Поразрядная *LSD* - сортировка

LSD – least significant digit radix sort – поразрядная сортировка сначала по **младшей** цифре.

- Анализ цифр ключа справа налево;
- сортировка работает только в случае устойчивого способа перестановки элементов;
- устойчив, например, способ подсчета.

Поразрядная *LSD* - сортировка

01001	10010	10010	10100
10010	10110	10110	00000
11011	10100	10100	10000
10110	00000	00000	01001
00011	10000	10000	10010
10100	01001	01001	10110
00000	11011	11011	11011
10000	00011	00011	00011

Поразрядная *LSD* - сортировка

10100	00000	00000	00000
00000	10000	10000	10000
10000	01001	01001	10010
01001	10010	10010	00011
10010	11011	11011	10100
10110	00011	00011	10110
11011	10100	10100	01001
00011	10110	10110	11011

Поразрядная *LSD* - сортировка

00000	00000
10000	00011
10010	01001
00011	10000
10100	10010
10110	10100
01001	10110
11011	11011

LSD-сортировка

цикл для *i* от 0 до *D* нц

|| D – количество разрядов ключа

Сортировка разряда *i*

кц

конец

Сортировка не рекурсивная

Очереди по приоритетам

Во многих приложениях требуется обработка записей с упорядоченными определенным образом ключами:

- **не обязательно в строгом порядке;**
- **не обязательно всех сразу.**

накапливается набор записей

обрабатывается запись с максимальным ключом

накапливается набор записей

обрабатывается запись с максимальным ключом

Очереди по приоритетам

Структура данных, поддерживающая операции

- **вставки нового элемента;**
- **удаления элемента с максимальным значением ключа**

называется

очередью по приоритетам.

Очереди по приоритетам

- Системы моделирования – каждое событие системы характеризуется моментом возникновения, это помогает обслуживать события в хронологическом порядке.
- Системы планирования заданий в вычислительных системах – приоритет указывает, какая задача должна выполнятся первой
- Численные расчеты – приоритет (ошибка), наиболее грубые ошибки должны исправляться первыми.

Очереди по приоритетам

На практике очереди по приоритетам более сложны:

- создать очередь по приоритетам из *N* заданных элементов;
- изменить приоритет произвольного элемента;
- удалить произвольный элемент;
- объединить две очереди по приоритетам в одну.

Очереди по приоритетам

Базовые структуры для очереди:

- односвязный список,
- двусвязный список
- массив.

Каждая из описанных процедур может быть реализована различными способами, в зависимости от конкретной решаемой задачи.

Очереди по приоритетам

Процедура вставки

- Вставлять элемент в начало очереди.
- Вставлять элемент в конец очереди.
- Вставлять элемент в заданное место.

Процедура удаления элемента с наибольшим ключом

- Найти элемент с максимальным ключом и переставить его в конец. Удалить элемент.
- Найти элемент с максимальным ключом и сразу удалить его.

Очереди по приоритетам

Упорядоченные последовательности элементов

- Процедура вставки требует просмотра всей последовательности элементов - O(n).

- Процедура удаления и поиска максимального выполняется за постоянное время - O(1).

Очереди по приоритетам

Неупорядоченные последовательности

- Процедура вставки выполняется за постоянное время – O(1).
- Процедура удаления и поиска требует просмотра всей последовательности – O(n).

Подходы к организации работы

Ленивый – основная работа откладывается.

Энергичный – основная работа выполняется на этапе подготовки последовательности.

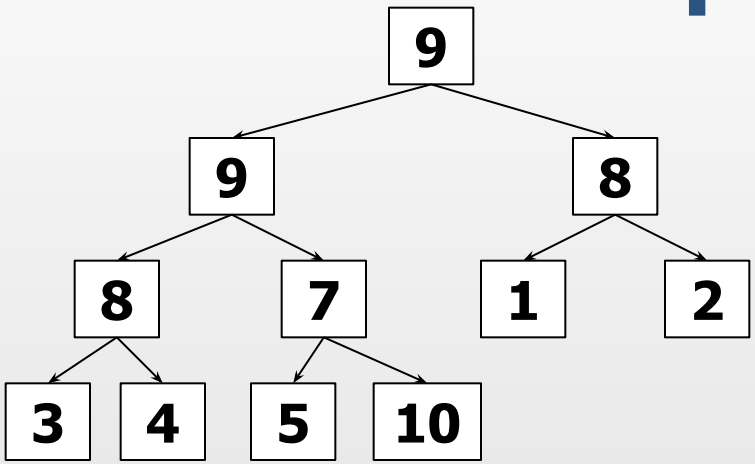
Очереди по приоритетам

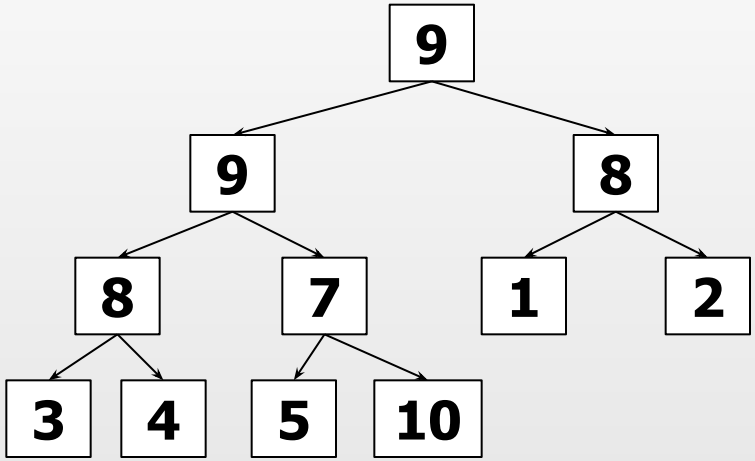
Представление очереди с помощью пирамиды

