

UNIT - I

PRIORITY QUEUES



Objectives

- Priority queue definition
- Priority queue ADT
- Implementation of priority queues
- Binary heap/heap
- Heap operations
- Binomial queues
- Binomial queues operations

PRIORITY QUEUE DEFINITION

- A Priority Queue is a Collection of Zero or more elements where each element is assigned a priority and the order in which elements are processed is determined from the following rules.

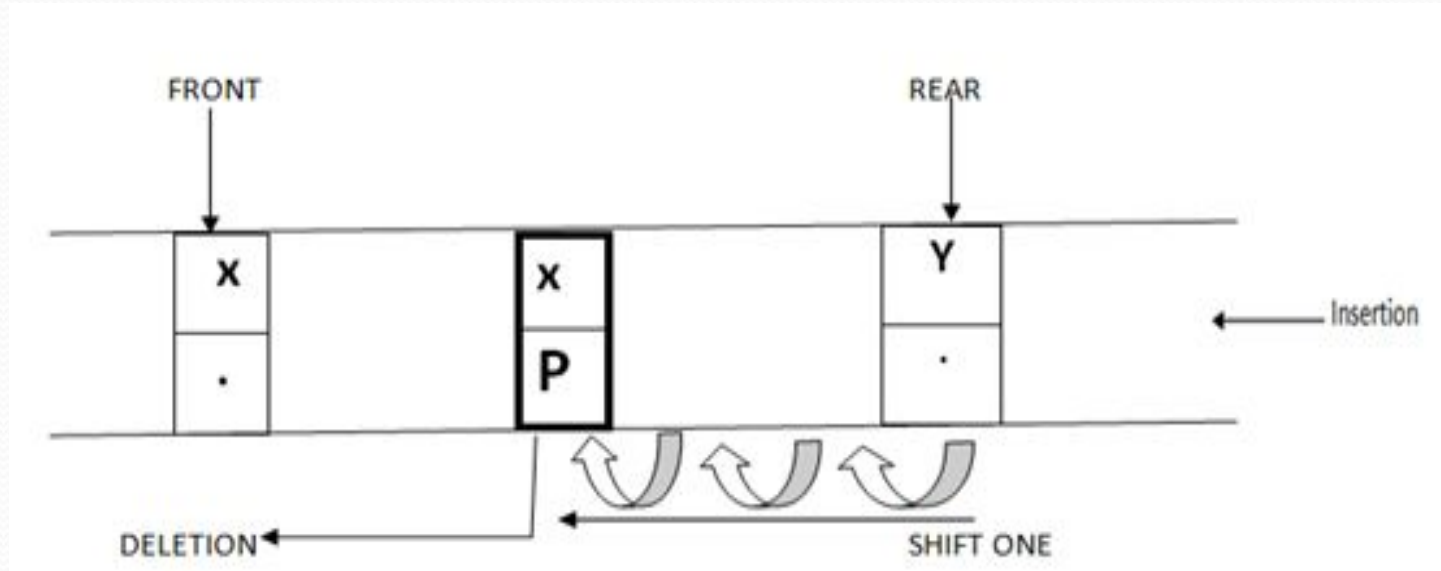
PRIORITY QUEUE ADT

- A priority queue stores a collection of entries
- Each **entry** is a pair (key, value)
- Main methods of the Priority Queue ADT
 - `insert(k, x)`
inserts an entry with key k and value x
 - `removeMin()`
removes and returns the entry with smallest key
- Additional methods
 - `min()`
returns, but does not remove, an entry with smallest key
 - `size()`, `isEmpty()`
- Applications:
 - Standby flyers
 - Auctions
 - Stock market

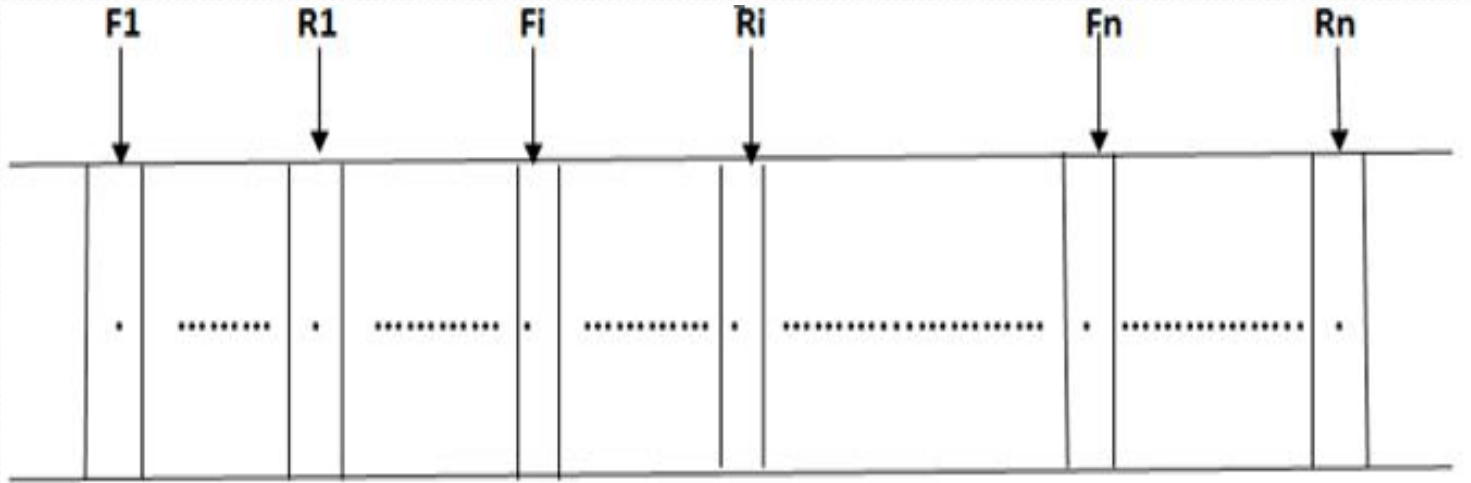
IMPLEMENTATION OF PRIORITY QUEUES

- Priority Queues can be implemented using
 - A simple /circular array
 - Multi-queue implementation
 - Linked list
 - Heap tree

