



ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование



Меня хорошо видно && слышно?

Ставьте ☐+, если все хорошо
Напишите в чат, если есть проблемы

The background of the slide is an aerial photograph of a city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer. A network of thin, light blue lines connects various points across the blue area, creating a web-like pattern. The title text is centered within this blue area.

Деревья и кучи

Приходько Роман

Правила вебинара



Активно участвуем



Задаем вопрос в чат или голосом



Off-topic обсуждаем в Slack #канал группы или #general



На вопросы отвечаю в конце секций

Не забыть включить запись!



Смысл | Зачем вам это уметь

1

Понимание **базовой теории деревьев** и кучи поможет изучать более оптимальные алгоритмы работы с данными

2

Многие **программы на C#** не обходятся без использования деревьев

3

Навык работы с **бинарным деревом поиска** позволяет перейти к знакомству с более сложными алгоритмами

Маршрут вебинара

Общая теория деревьев



Бинарное дерево поиска



Куча



1

Общая теория деревьев

Игра «Да/Нет»

Один из игроков загадывает слово, остальные задают вопросы на которые можно отвечать только "да" и "нет"

Игра «Да/Нет»

Стратегии:

1. Перебирать все что приходит в голову

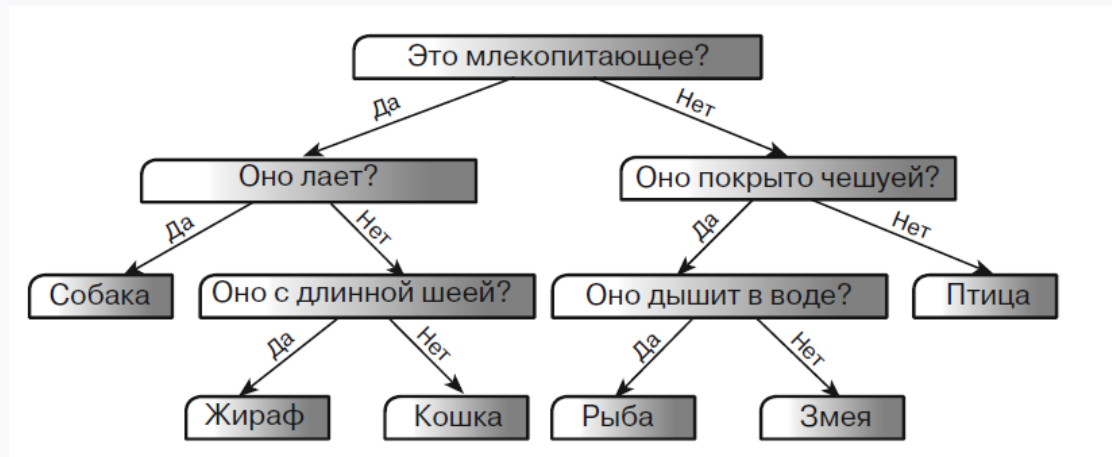
Утка? Гусь? Кошка?

Эквивалентна перебору массива в поиске элемента

2. Задавать вопросы, отсекающие целые множества вариантов

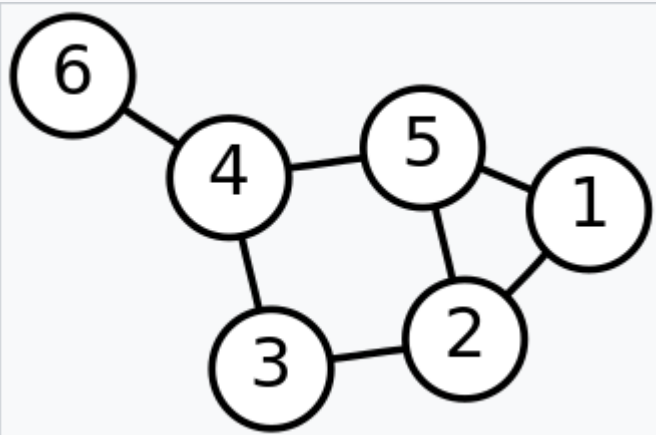
Живое? Может летать? Хищное?

Эквивалентно поиску элемента в древовидной структуре



Вторая стратегия более эффективна

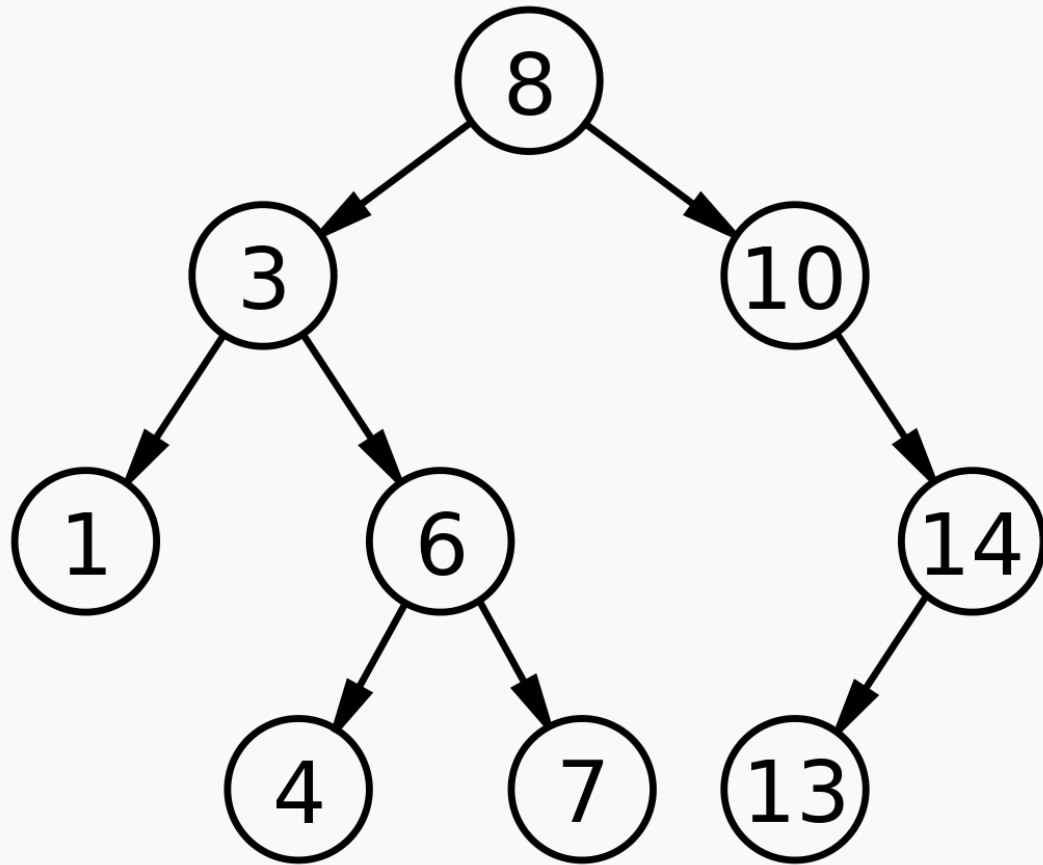
Терминология



Граф – структура, состоящая из набора объектов, имеющих связи

Объекты называются вершинами (узлами)
Связи называются ребрами

Терминология



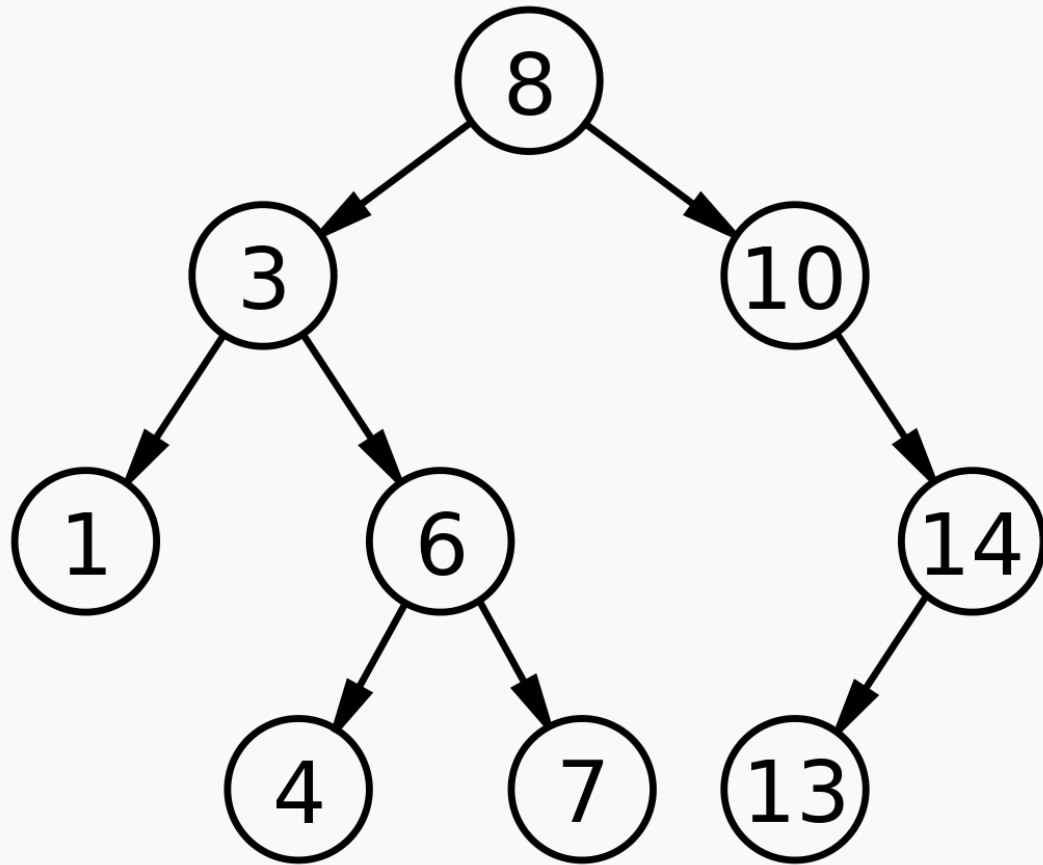
Дерево – ориентированный связный ациклический граф