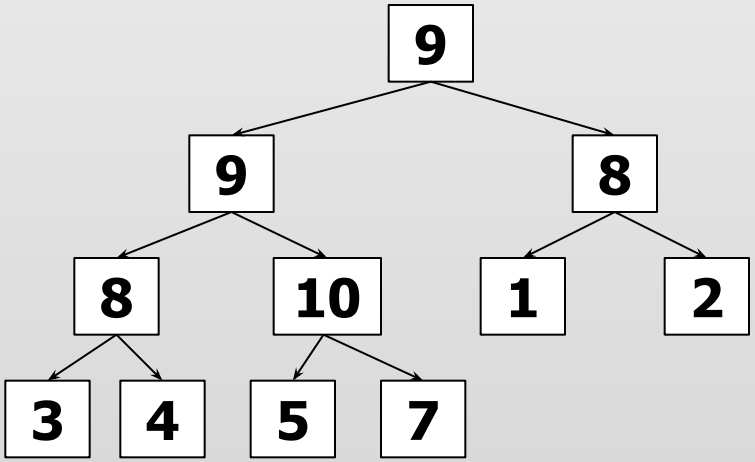
- Частично упорядоченный массив;
- в корне дерева (первый элемент массива) находится элемент с наибольшим ключом.

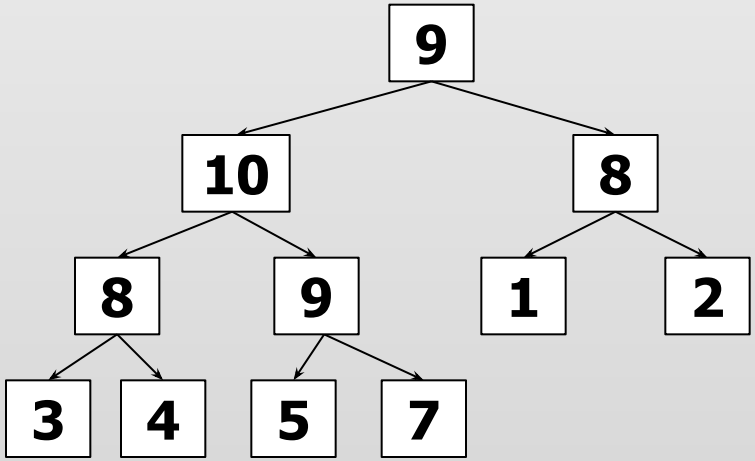
Процедура добавления нового элемента

Представление очереди пирамидой









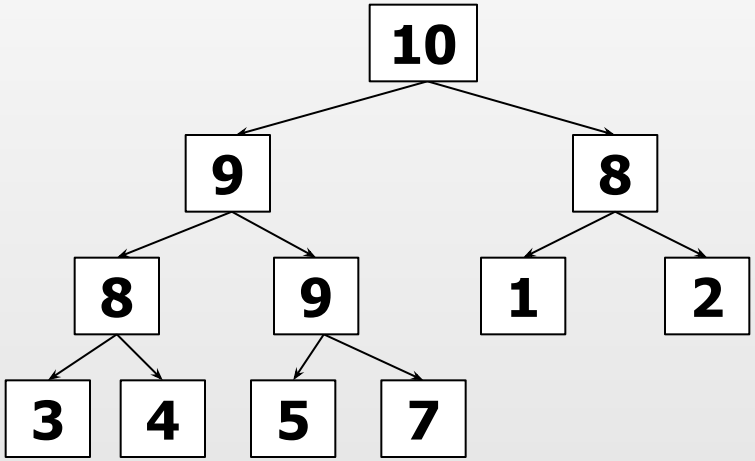
Добавление нового элемента

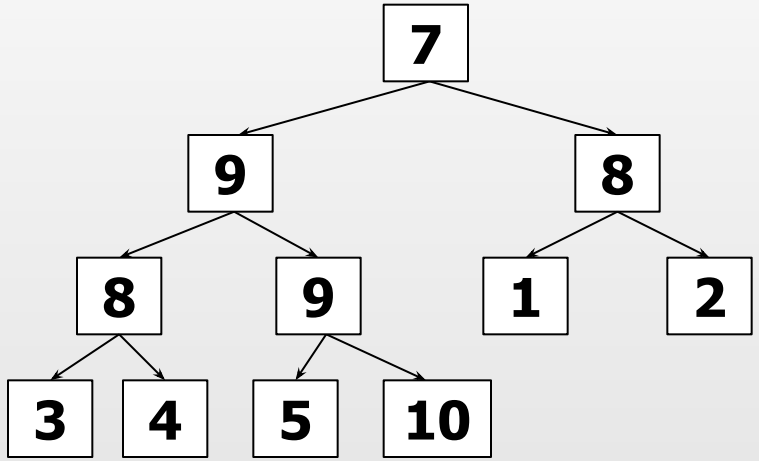
Представление очереди пирамидой

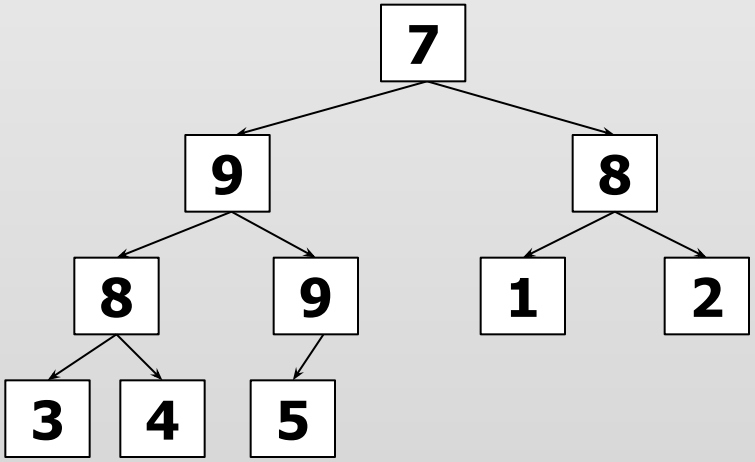
- **Добавление нового элемента в конец массива.**