Priority queue using an array



Multi-queue Implementation



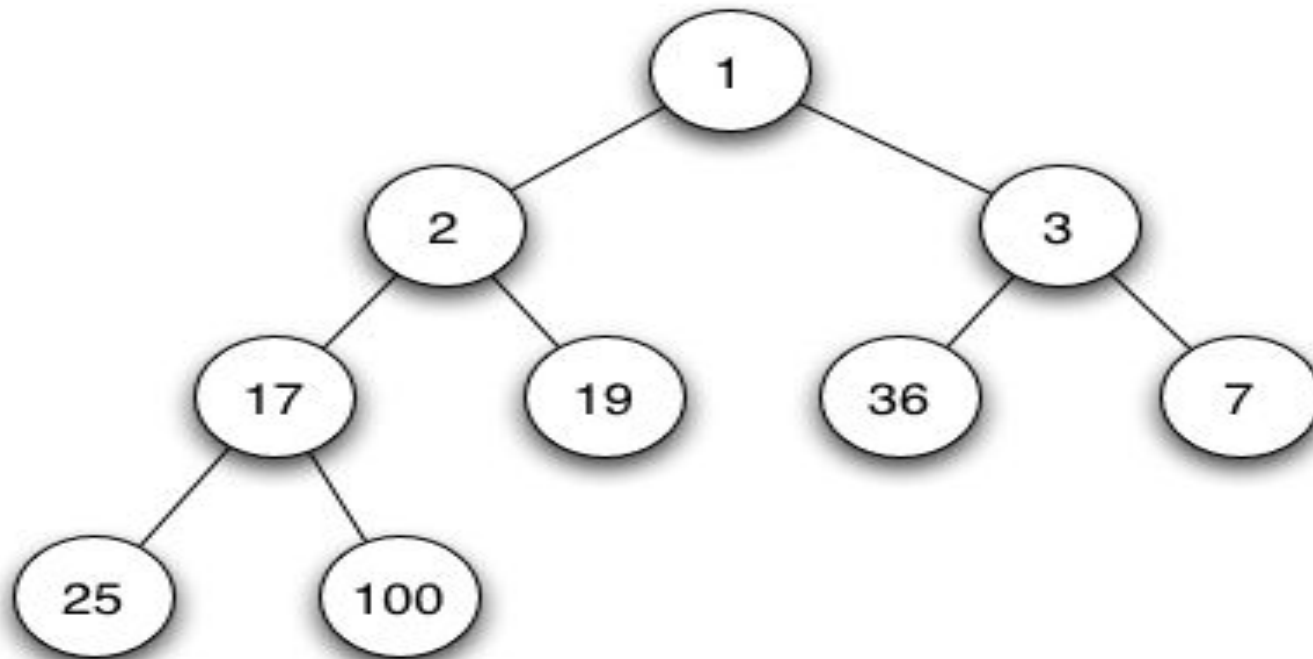
Priority queue using Linked list

- Performing insertions at front in $O(1)$ and traversing the list which requires $O(N)$ time
- To delete the minimum, we could insist that the list be kept always sorted: this makes insertions expensive $O(N)$ and delete-mins cheap $O(1)$

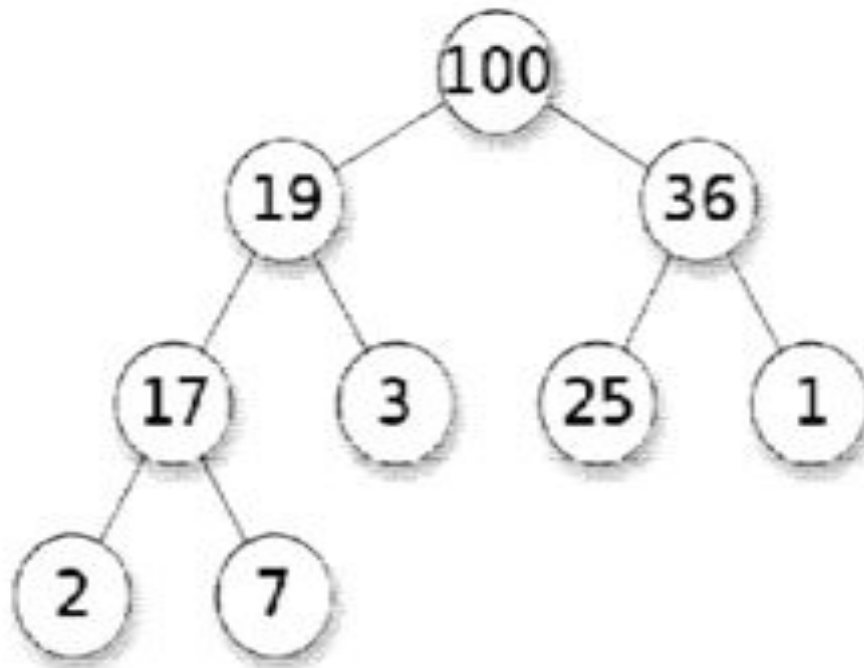
Binary Heap/Heap Tree

- ***Binary Heap/Heap Tree:*** A Heap is a Complete Binary tree with elements from partially ordered set which satisfies Heap Ordering property. The ordering can be of 2 types, They are
 - ***Min Heap***
 - ***Max Heap***

Min Heap: For each node N in a complete binary tree, the value of N is less than or equal to the value of its children's, such a heap tree is called a Min Heap.



Max Heap: For each node N in a complete binary tree, the value of N is greater than or equal to the value of its children's, such a heap tree is called a Max Heap.