Вершины (узлы) соединяются ветвями (ребрами). Направление от родителя к дочерней вершине обозначается стрелкой

Число ребер = (число вершин) – 1

Между любыми парами вершин один и только один путь

Терминология



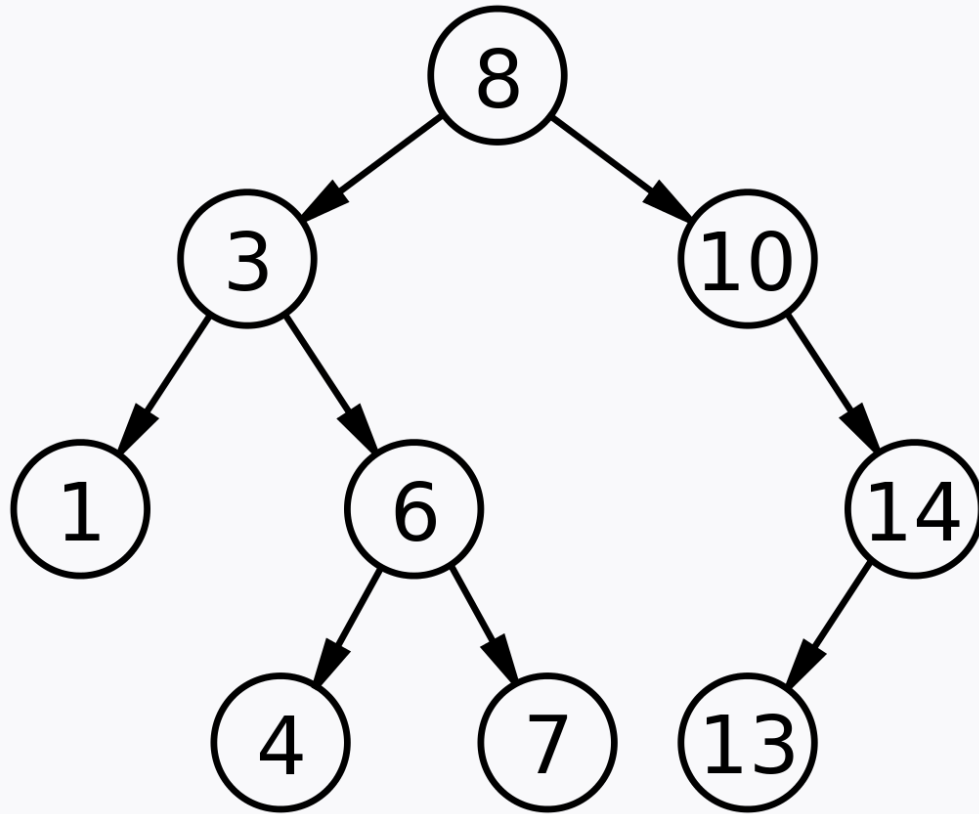
Родительские и дочерние
вершины

Сестры – две вершины с общим
родителем

Потомки – дочерние вершины
дочерних вершин

Предки – родительские вершины
родительских вершин

Терминология

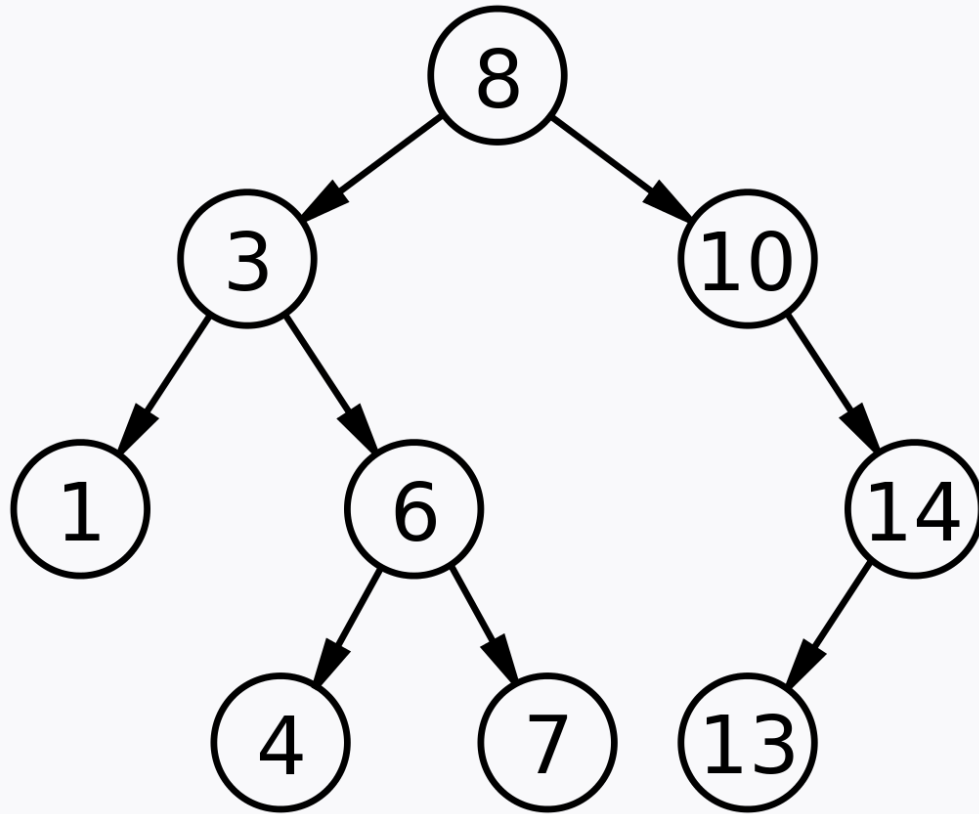


Степень вершины – количество дочерних узлов у вершины

Степень дерева – максимальная степень входящих в него вершин

Бинарное дерево – дерево степени 2

Терминология

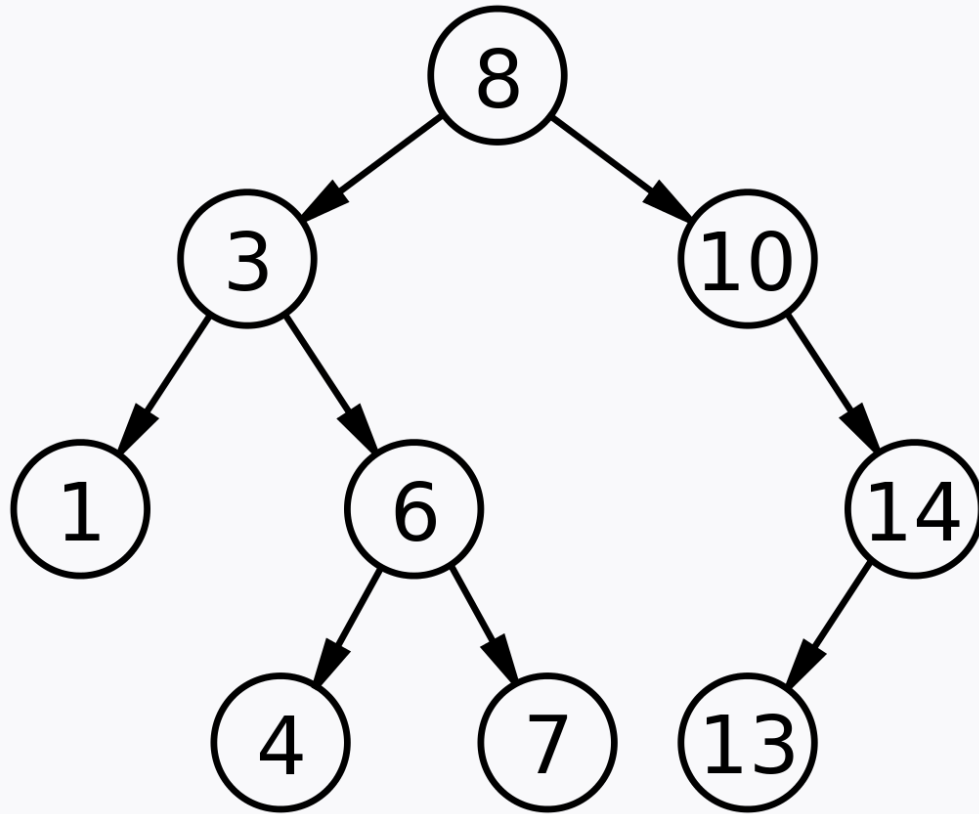


Терминальная (листовая) вершина – вершина без дочерних узлов

Внутренняя вершина – вершина, содержащая хотя бы 1 дочерний узел

Корневая вершина (корень) – вершина без родителей

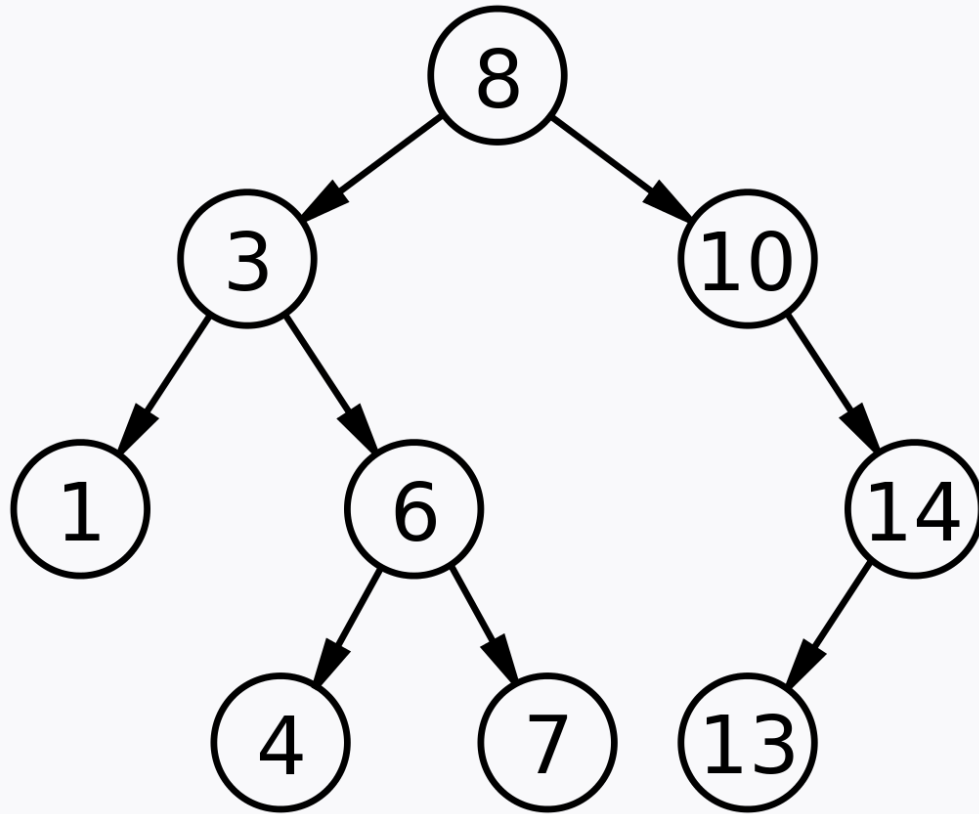
Терминология



Один единственный узел – тоже дерево, этот узел и корень, и терминальный.