- **Передвижение элемента к своему месту.**

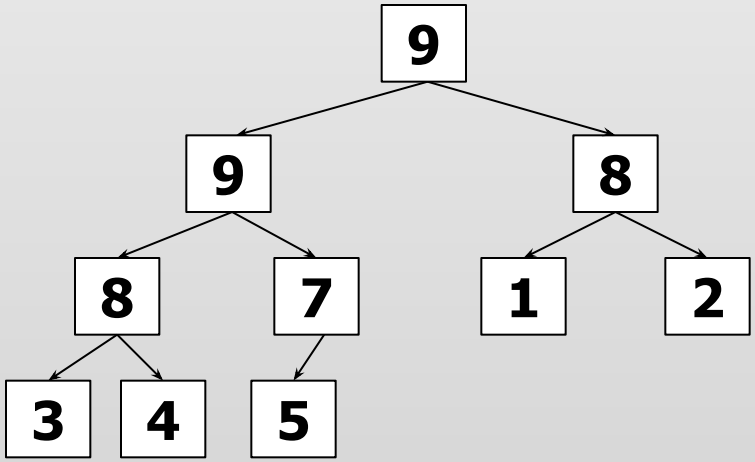
Добавление нового элемента

Представление очереди пирамидой









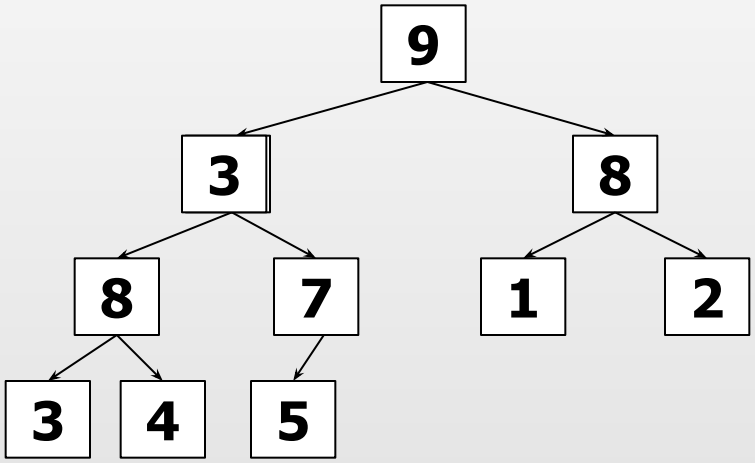
Удаление максимального элемента

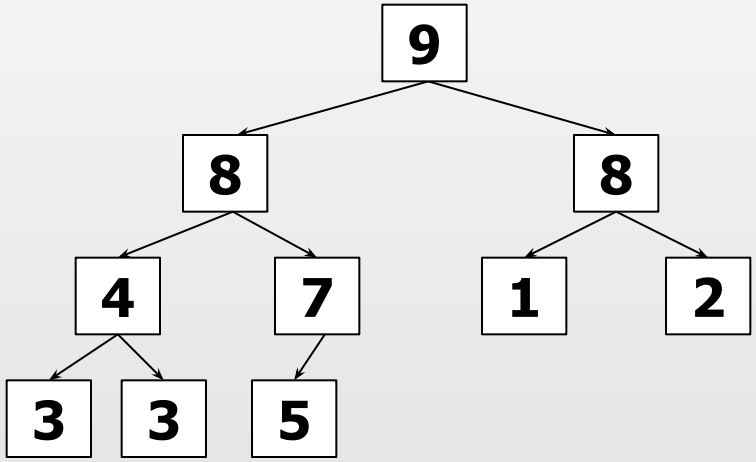
Представление очереди пирамидой

- **Обмен нулевого и последнего элемента**
- **Удаление последнего элемента массива.**
- **Перестройка пирамиды на нулевом элементе.**

Удаление максимального элемента

Представление очереди пирамидой





При уменьшении приоритета – «спуск элемента»

