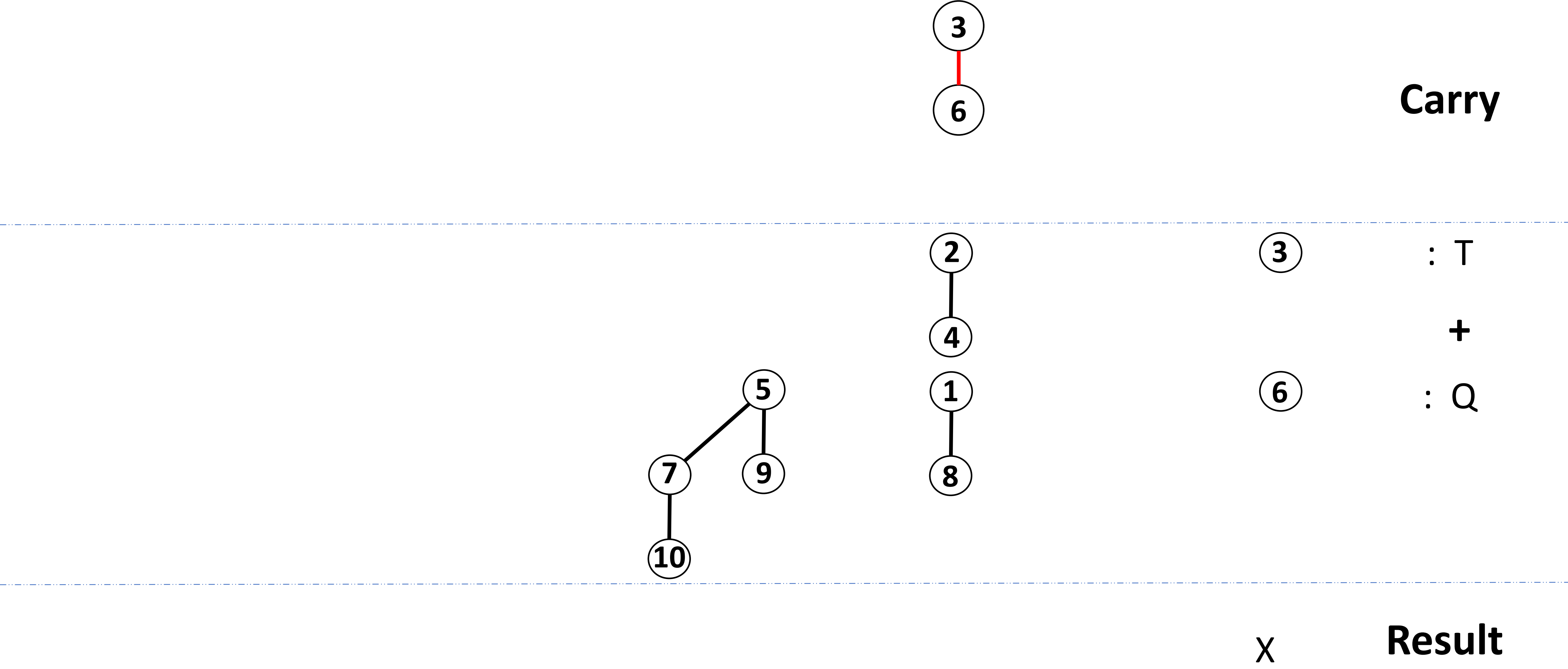
: T

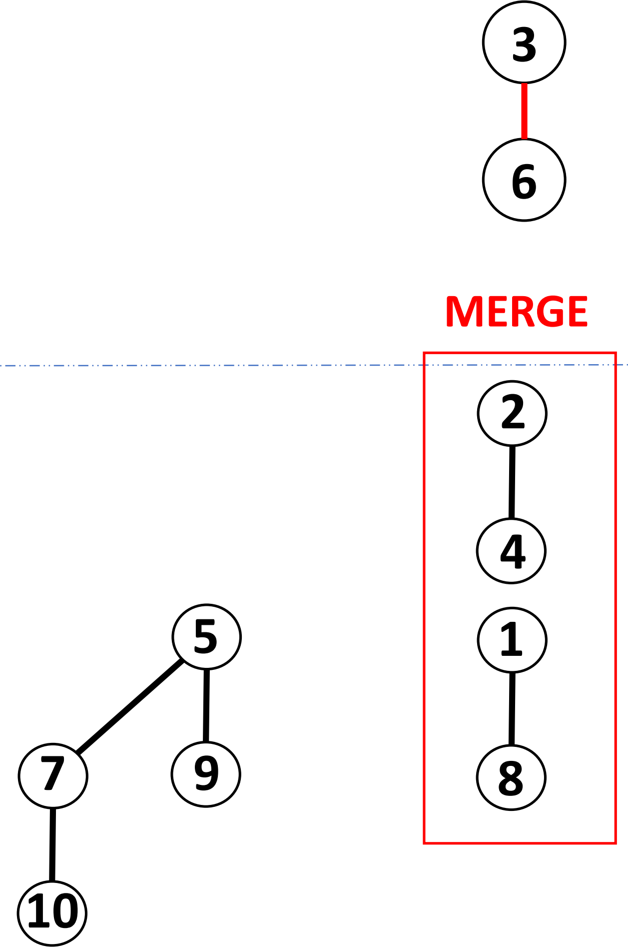


2
3
: T
4
5
1
6
: Q
7
9
8
10


: T
+
: Q


MERGE
3
6
: T
+
: Q
2
4
1
8
5
7
9
10


Carry
3 : T
+
6 : Q