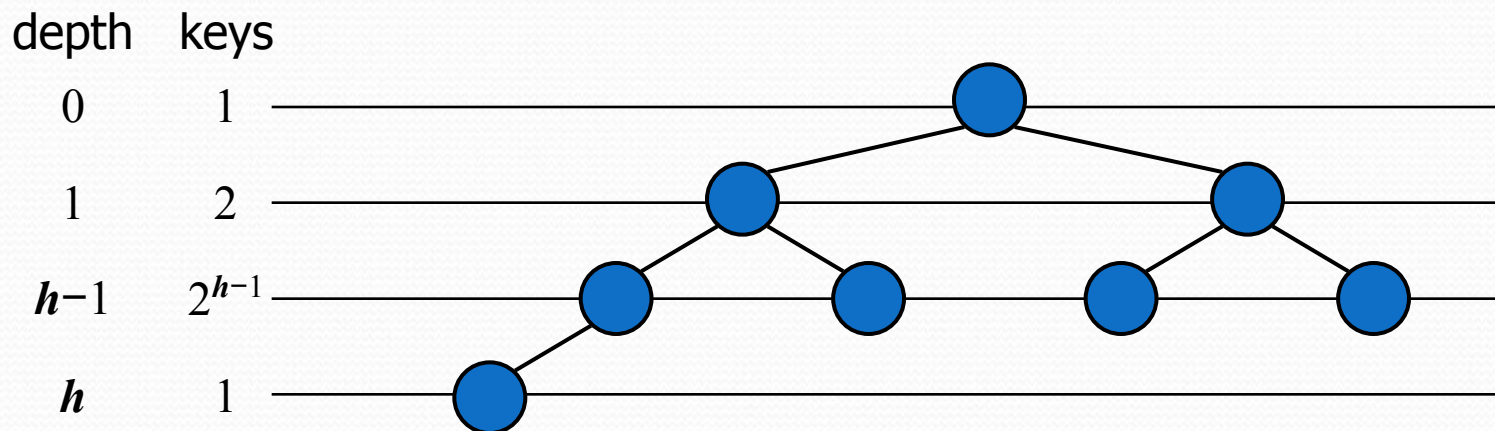


Height of a Heap

- **Theorem:** A heap storing n keys has height $O(\log n)$

Proof: (we apply the complete binary tree property)

- Let h be the height of a heap storing n keys
- Since there are 2^i keys at depth $i = 0, \dots, h-1$ and at least one key at depth h , we have $n \geq 1 + 2 + 4 + \dots + 2^{h-1} + 1$
- Thus, $n \geq 2^h$, i.e., $h \leq \log n$



Heap Operations

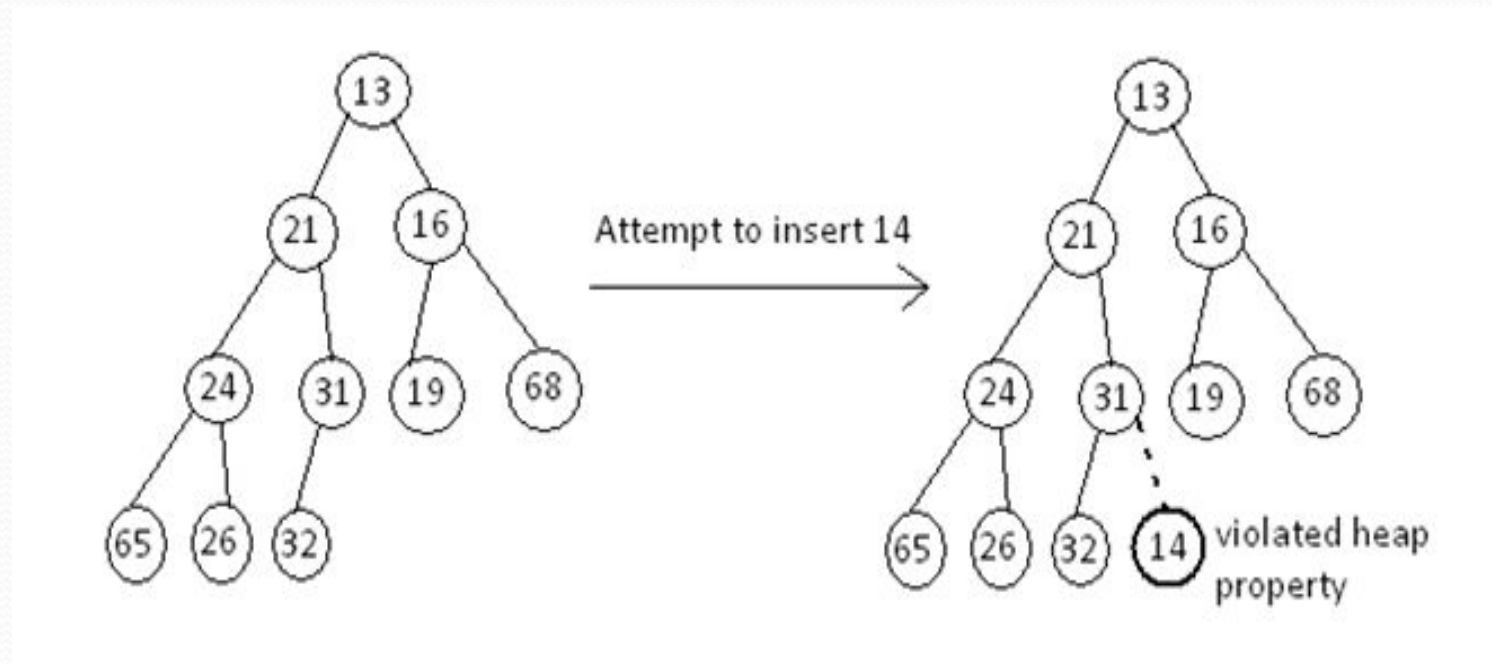
It is easy to perform the two required operations. All the work involves ensuring that the heap order property is maintained.