В любом дереве, где больше 1 вершины, можно выделить поддерево.

Терминология

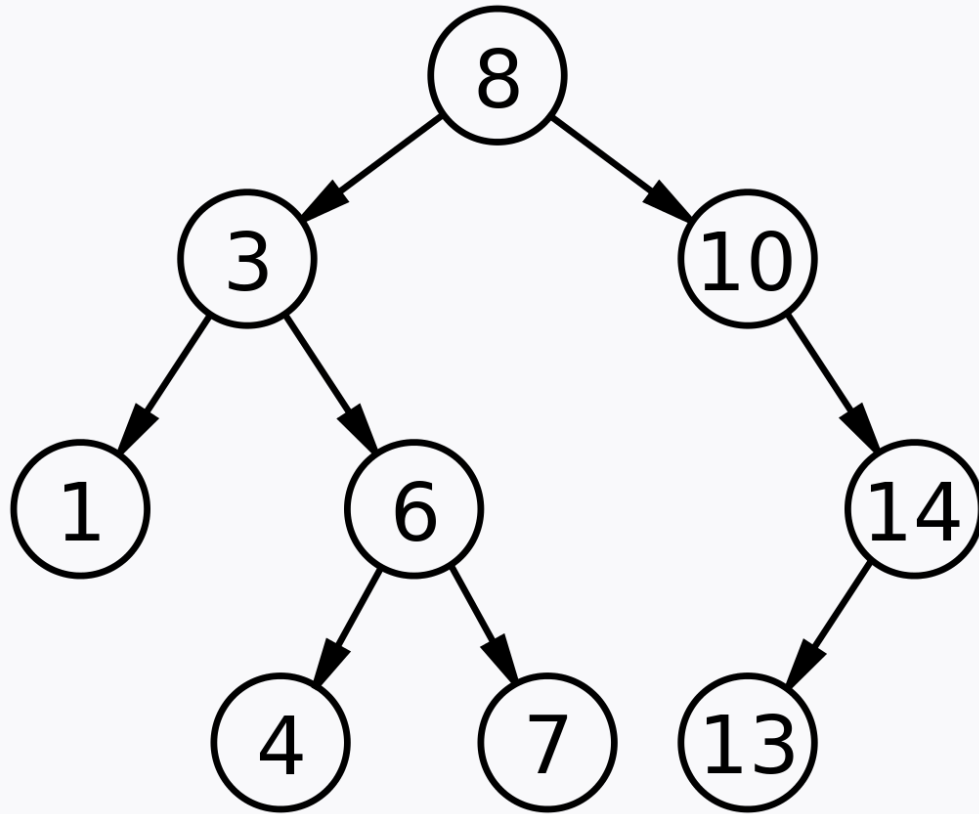


Глубина (уровень) вершины – количество ветвей от этой вершины до корня. Глубина корня = 0

Высота вершины – количество ветвей от нее до листа по самому длинному пути

Высота дерева = высота корневой вершины

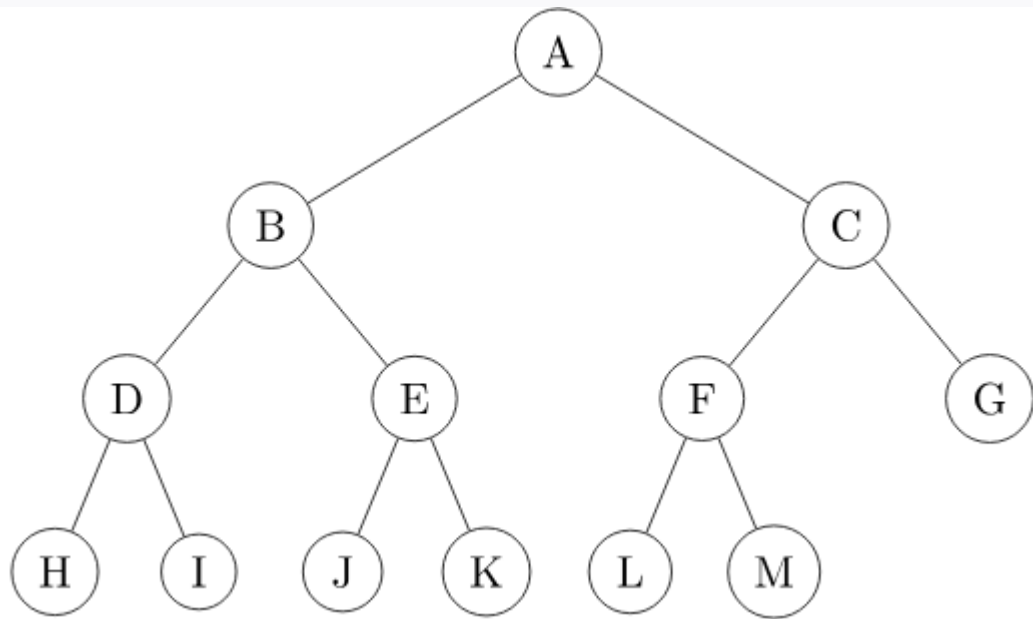
Терминология



В упорядоченном дереве
расположение дочерних деревьев
имеет значение

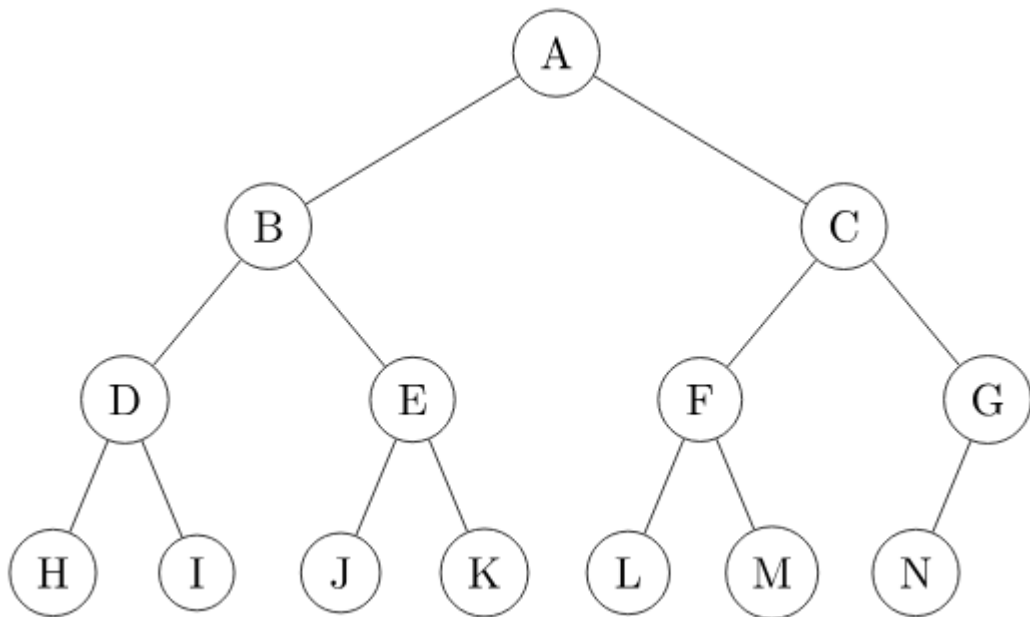
На практике деревья обычно
упорядочены

Терминология



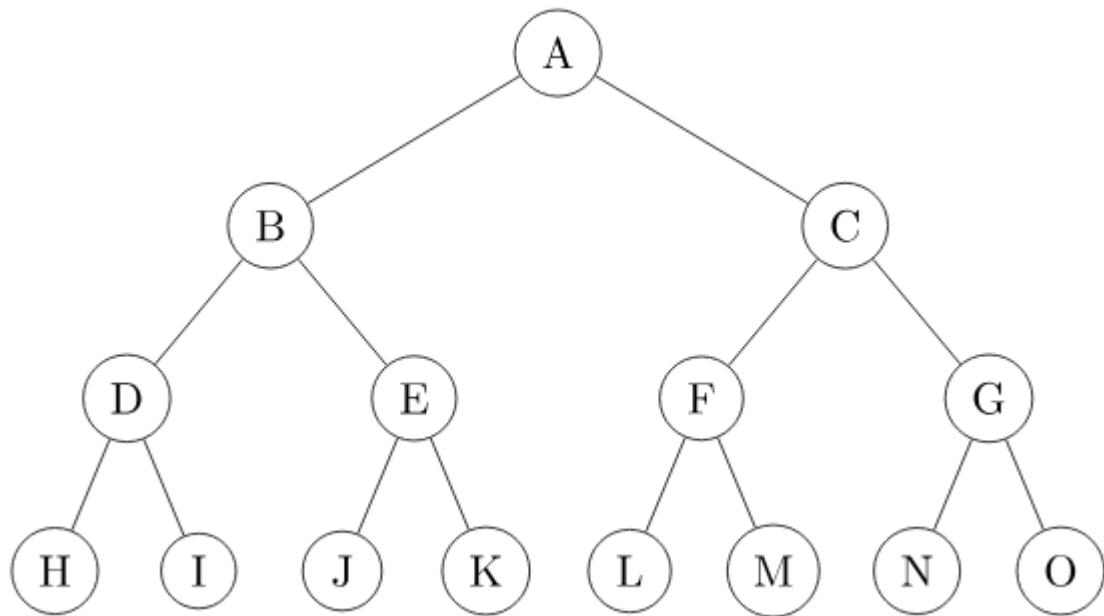
Полное дерево – дерево, в котором любая вершина не имеет дочерних или их количество равно степени дерева