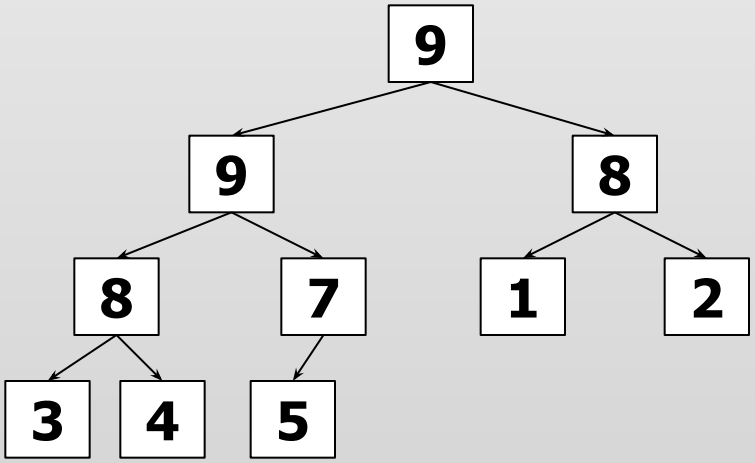
При увеличении приоритета – «подъем» элемента

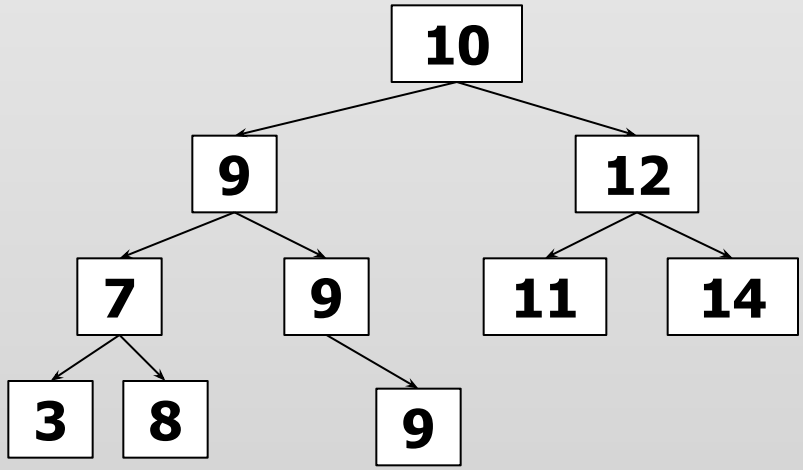
Изменение приоритета элемента

Биномиальная очередь

(1978 г. Вильемин)

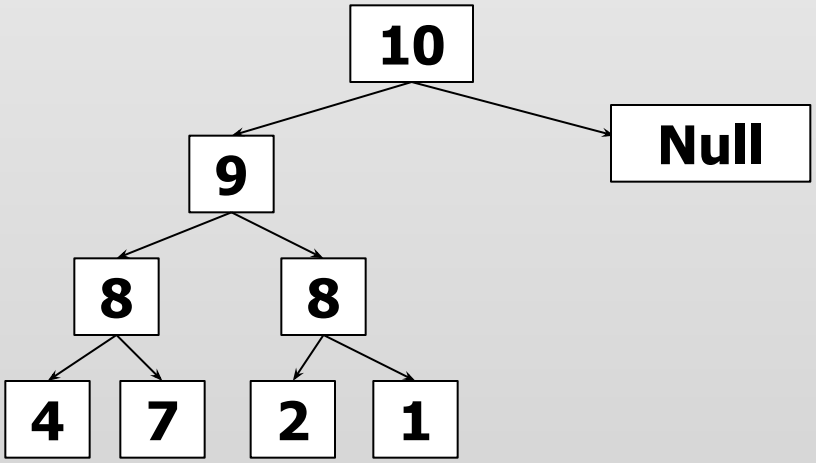
Бинарное дерево называется левосторонним пирамидально упорядоченным, если ключ каждого узла больше или равен всем ключам левого поддерева, если таковое имеется.





Биномиальная очередь

Сортирующее дерево степени 2 есть левостороннее пирамидально упорядоченное дерево, состоящее из корневого узла с пустым правым поддеревом и полным левым поддеревом.



Биномиальная очередь

Левый потомок сортирующего дерева степени 2 называется биномиальным деревом.