- Finding/searching an element
- Inserting a new element
- Deleting an element
- Merging of two heap trees

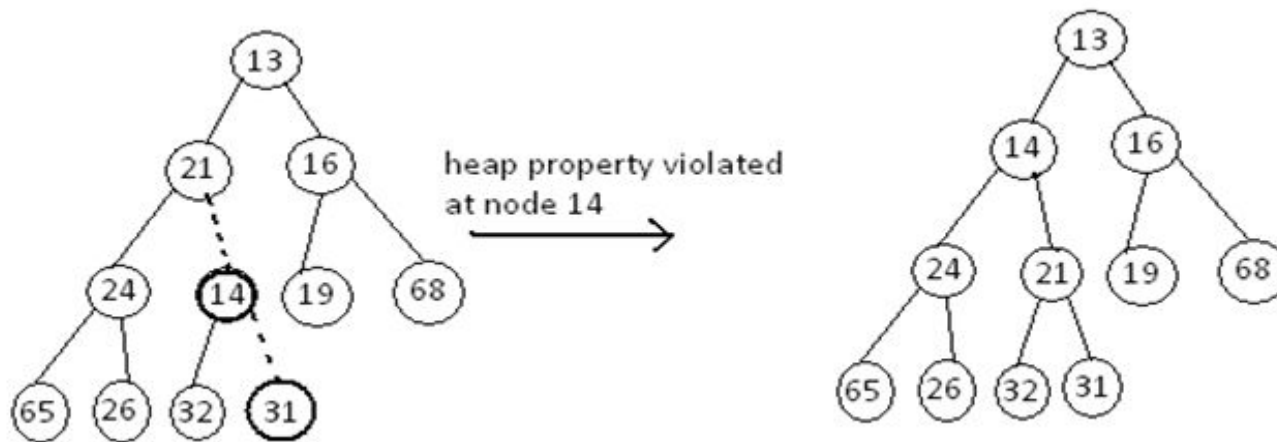
Insertion Operation On Binary Heap

- To insert an element say x , into the heap with n elements, we first create a hole in position $(n+1)$ and see if the heap property is violated by putting x into the hole.
- If the heap property is not violated, then we have found the correct position for x . Otherwise, we ``Re heap -up" or ``percolate-up" x until the heap property is restored.

Consider the heap



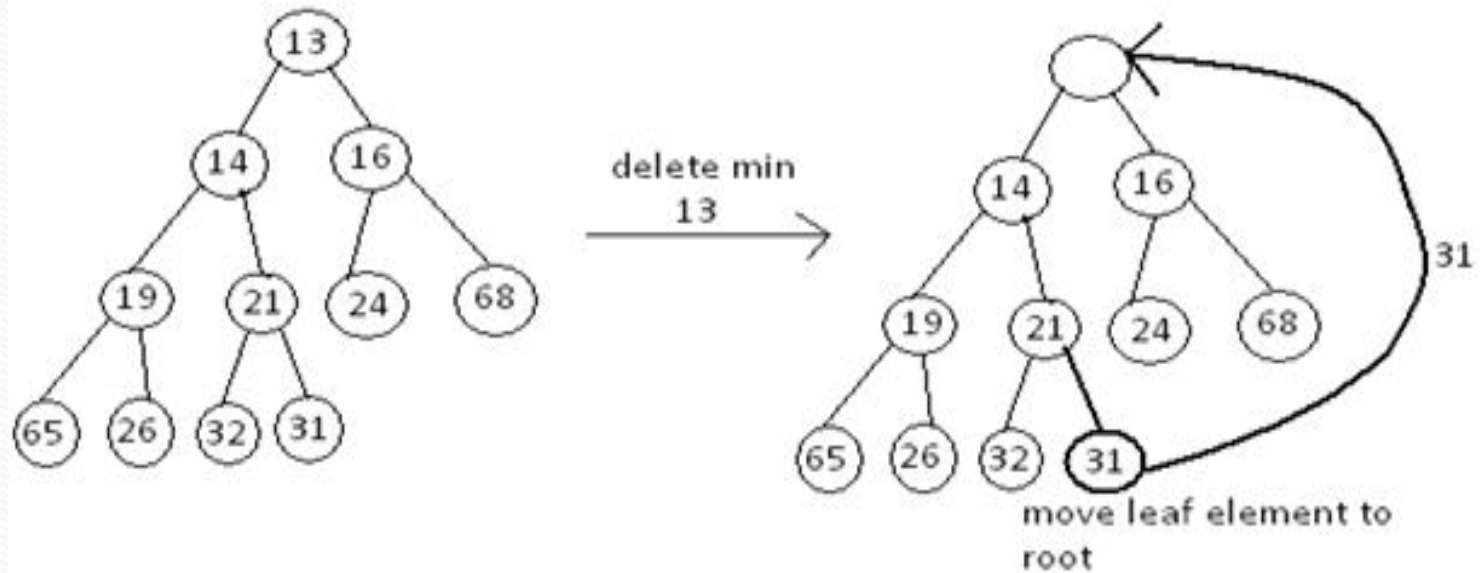
We create a hole in the next available heap location. Inserting 14 in the hole would violate the heap order property so 31 is slid down into the hole. This strategy is continued until the correct location for 14 is found