Терминология



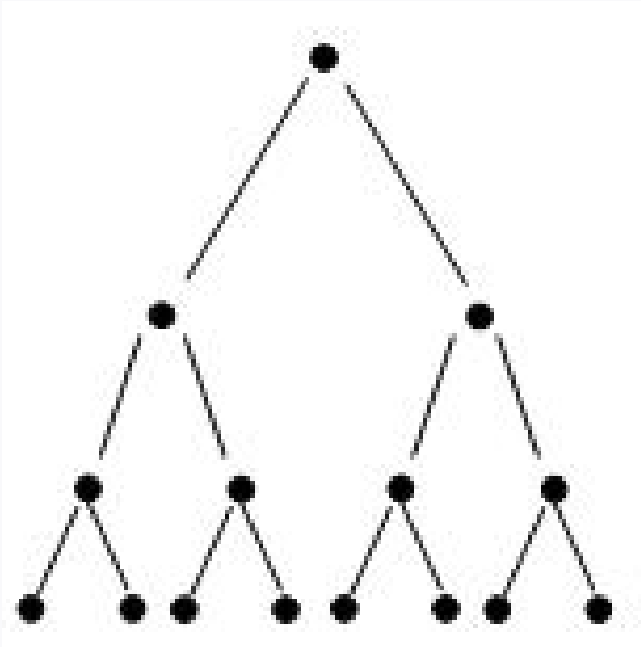
Завершенное дерево – дерево, в котором заполнены все уровни кроме, возможно, самого нижнего, где вершины сдвинуты влево

Терминология



Идеальное бинарное дерево – дерево, в котором все внутренние вершины имеют по две дочерних и все листья находятся на одном уровне

Свойства бинарного дерева



Кол-во ветвей $V = N - 1$, где N - кол-во вершин

В идеальном бинарном дереве:

$N = 2^{H+1} - 1$, где H - высота

$H = \log_2(N + 1) - 1$

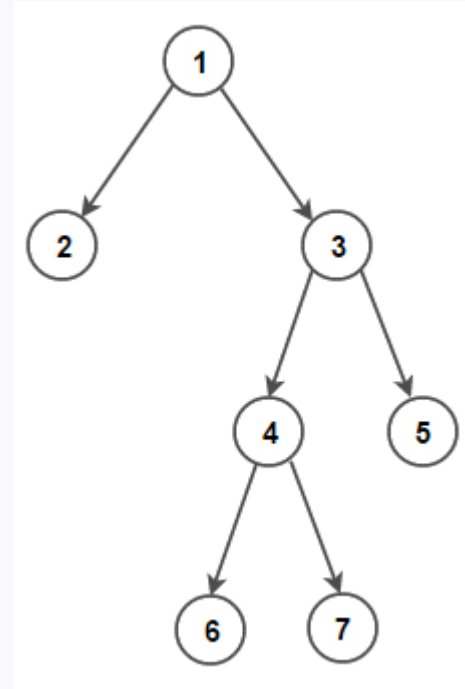
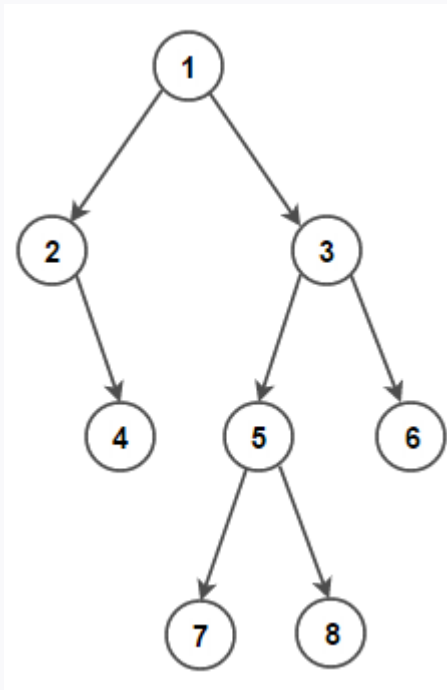
кол-во терминальных вершин $L = \underline{2^H}$

кол-во нетерминальных вершин: $N - L = 2^{H+1} - 1 - 2^H = 2^H(2 - 1) - 1 = \underline{2^H - 1}$

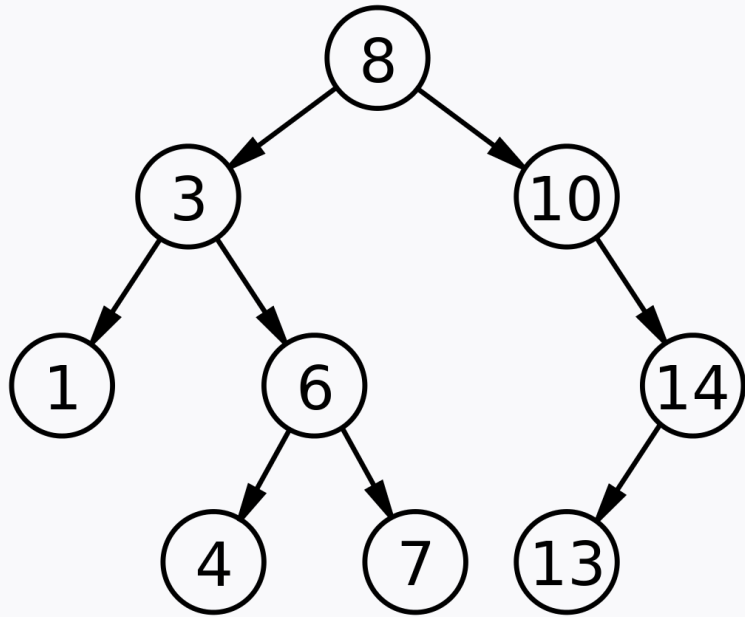
проход от корня до листьев за $O(\log_2 N)$

Сбалансированное дерево

Сбалансированное дерево – дерево, в котором для каждого узла высоты его подузлов отличаются не более чем на 1



Алгоритмическая сложность



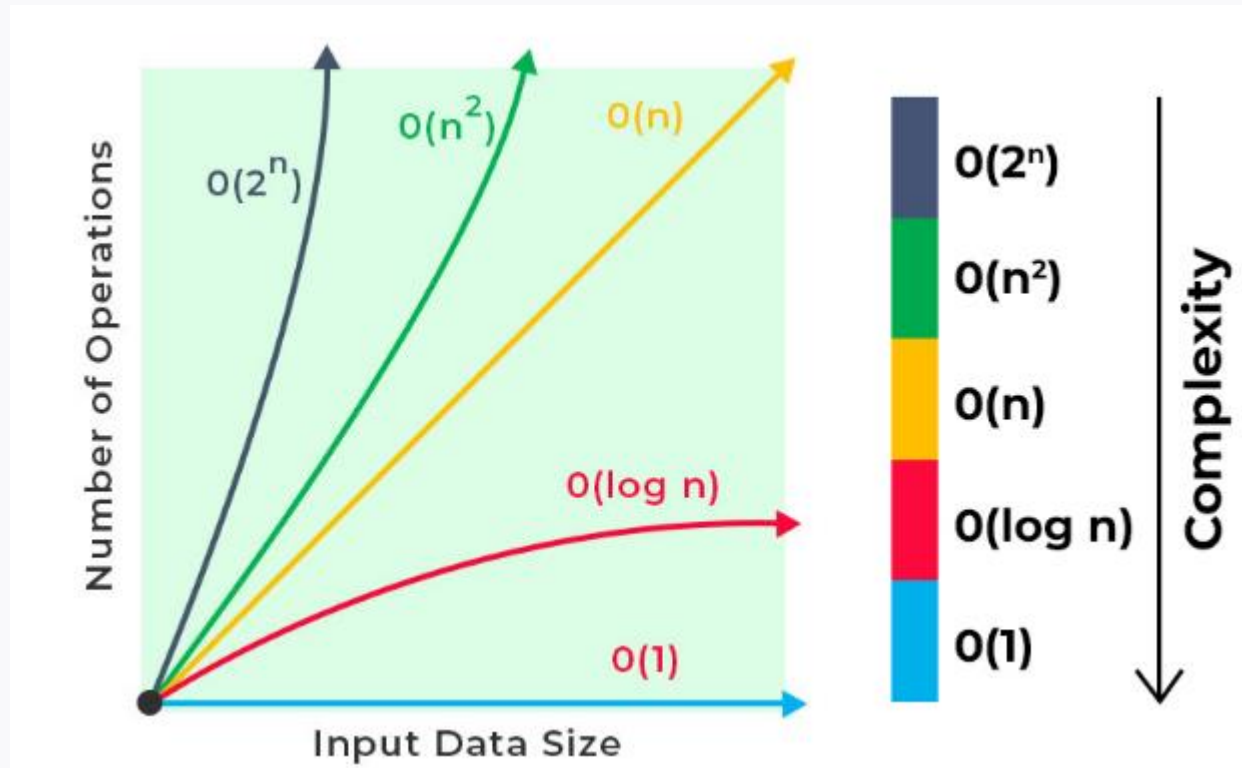
Проход от корня до листьев

- для сбалансированных деревьев за $O(\log N)$

- для несбалансированных - $O(N)$

Проход всего дерева - $O(N)$

Почему алгоритмическая сложность – это важно?