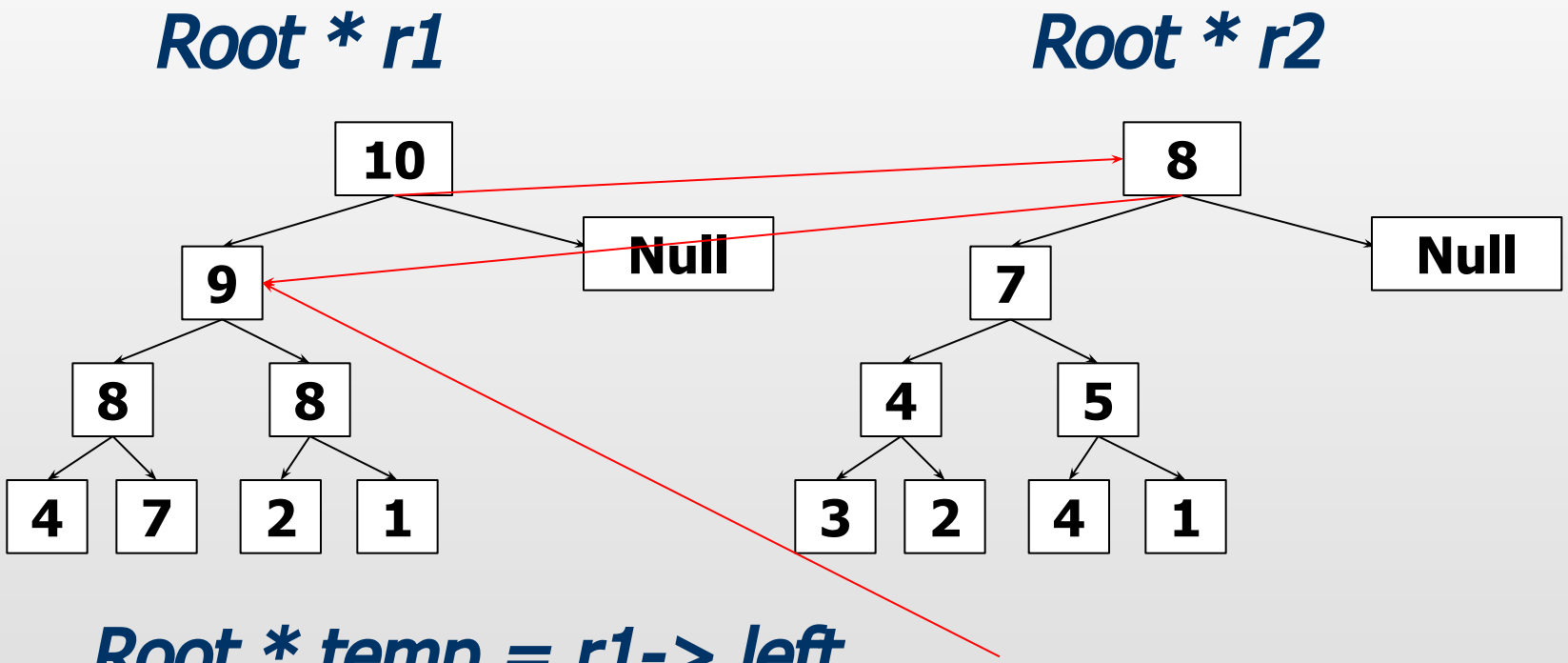
- **Кол-во узлов сортирующего дерева степени 2 есть число степени 2.**
- **Ни один из ключей не обладает значением, большим ключа корня дерева.**
- **Биномиальные деревья пирамидально упорядочены.**

Биномиальная очередь

Если реализация биномиального дерева выполнена на основе связных структур, то объединение двух деревьев одинакового размера в одно сводится к изменению двух связей.

```
struct  Root {
int d;
Root *left;
Root *rigth; }
```