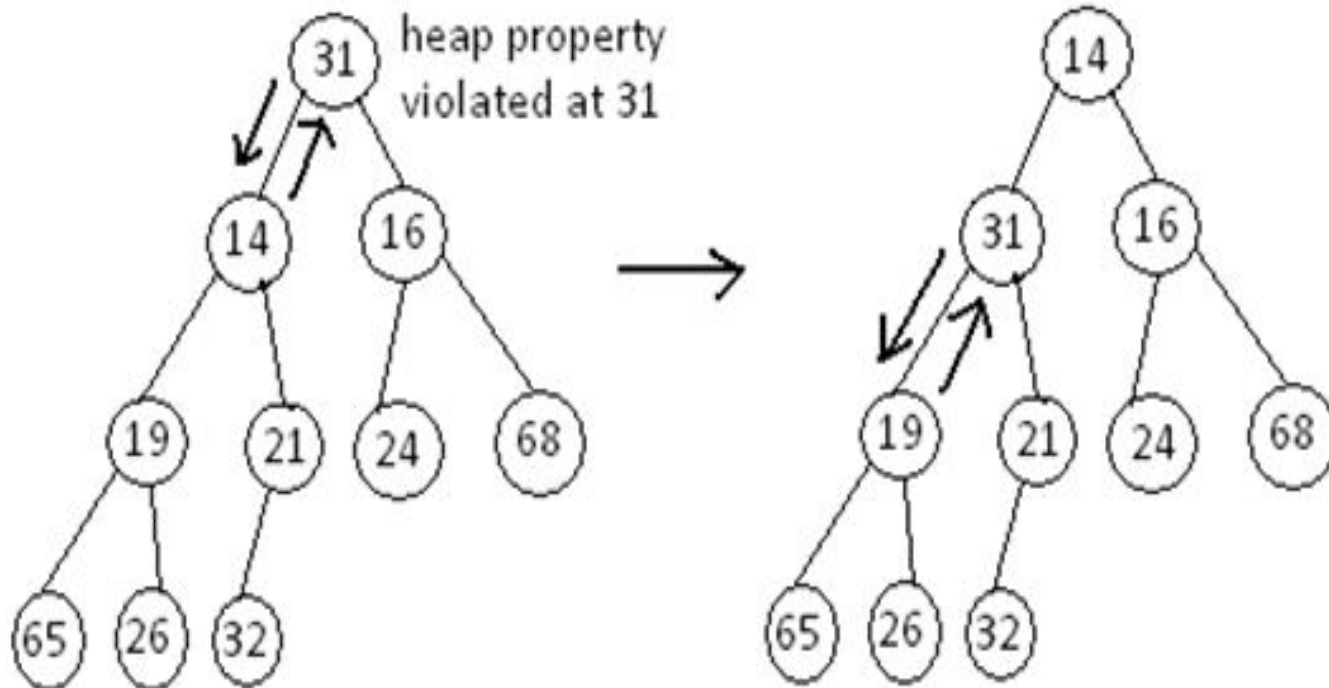


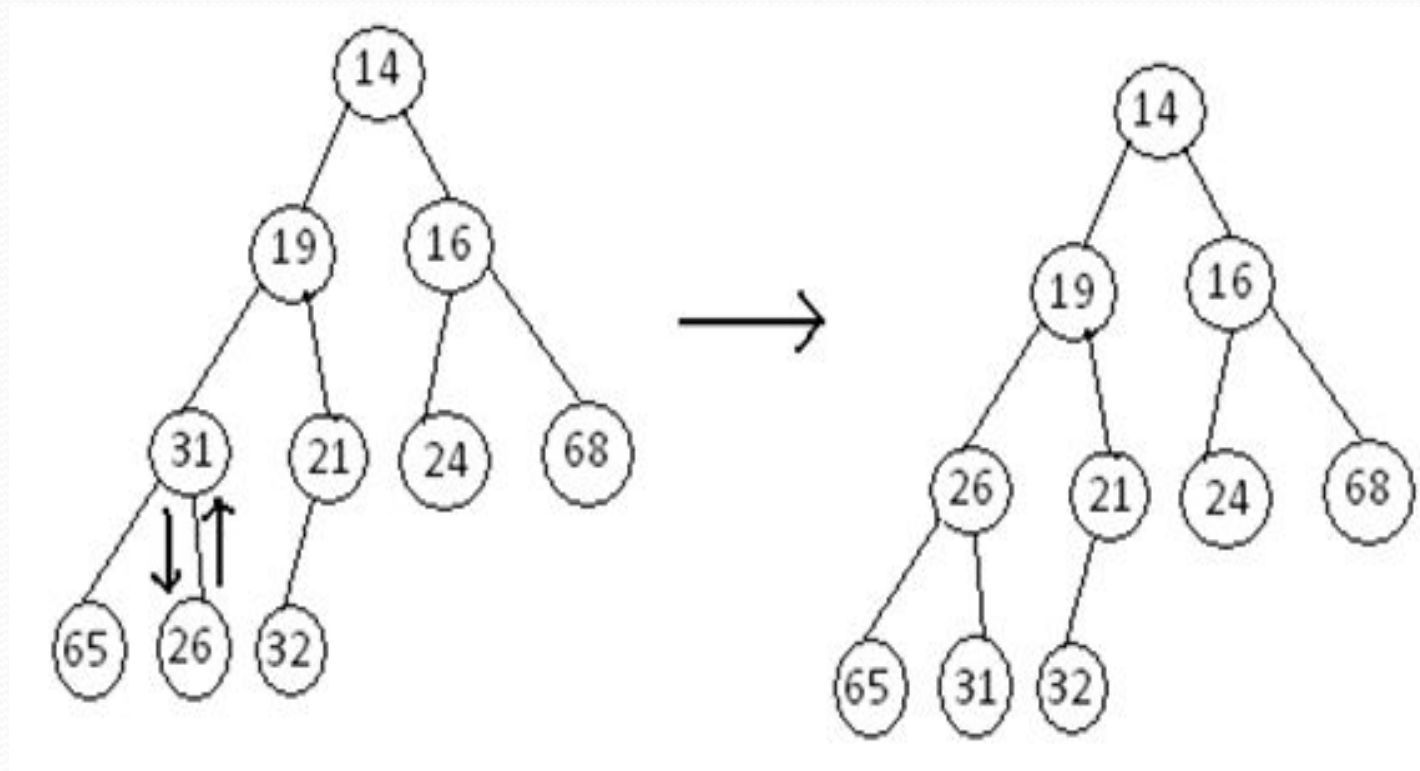
Delete min

- Where the minimum is deleted a hole is created at the root level. Since the heap now has one less element and the heap is a complete binary tree, the element in the least position is to be relocated