X
Result



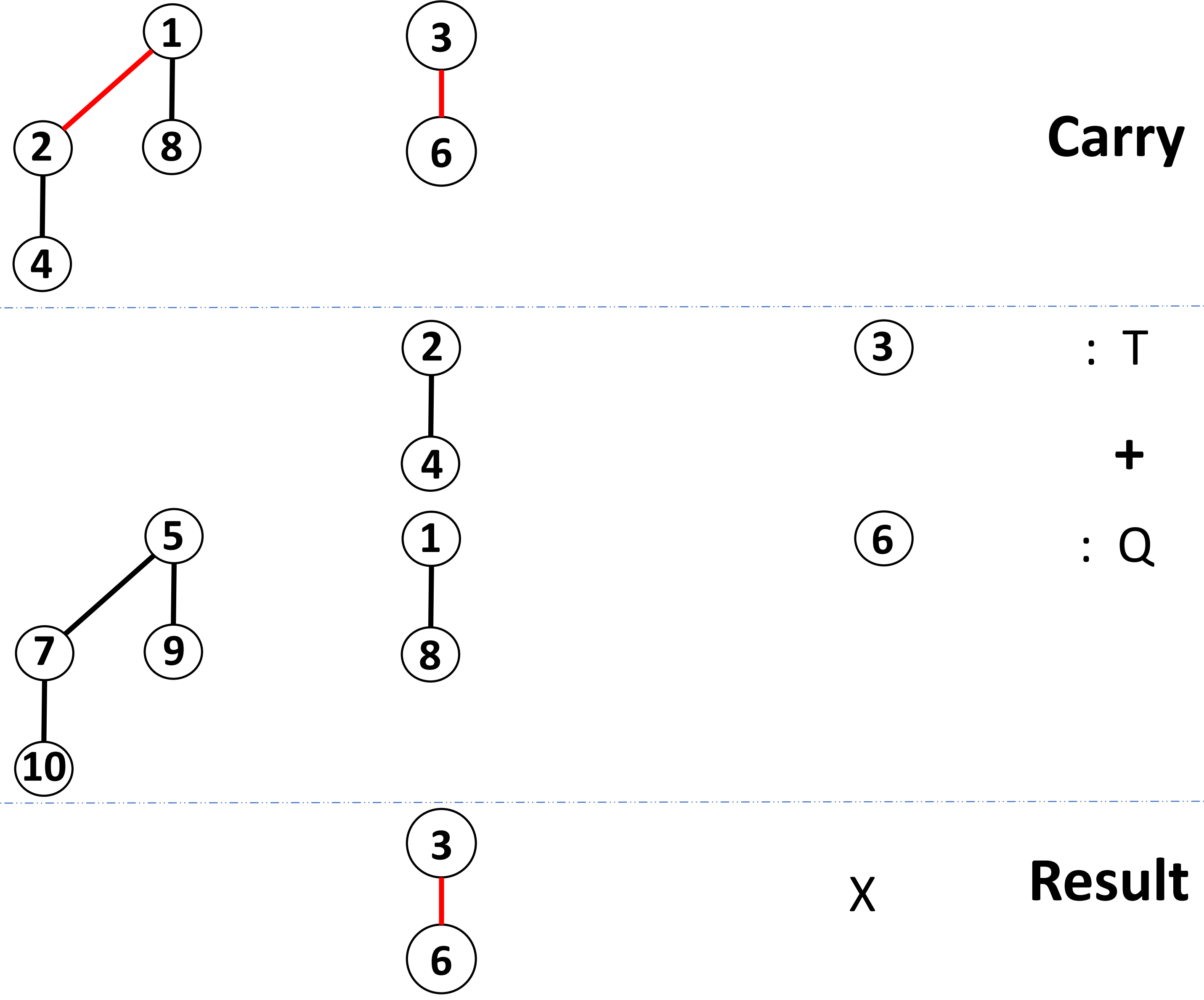
Carry

3 : T

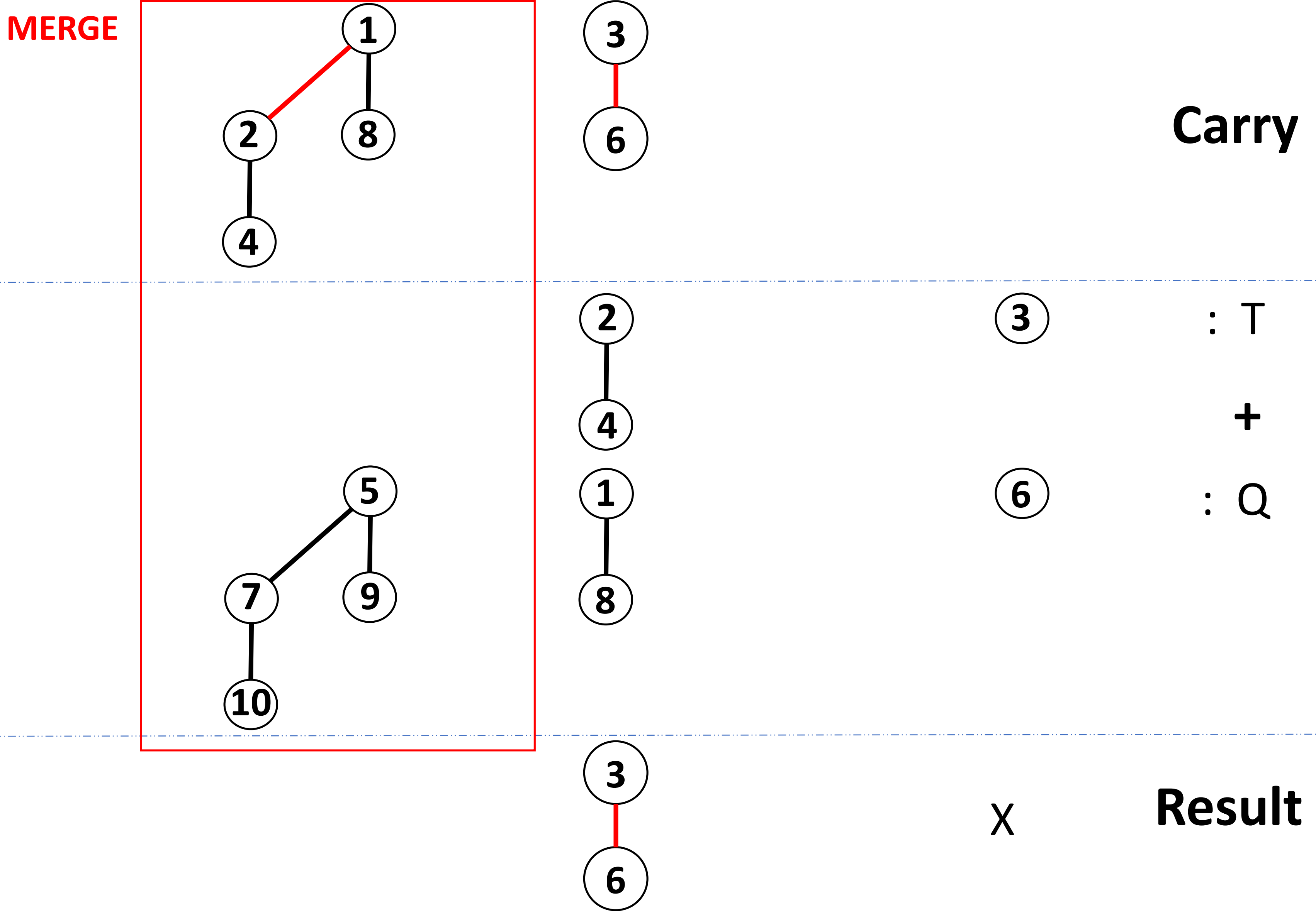
+

6 : Q

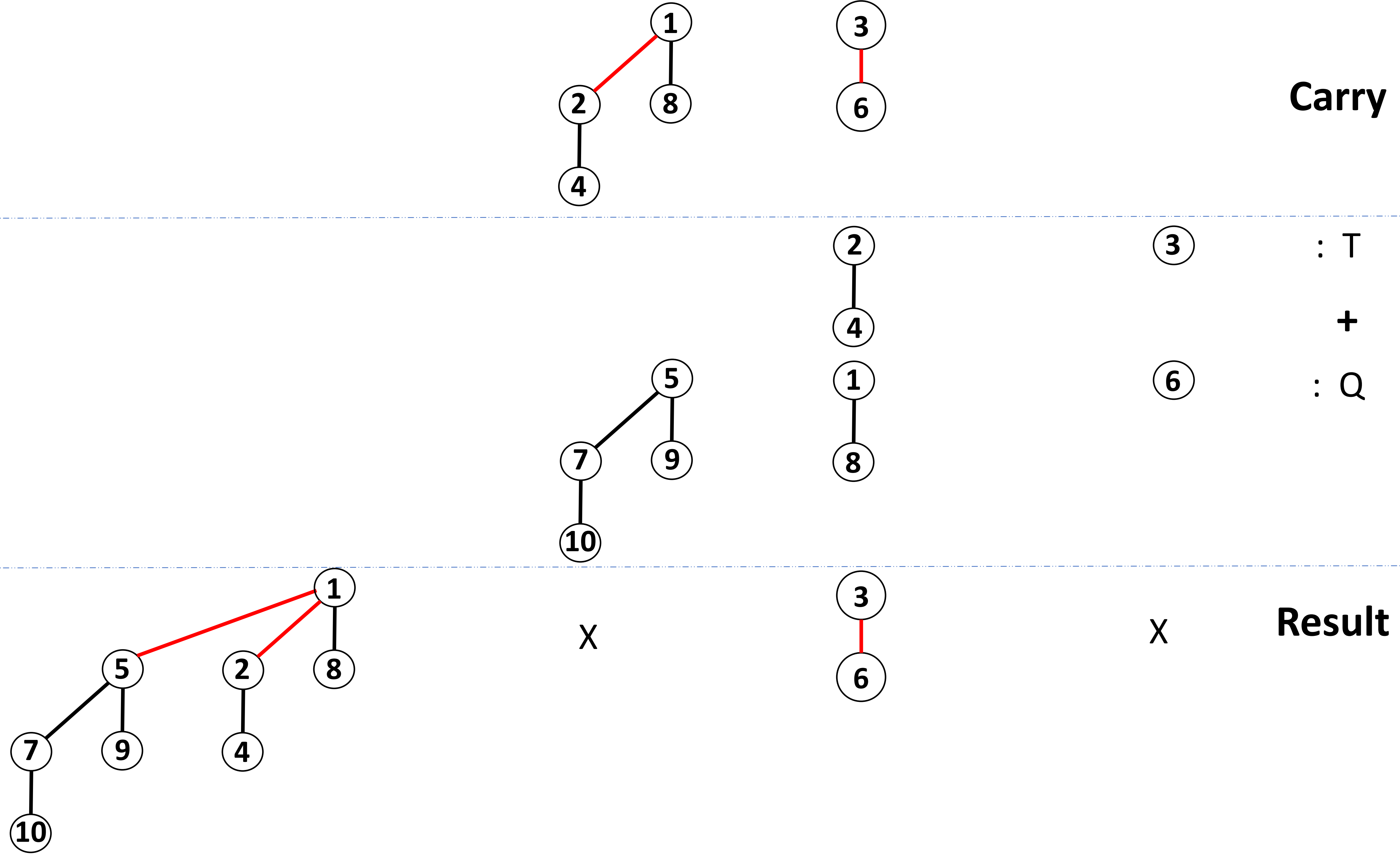
X **Result**