Биномиальная очередь

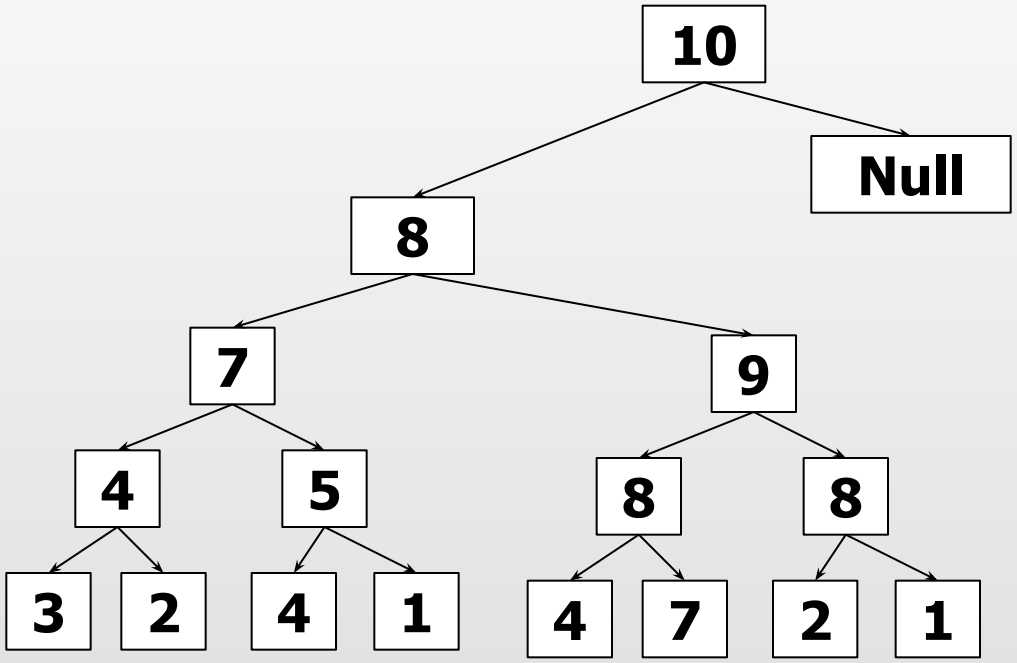


*Root * temp = r1-> left*

r1-> left = r2

r2-> right = temp

Биномиальная очередь

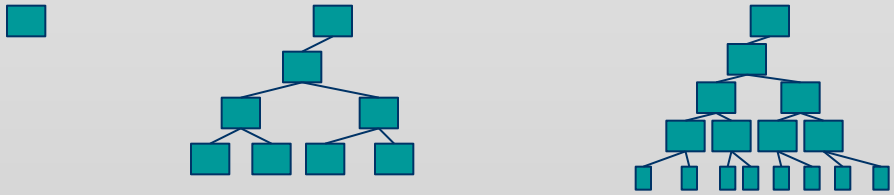


Биномиальная очередь

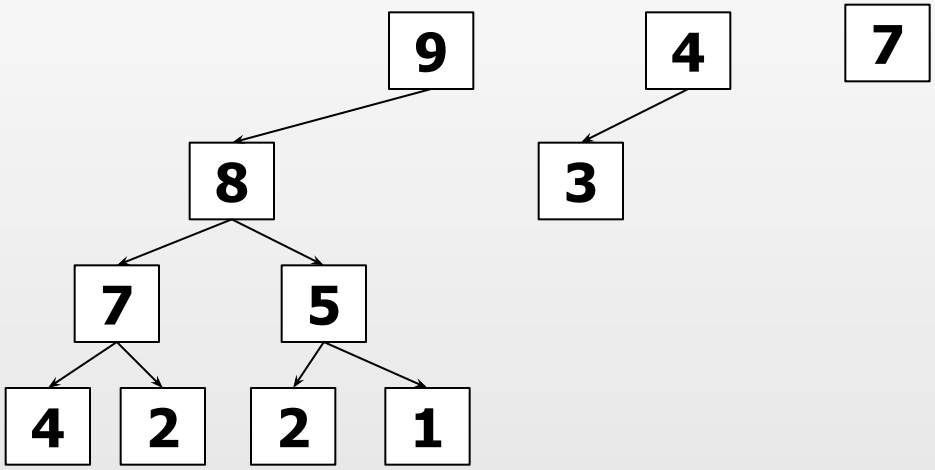
Биномиальная очередь представляет собой набор сортирующих деревьев степени 2 , ни одно из которых **не совпадает** с другими по размеру.

Структура биномиальной очереди совпадает с единичными разрядами двоичного представления числа узлов этой очереди.

$25_{10} = 11001_2$



Биномиальная очередь



аналог операции +1 $1010_2 + 1_2 = 1011_2$

Добавление нового элемента

Биномиальная очередь

Алгоритм добавления нового элемента

проинициализировать новый элемент как 1-дерево переноса

i=0

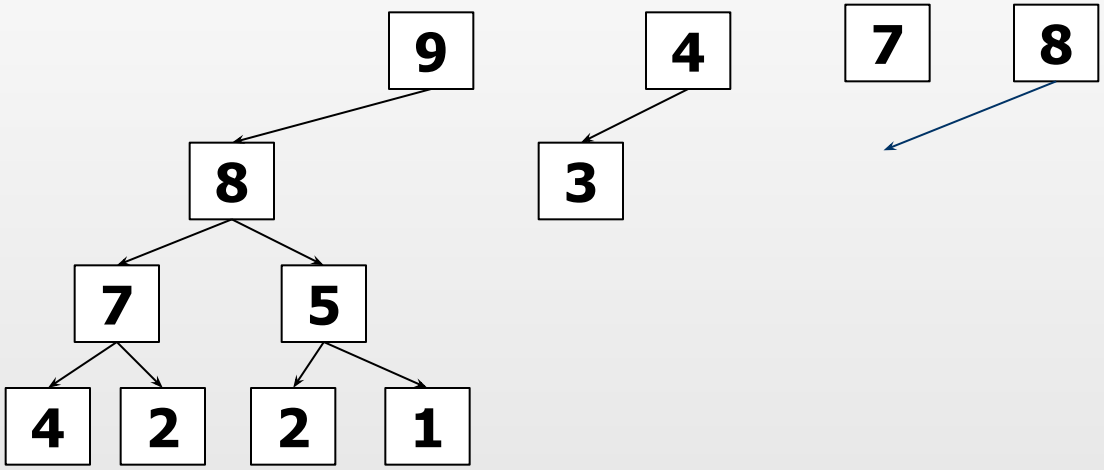
пока не обнаружится пустая позиция для сортирующего дерева нц

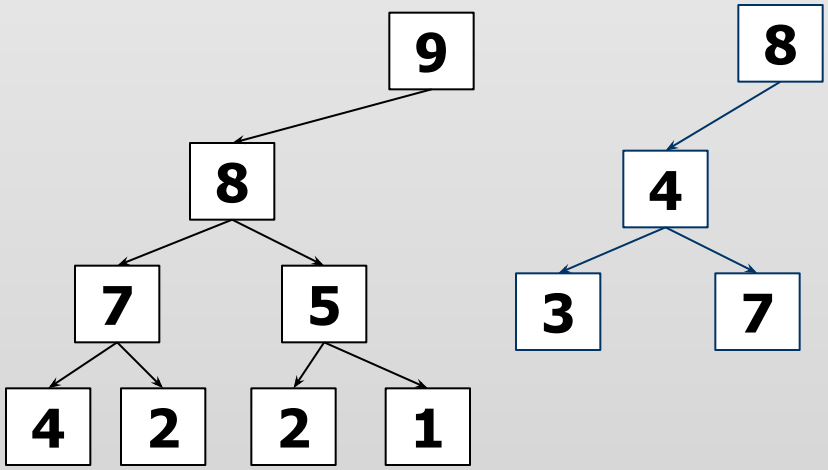
если в очереди есть 2^i дерево, то объединить его с i - деревом переноса

записать полученное дерево в дерево 2^{i+1} переноса.