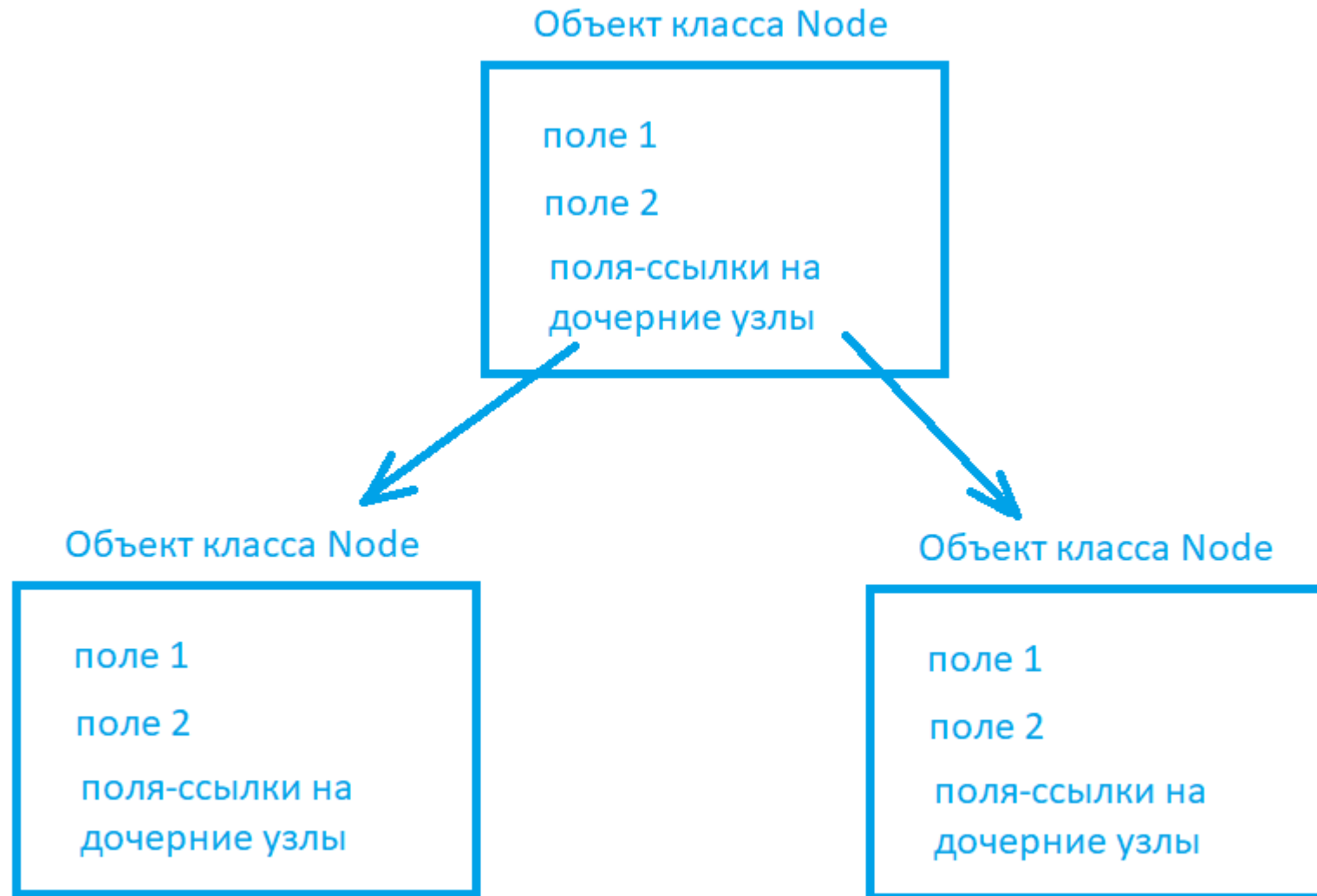
Почему алгоритмическая сложность – это важно?

размер сложность	10	20	30	40	50	60
n	0,00001 сек.	0,00002 сек.	0,00003 сек.	0,00004 сек.	0,00005 сек.	0,00005 сек.
n^2	0,0001 сек.	0,0004 сек.	0,0009 сек.	0,0016 сек.	0,0025 сек.	0,0036 сек.
n^3	0,001 сек.	0,008 сек.	0,027 сек.	0,064 сек.	0,125 сек.	0,216 сек.
n^5	0,1 сек.	3,2 сек.	24,3 сек.	1,7 минут	5,2 минут	13 минут
2^n	0,0001 сек.	1 сек.	17,9 минут	12,7 дней	35,7 веков	366 веков
3^n	0,059 сек.	58 минут	6,5 лет	3855 веков	2×10^8 веков	$1,3 \times 10^{13}$ веков

Как можно представить дерево в программе

- Через объекты и ссылки между ними
 - Узлы – объекты, ветви – ссылки между объектами
- Через массивы

Представление узлов с помощью объектов



Представление дерева в виде массива

- Через массивы (эффективен только для ~полных деревьев)

1. Корневой узел записывается 0м элементом массива

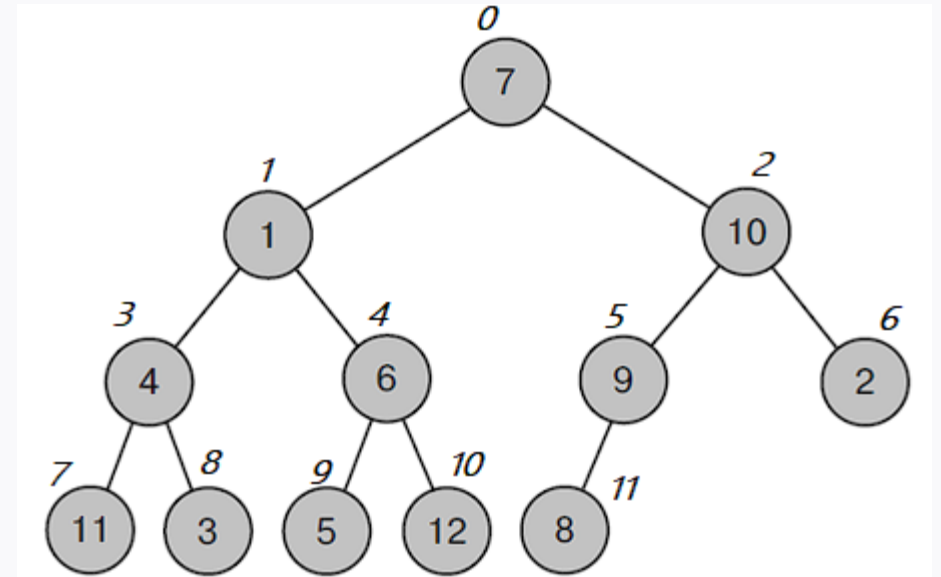
2. Его дочерние узлы - 1 и 2 элементы массива

3. Далее номера узлов в массиве рассчитываются по формулам

- $2i+1$ (левый дочерний)

- $2i+2$ (правый дочерний)

где i - номер узлового в массиве



0	1	2	3	4	5	6	7	8	9	10	11
7	1	10	4	6	9	2	11	3	5	12	8

Операции

Операции над произвольными деревьями

1.Обход

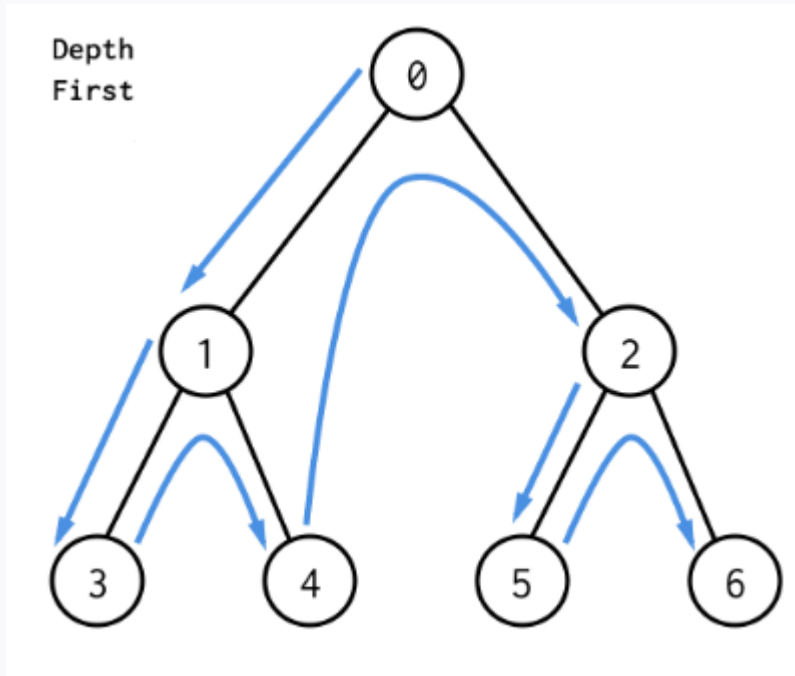
Что такое обход дерева

«Перемещение» по всем вершинам дерева в определенном порядке, чтобы выполнить с каждой вершиной некоторое действие.

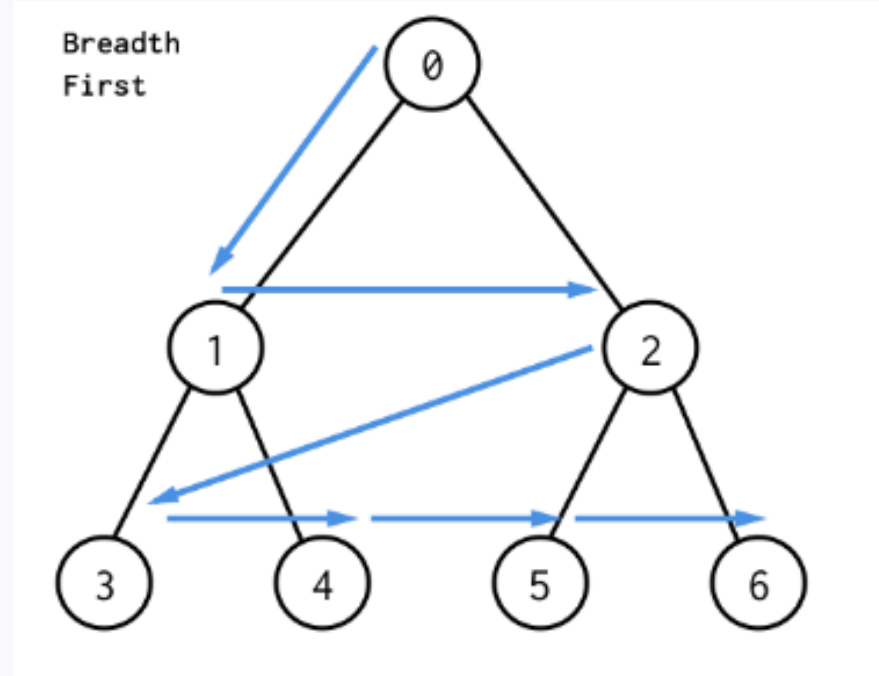
Зачем может понадобиться:

- Перечислить все вершины
- Найти вершину по какому-то критерию
- Найти вершину, к которой добавить новую дочернюю вершину
- И т.д.