First remove or delete min is 13







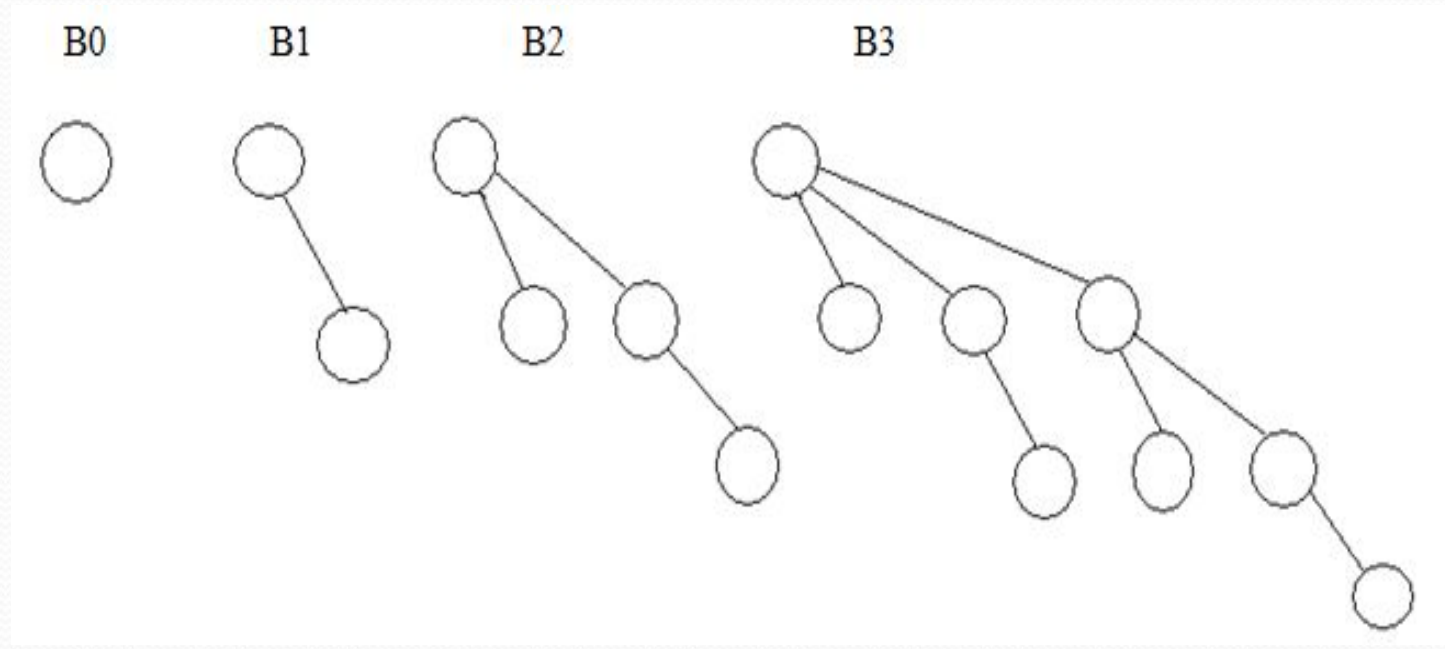
Binomial Queues

Binomial Queue Structure

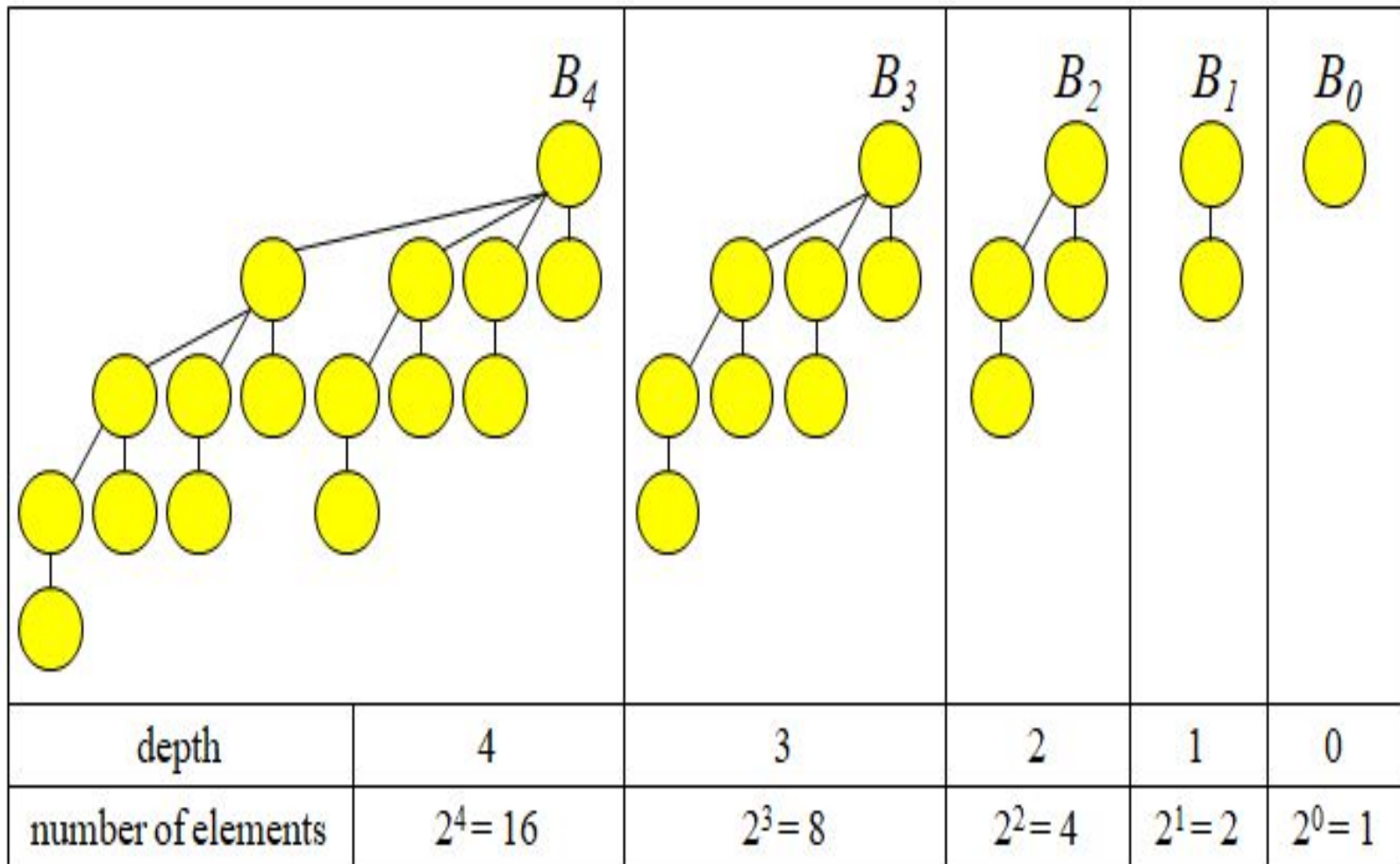
- It differs from all the priority queue implementations that a binomial queue is not a heap – ordered tree but rather a collection of heap – ordered trees known as a forest.