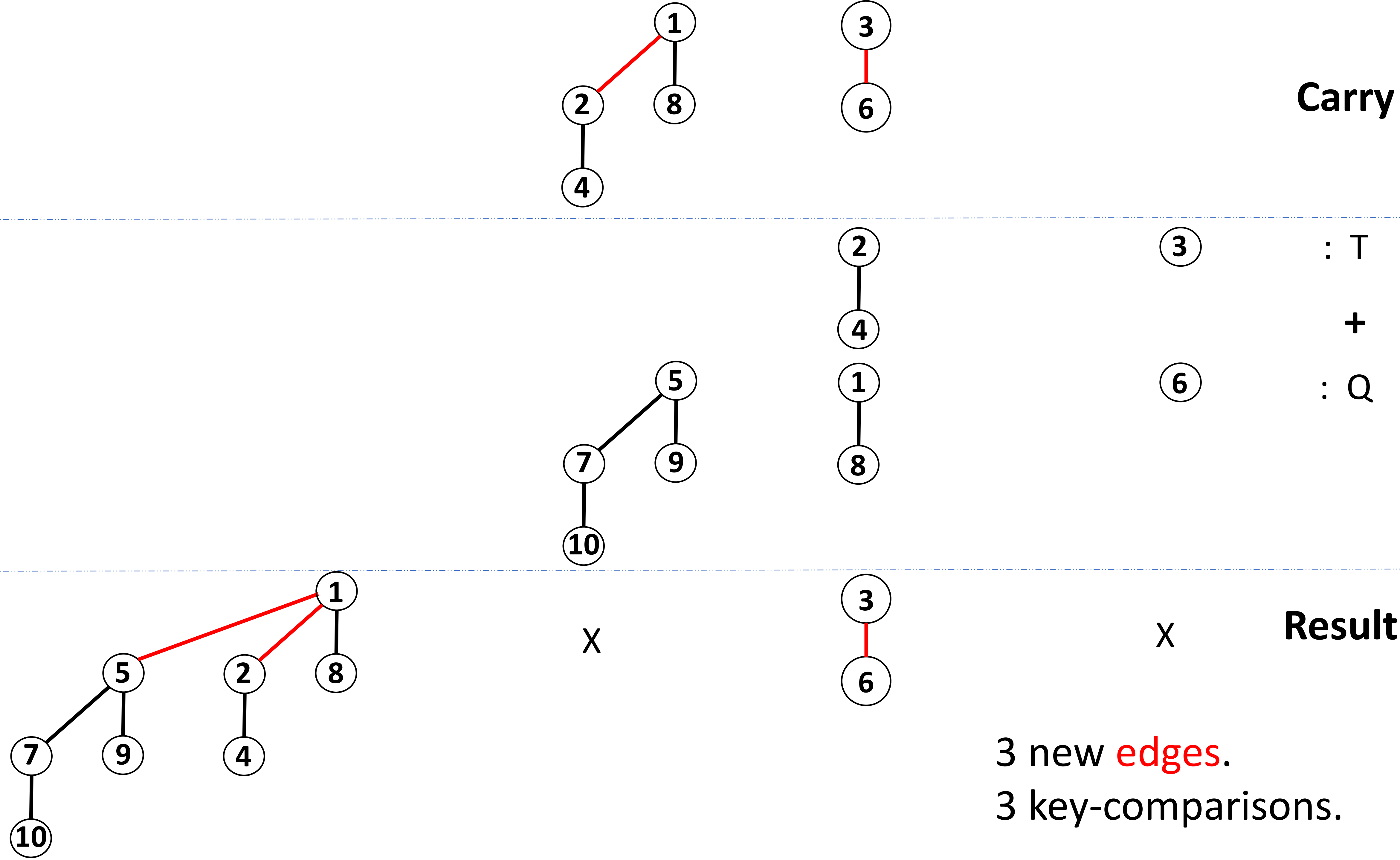
Carry
: T
+
: Q
X
Result



MERGE
1
2
8
4
5
7
9
10
3
6
Carry
2
4
1
8
3
: T
+
6
: Q
3
6
X
Result

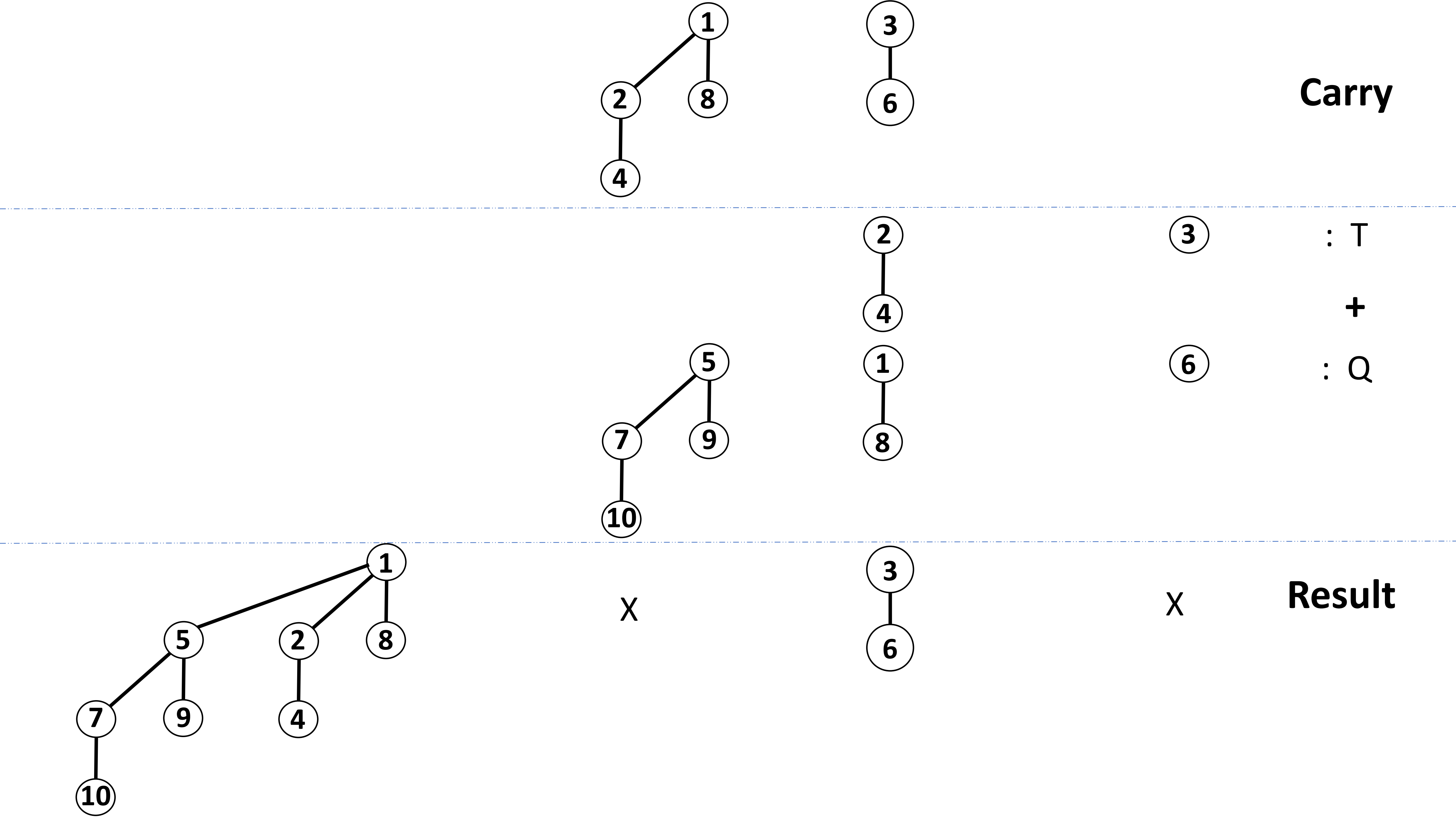


Carry
: T
+
: Q
X
Result



3 new edges.

3 key-comparisons.


Carry
: T
+
: Q
X
X
Result

Worst-Case Complexity of **Union(T, Q)**

Say |T| <= n and |Q| <= n (i.e. each contains at most n elements)

Worst-Case Complexity of **Union(T, Q)**

Say |T| <= n and |Q| <= n (i.e. each contains at most n elements)

$\Rightarrow$ Each of T, Q have O(log n) B_k trees.

Worst-Case Complexity of **Union(T, Q)**

Say |T| <= n and |Q| <= n (i.e. each contains at most n elements)

$\Rightarrow$ Each of T, Q have O(log n) B_k trees.

$\Rightarrow$ **Union**(T, Q) takes at most O(log n) key-comparisons