Варианты обхода дерева



Реализуется через рекурсию
 $O(n)$
Не требует расходов памяти



Реализуется через очереди
 $O(n)$
Требует расходов памяти

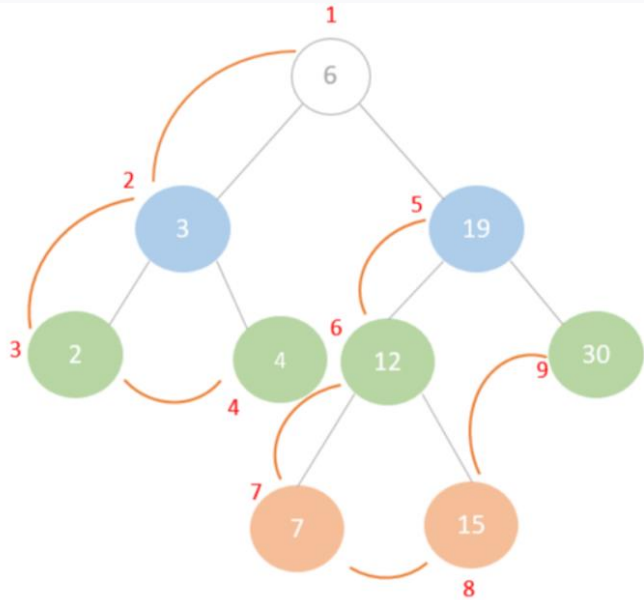
Варианты обхода поддерева

Pre-order: вершина -> левый -> правый

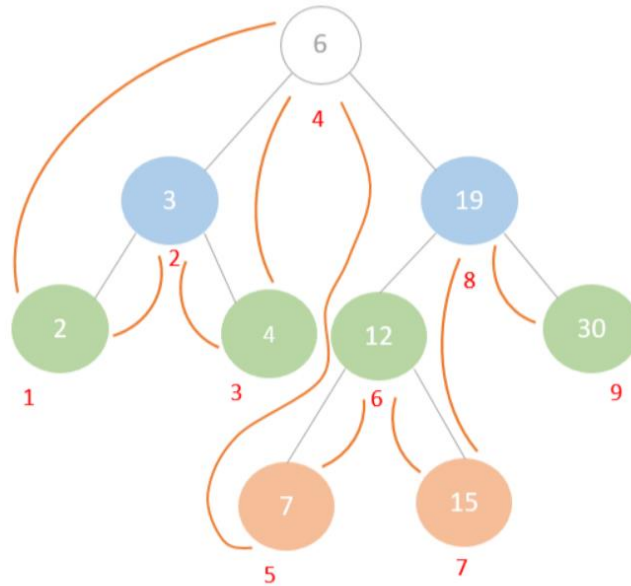
In-order: левый -> вершина -> правый

Post-order: левый -> правый -> вершина

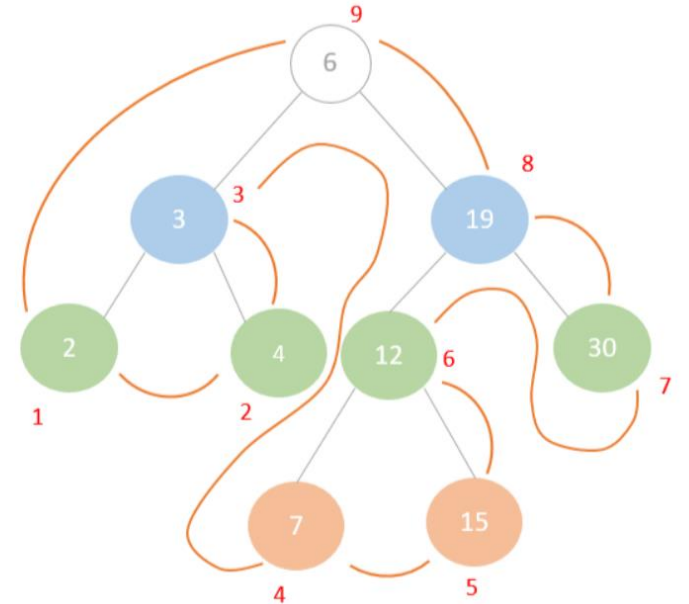
Варианты обхода дерева



Pre-order



In-order



Post-order

Алгоритмическая сложность одинакова

In-order для BST извлекает в отсортированном порядке

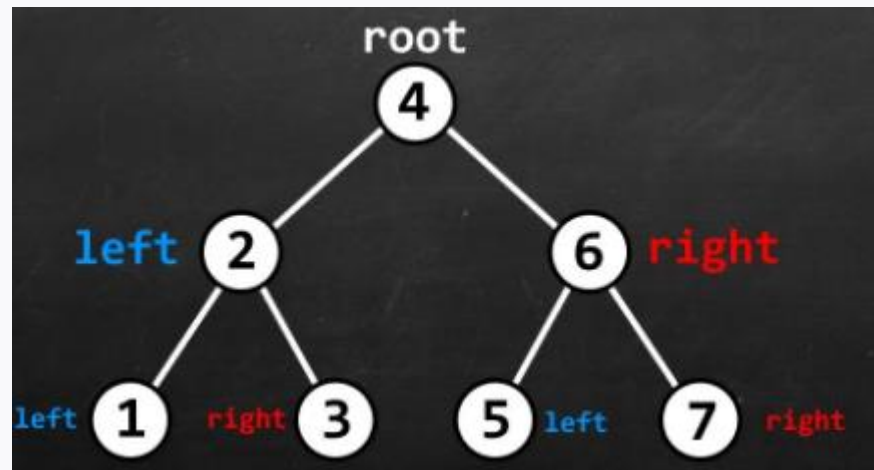


2 Упорядоченные деревья

Что такое упорядоченное дерево

Упорядоченное дерево - дерево, построение которого выполнено в соответствии с некоторым порядком

Бинарное дерево поиска (Binary Search Tree, BST) – бинарное упорядоченное дерево, в котором содержимое узла больше содержимого левого подузла и не больше содержимого правого подузла



Операции над BST

Операции над BST:

1. Совершить обход
2. Добавить элемент
3. Найти элемент
4. Удалить элемент

Операции над BST

Добавить элемент

1. Двигаясь от корня поочередно сравниваем элемент с вершинами и добавляем в виде листа

Операции над BST

Найти элемент

1. Двигаясь от корня поочередно сравниваем искомый элемент с вершинами доходим таким образом до цели

Операции над BST

Удалить вершину

1. Лист удаляется без дополнительных действий
2. Вершина с одним дочерним удаляется, при этом создается ветвь от ее родителя к ее дочернему напрямую
3. Вершина с двумя дочерними меняется местами или с наименьшей дочерней справа или с наибольшей дочерней слева. Далее эта дочерняя удаляется (пункт 1)

Алгоритмическая сложность операций в BST

	Сбалансированное	Несбалансированное
Поиск	$O(\log N)$	$O(N)$
Добавление	$O(\log N)$	$O(N)$
Удаление	$O(\log N)$	$O(N)$



Что такое куча

Куча (англ. Heap) – древовидная структура данных, в которой для любой пары родительского (Р) и дочернего (Д) узла выполняется неравенство:

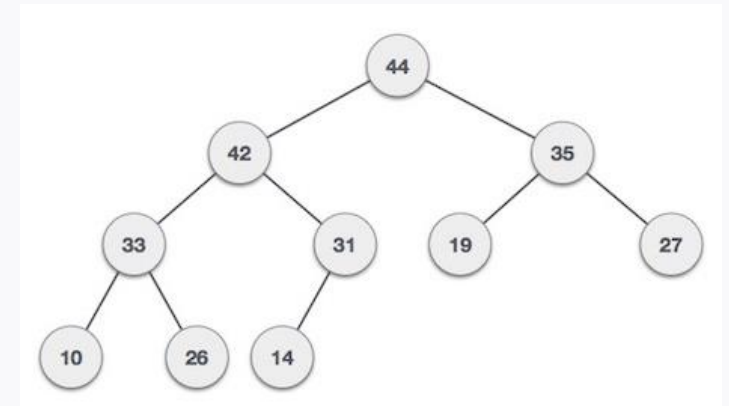
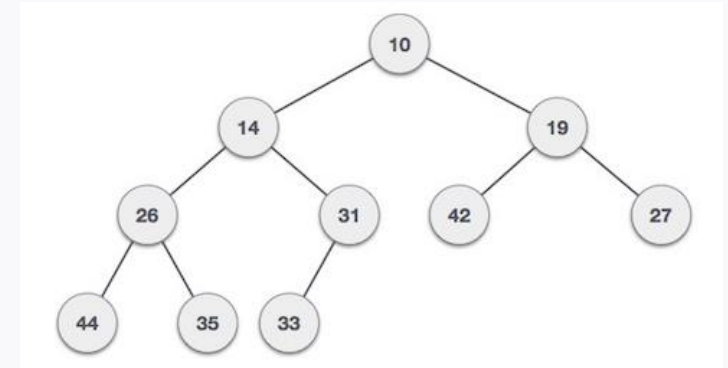
Значение в Р \geq значения в Д

(Значение в Р \leq значения в Д)