A binomial tree of height 0 is a one – node tree

A binomial tree B_k of height k is formed by attaching a binomial tree B_{k-1} to the root of another binomial tree B_{k-1}

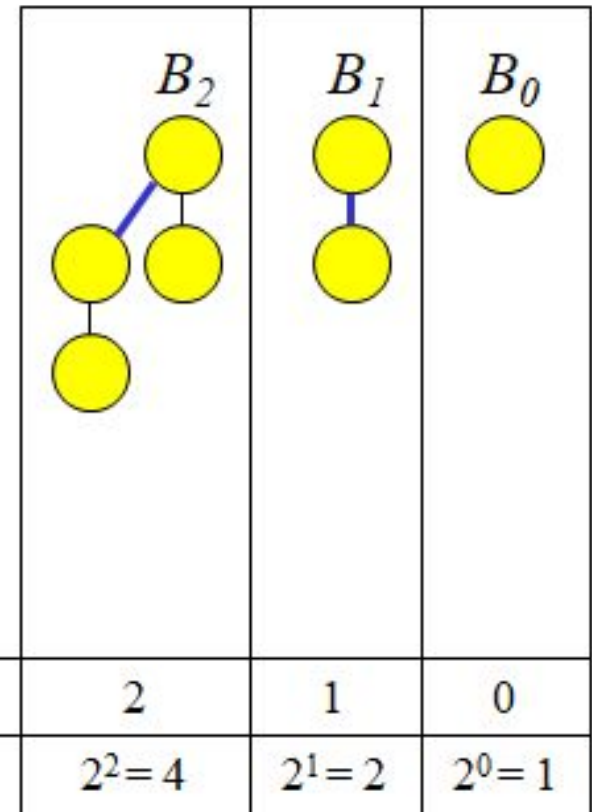


Binomial Queue with 5 Trees



Structure Property

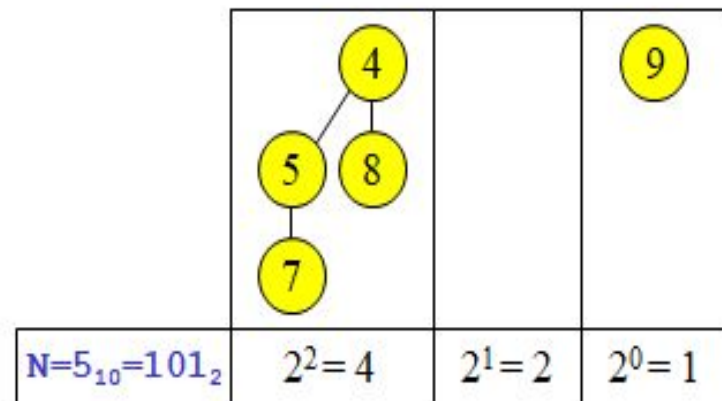
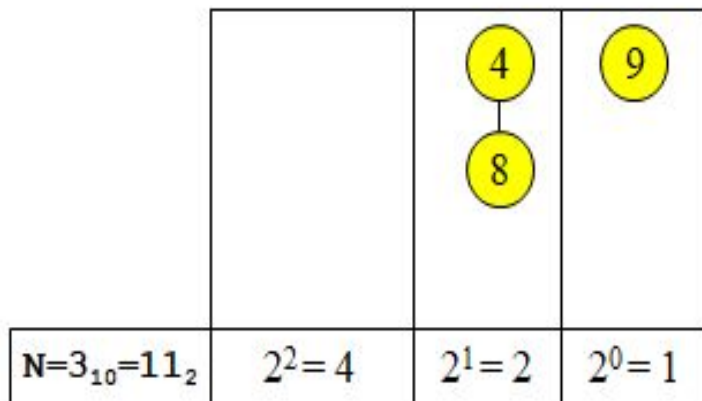
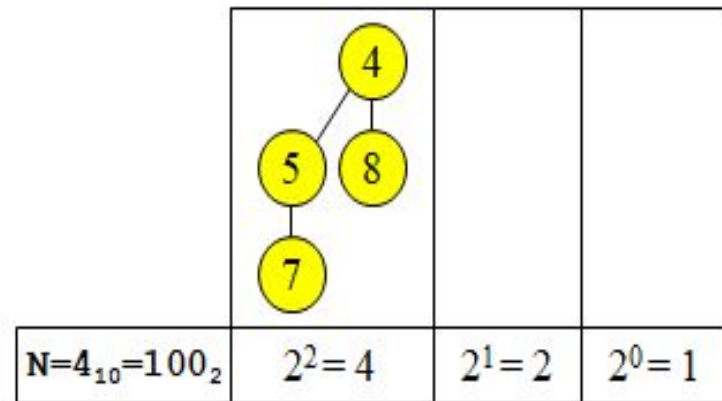
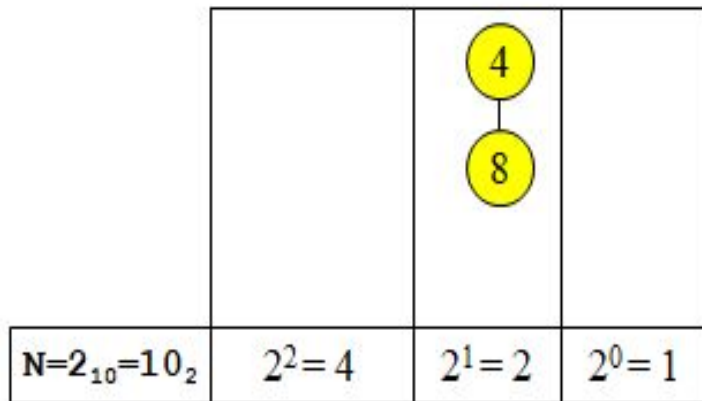
- Each tree contains two copies of the previous tree
 - › the second copy is **attached** at the root of the first copy
- The number of nodes in a tree of depth d is exactly 2^d



Numbers of nodes

- Any number of entries in the binomial queue can be stored in a forest of binomial trees
- Each tree holds the number of nodes appropriate to its depth, i.e., 2^d nodes
- So the **structure** of a forest of binomial trees can be characterized with a single binary number
 - $101_2 \rightarrow 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5$ nodes

Structure Examples



Binomial Queue operations

Find – min:

- This is implemented by scanning the roots of the entire tree. Since there are at most $\log n$ different trees, the minimum can be found in $O(\log n)$ time.
- Alternatively, one can keep track of the current minimum and perform find – min in $O(1)$ time. If we remember to update the minimum if it changes during other operations

Merge

- Merging two binomial queues is a conceptually easy operation, which we will describe by example.
- Consider the two binomial queues H1 and H2 with six and seven elements respectively as shown below.