i=i+1 кц

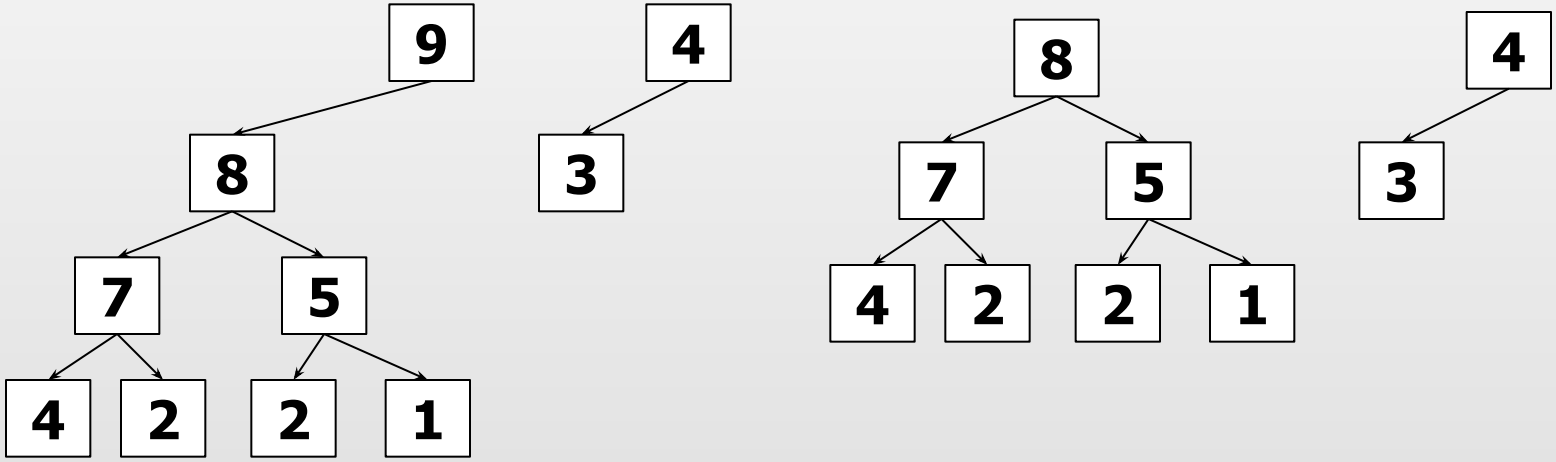
Биномиальная очередь





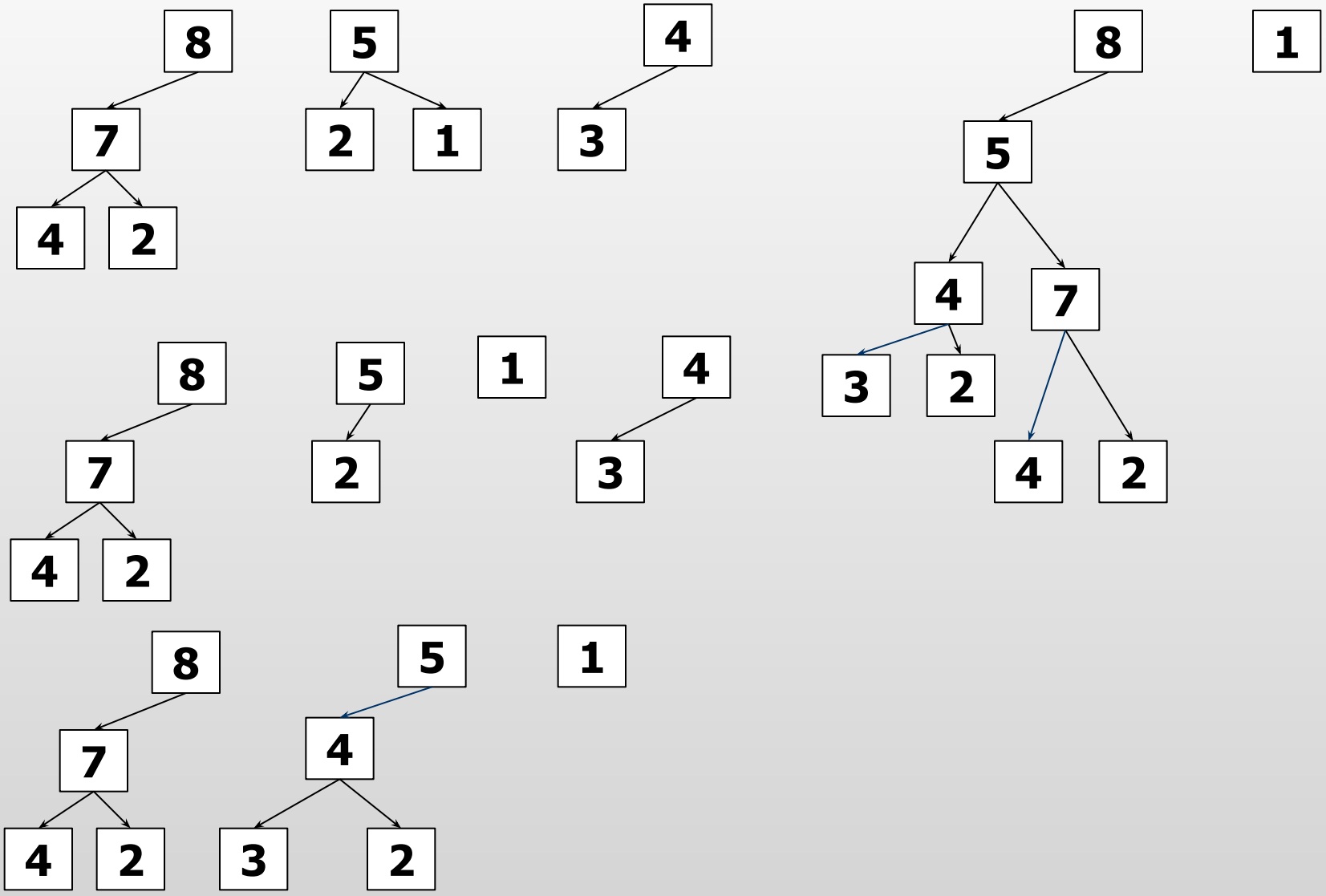
Биномиальная очередь

аналог операции -1 $1010_2 - 1_2 = 1001_2$



Удаление максимального элемента

Биномиальная очередь



Биномиальная очередь

Алгоритм удаления элемента

найти дерево 2^i в корне которого расположен максимальный элемент.

удалить максимальный элемент

2^i дерево разбить на i-1, i-2, … 0 деревья переноса.

i=0