В первом случае куча называется

максимальной,

во втором - минимальной

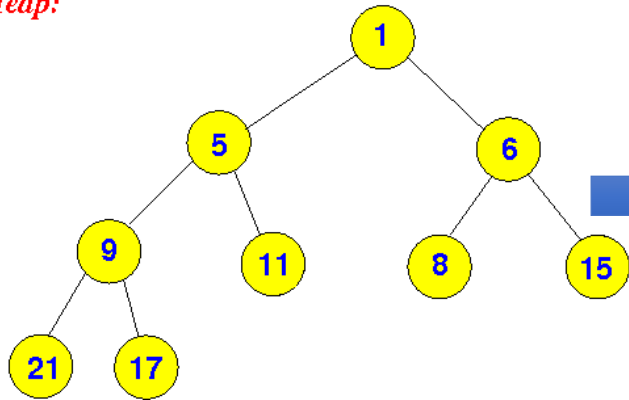


Используемые операции

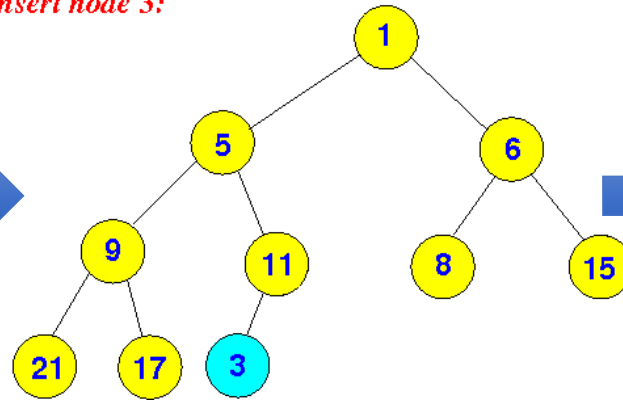
- Добавление элементов
- Извлечение верхнего элемента

Добавление нового элемента

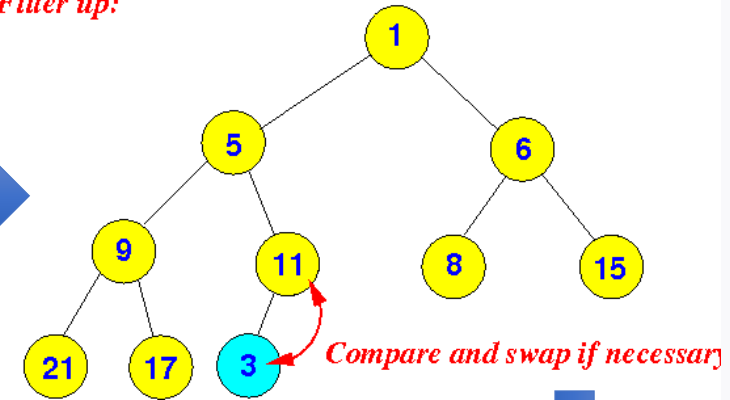
Heap:



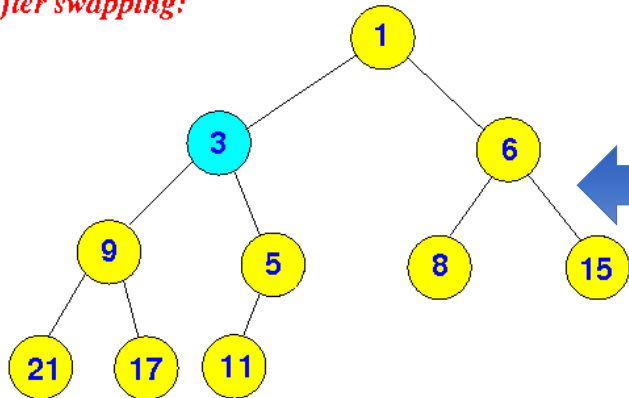
Insert node 3:



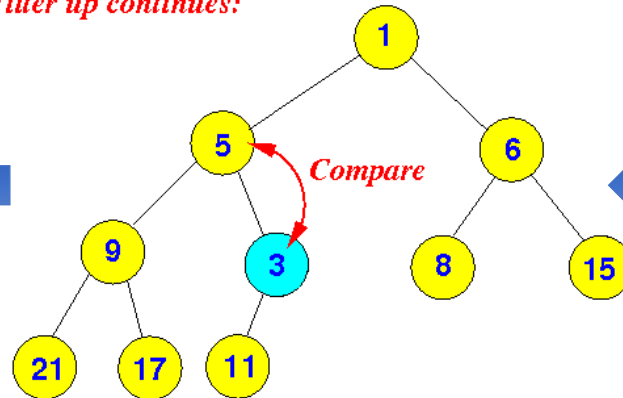
Filter up:



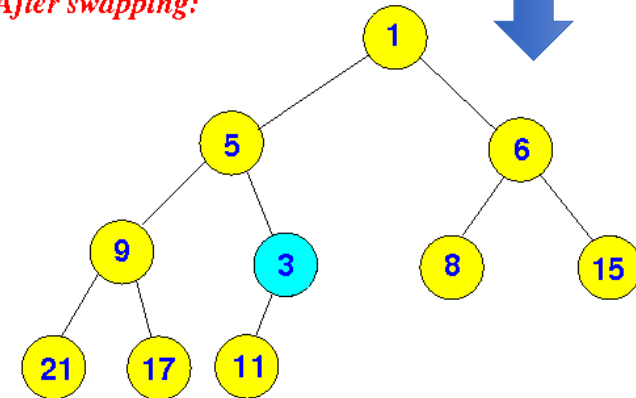
After swapping:



Filter up continues:



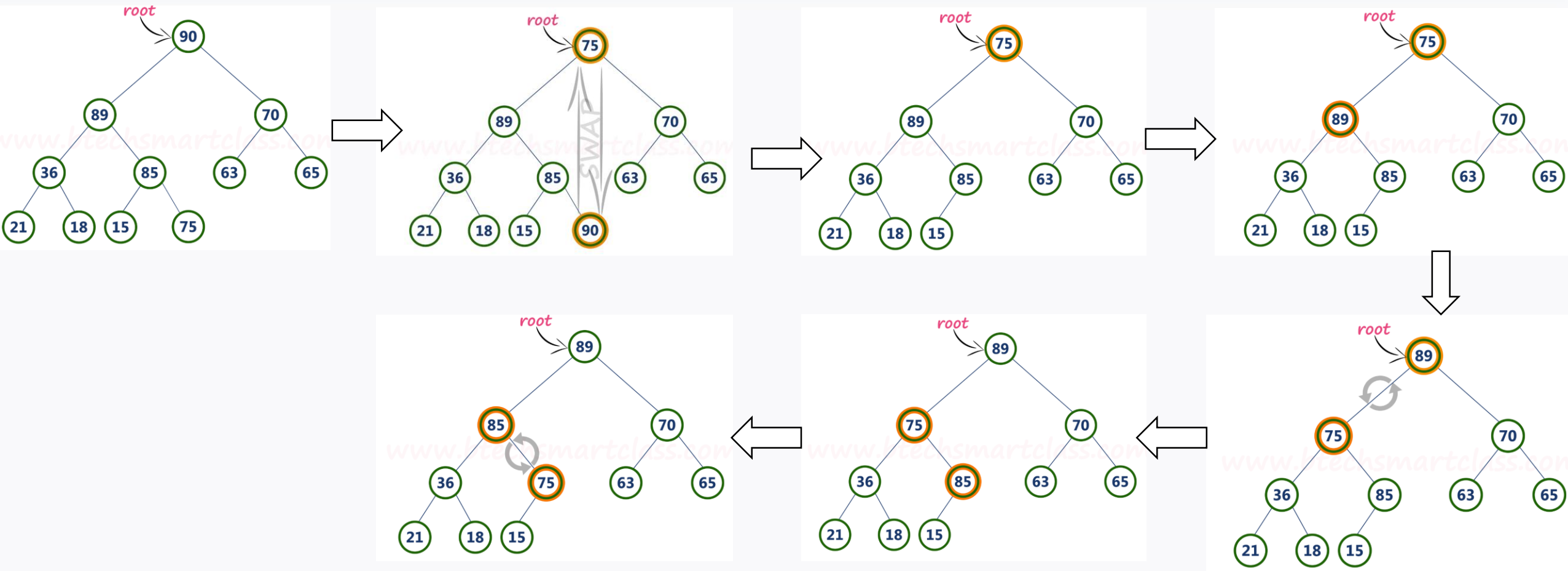
After swapping:



1. Добавить элемент в конец текущего уровня кучи

2. Перестроить кучу поочередно меняя местами элементы в случае необходимости

Удаление корневого элемента



1. Поменять местами корневой элемент и последний
2. Удалить последний элемент
3. Перестроить кучу (в случае необходимости) начиная с нового корневого элемента

Алгоритмическая сложность операций

1. Удаление корневого элемента - $O(\log N)$
2. Добавление нового элемента - $O(\log N)$