What is a merge?

- There is a direct correlation between
 - the number of nodes in the tree
 - the representation of that number in base 2
 - and the actual structure of the tree
- When we merge two queues of sizes N_1 and N_2 , the number of nodes in the new queue is the *sum of* $N_1 + N_2$
- We can use that fact to help see how fast merges can be accomplished

Example 1.

Merge BQ.1 and
BQ.2

Easy Case.

There are no
comparisons and
there is no
restructuring.

BQ.1			9
$N=1_{10}=1_2$	$2^2=4$	$2^1=2$	$2^0=1$
+ BQ.2		4 8	
$N=2_{10}=10_2$	$2^2=4$	$2^1=2$	$2^0=1$
= BQ.3		4 8	9
$N=3_{10}=11_2$	$2^2=4$	$2^1=2$	$2^0=1$

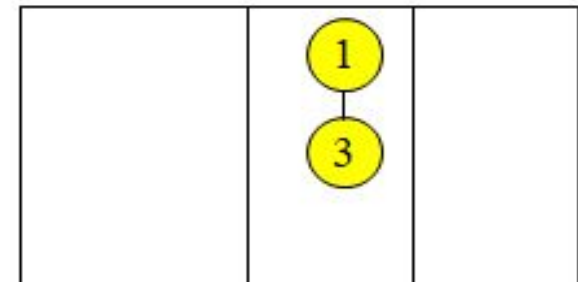
Example 2.

Merge BQ.1 and BQ.2

This is an add with a carry out.

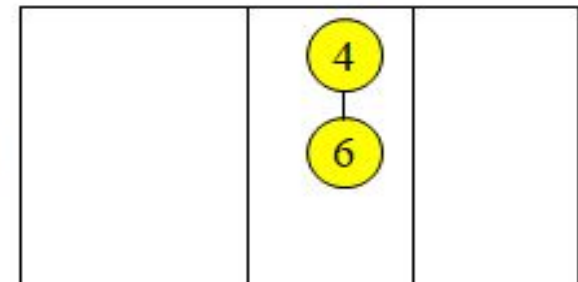
It is accomplished with one comparison and one pointer change:
 $O(1)$

BQ.1



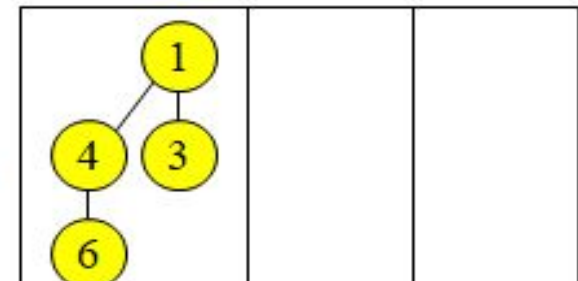
$N=2_{10}=10_2$	$2^2=4$	$2^1=2$	$2^0=1$
-----------------	---------	---------	---------

+ BQ.2



$N=2_{10}=10_2$	$2^2=4$	$2^1=2$	$2^0=1$
-----------------	---------	---------	---------

= BQ.3



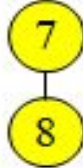
$N=4_{10}=100_2$	$2^2=4$	$2^1=2$	$2^0=1$
------------------	---------	---------	---------

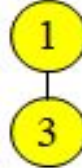

Example 3.

Merge BQ.1 and BQ.2

Part 1 - Form the carry.



BQ.1		<div>1</div> <div>3</div>	7
$N=3_{10}=11_2$	$2^2=4$	$2^1=2$	$2^0=1$
+ BQ.2		<div>4</div> <div>6</div>	8
	$N=3_{10}=11_2$	$2^2=4$	$2^1=2$
			$2^0=1$
= carry		<div>7</div> <div>8</div>	
	$N=2_{10}=10_2$	$2^2=4$	$2^1=2$
			$2^0=1$

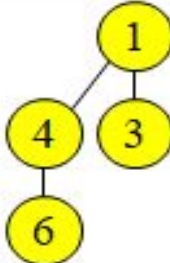

carry			
$N=2_{10}=10_2$	$2^2=4$	$2^1=2$	$2^0=1$

+ BQ.1			
$N=3_{10}=11_2$	$2^2=4$	$2^1=2$	$2^0=1$

Example 3.

Part 2 - Add the existing values and the carry.

+ BQ.2			
$N=3_{10}=11_2$	$2^2=4$	$2^1=2$	$2^0=1$

= BQ.3			
$N=6_{10}=110_2$	$2^2=4$	$2^1=2$	$2^0=1$

Text Books:

1. Data structures, Algorithms and Applications in C++, S.Sahni, University Press (India) Pvt Ltd, 2nd edition, Universities Press Orient Longman Pvt. Ltd.
2. Data structures and Algorithms in C++, Michael T.Goodrich, R.Tamassia and .Mount, Wiley student edition, John Wiley and Sons.
3. Data structures and Algorithm Analysis in C++, Mark Allen Weiss, Pearson Education. Ltd., Second edition.

References

1. Data structures and algorithms in C++, 3rd Edition, Adam Drozdek, Thomson
2. Data structures using C and C++, Langsam, Augenstein and Tanenbaum, PHI.
3. File Structures :An Object oriented approach with C++, 3rd ed, Michel J Folk, Greg Riccardi, Bill Zoellick

Links

- <https://www.slideshare.net/PriyankaRana17/priority-queues-75351526>
- https://www.powershow.com/view/254292-YmY3N/Priority_Queues_powerpoint_ppt_presentation

Thank You