пока не обнаружится пустая позиция для сортирующего дерева нц

если в очереди есть 2^i дерево, то объединить его с i - деревом переноса

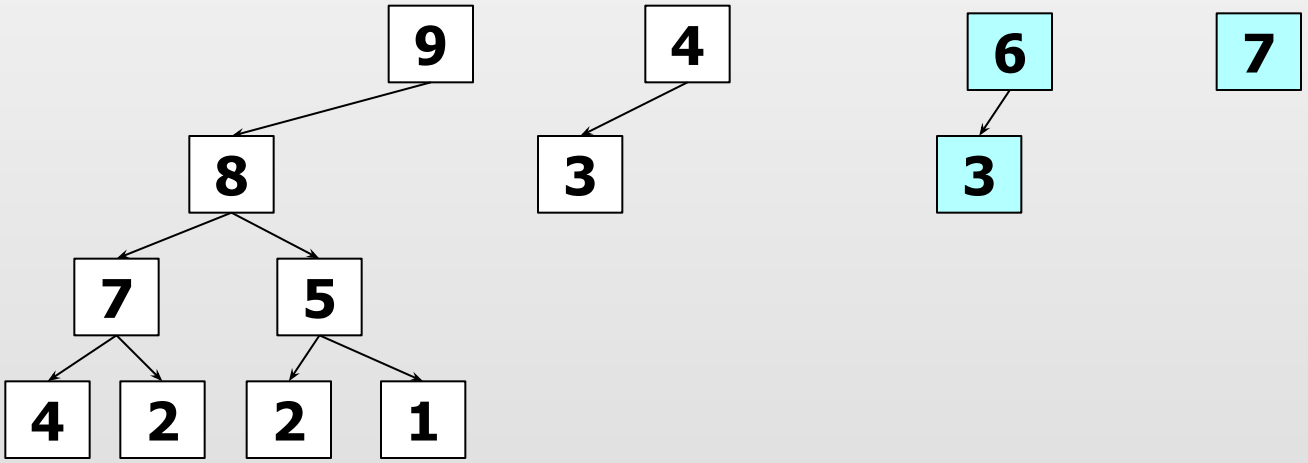
Биномиальная очередь

записать полученное дерево в дерево 2^{i+1} переноса.

i=i+1 кц

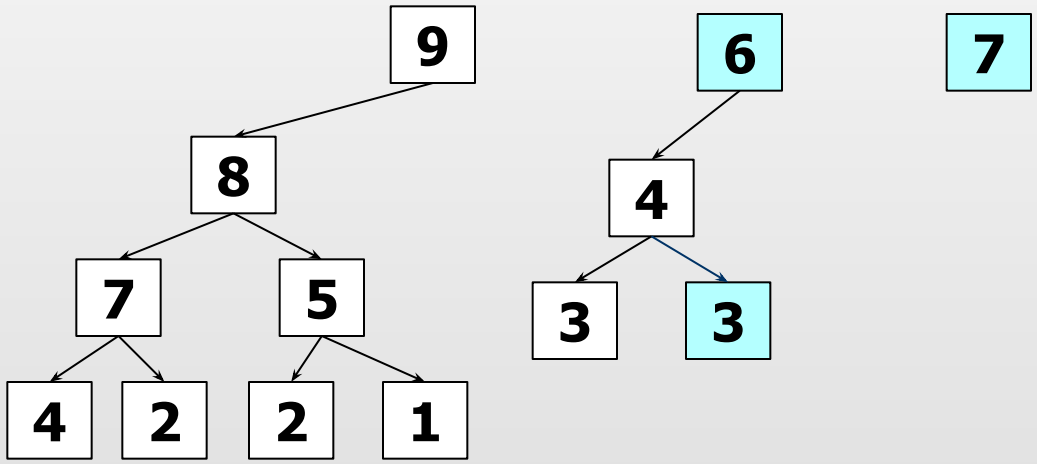
Биномиальная очередь

аналог операции + $1010_2 + 11_2 = 1101_2$



Объединение двух очередей

Биномиальная очередь



Объединение двух очередей