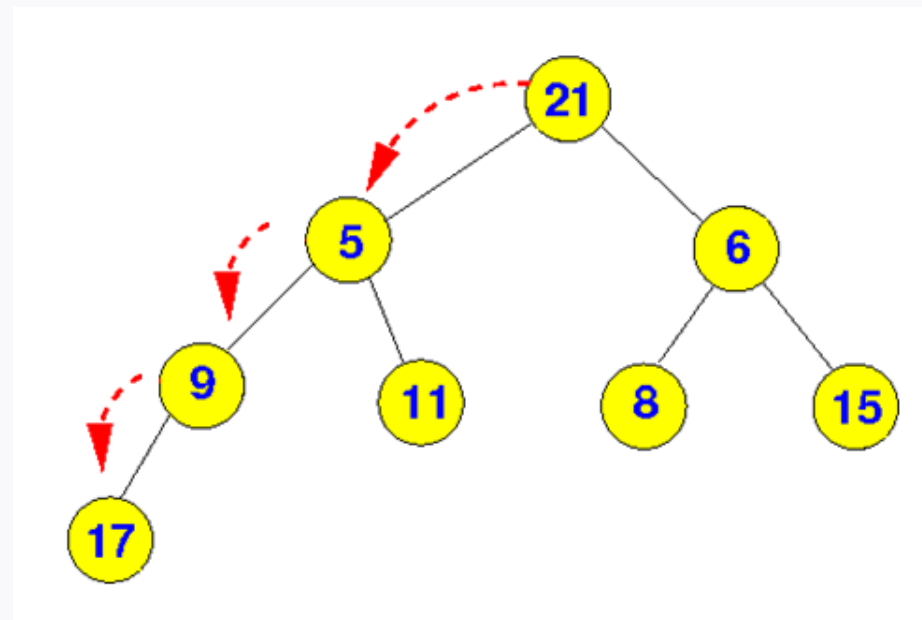
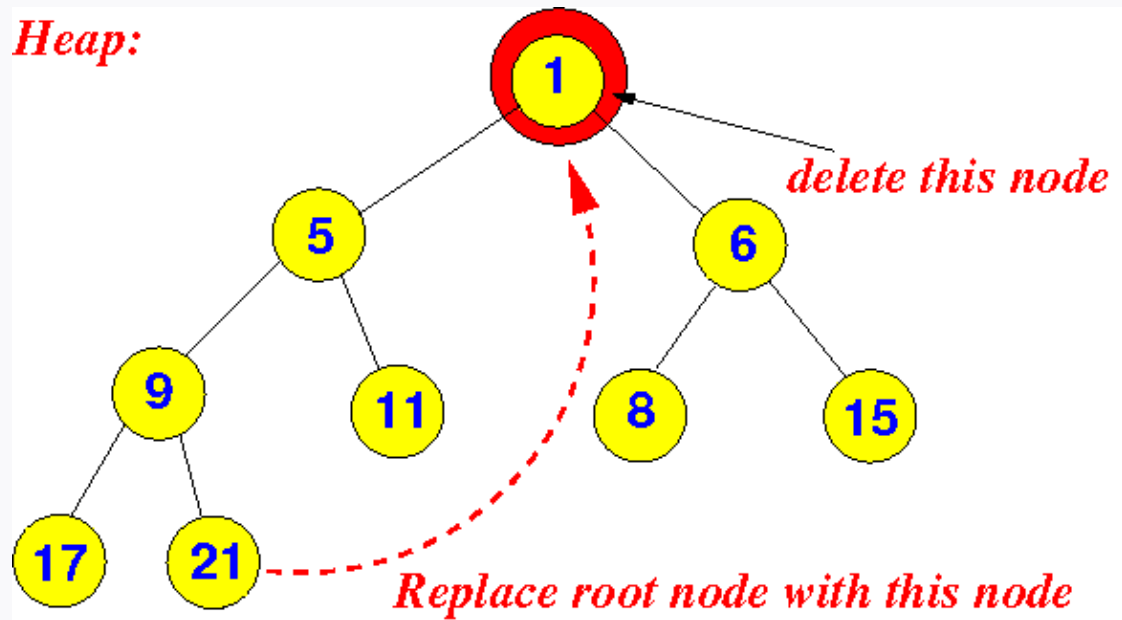
Применения кучи

- Очереди с приоритетом
- Пирамидальная сортировка

Очередь с приоритетом

Реализуется поочередным удалением корневого элемента

Heap:



Пирамидальная сортировка (Heap Sort)

У вас есть массив или список объектов, которые хочется отсортировать

1) Постройте из них кучу, последовательно добавляя каждый элемент в кучу.

Сложность $O(N \cdot \log N)$, т.к. N элементов добавляем в кучу, а одна операция добавления в кучу – $O(\log N)$

2) Последовательно удалением корней этой кучи постройте новый список. Он будет уже отсортирован согласно определению Кучи.

Сложность $O(N \cdot \log N)$, т.к. N элементов удаляем из кучи, а одна операция удаления из кучи – $O(\log N)$

Итоговая сложность: $O(N \cdot \log N)$

Пирамидальная сортировка (Heap Sort)

Реализация алгоритма для работы с массивом как с кучей:

1. Исключив из перебора узлы последнего уровня, пройти остальные узлы в обратном порядке с последовательностью действий для каждого:

а) выбрать узел с максимальным значением из тройки: узел, его левый и его правый дочерние узлы (УМ)

б) если значение УМ не совпадает со значением текущего узла (ТУ), поменять ТУ и УМ местами

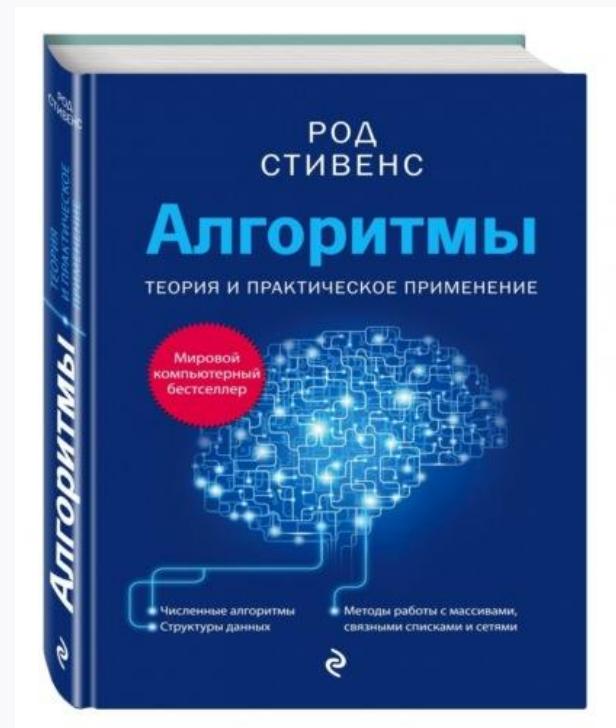
в) повторить пункты а)-б) для ТУ в его новой позиции

Литература

Род Стивенс

«Алгоритмы. Теория и практическое применение»

<https://www.litres.ru/avtor/algorithmy-teoriya-i-prakticheskoe-primenenie-2-e-izdanie-66497329/>



Домашнее задание

Инструкция в ЛК

Итоги занятия

1

Познакомились с базовой теорией работы с деревьями и кучей

2

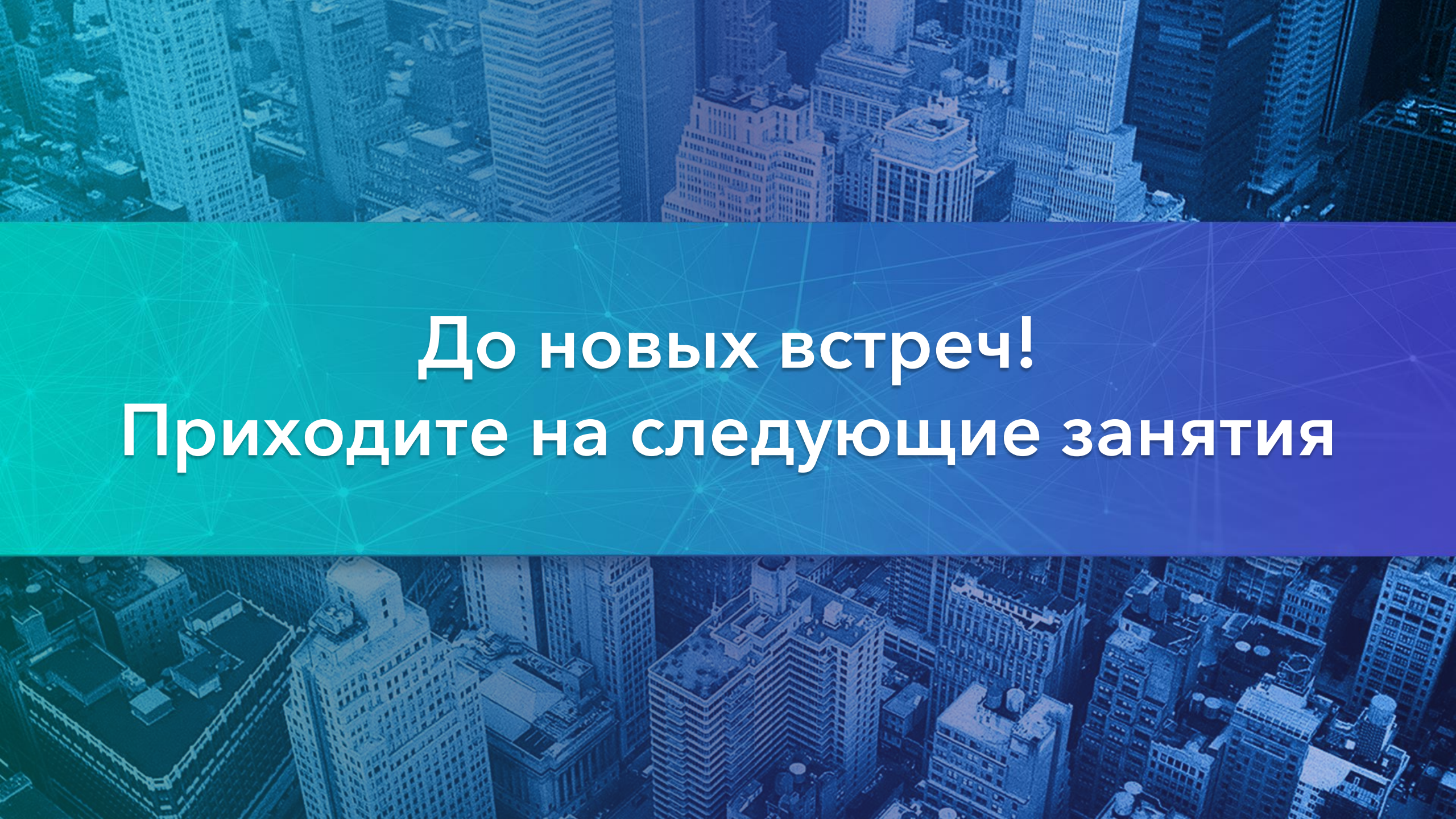
Научились строить и обходить деревья на C#

3

Научились строить на C# BST и искать в нем значение



Заполните, пожалуйста,
опрос о занятии по ссылке в чате



До новых встреч!
Приходите на следующие занятия