Идеально сбалансированное дерево

- Пусть задано ***n*** элементов.
- Требуется построить дерево минимальной глубины, содержащее эти элементы.

- Минимальная высота при заданном числе вершин достигается, если на всех уровнях, кроме последнего, помещается максимально возможное число вершин.
- Этого просто добиться, размещая приходящие вершины поровну слева и справа от каждой вершины.

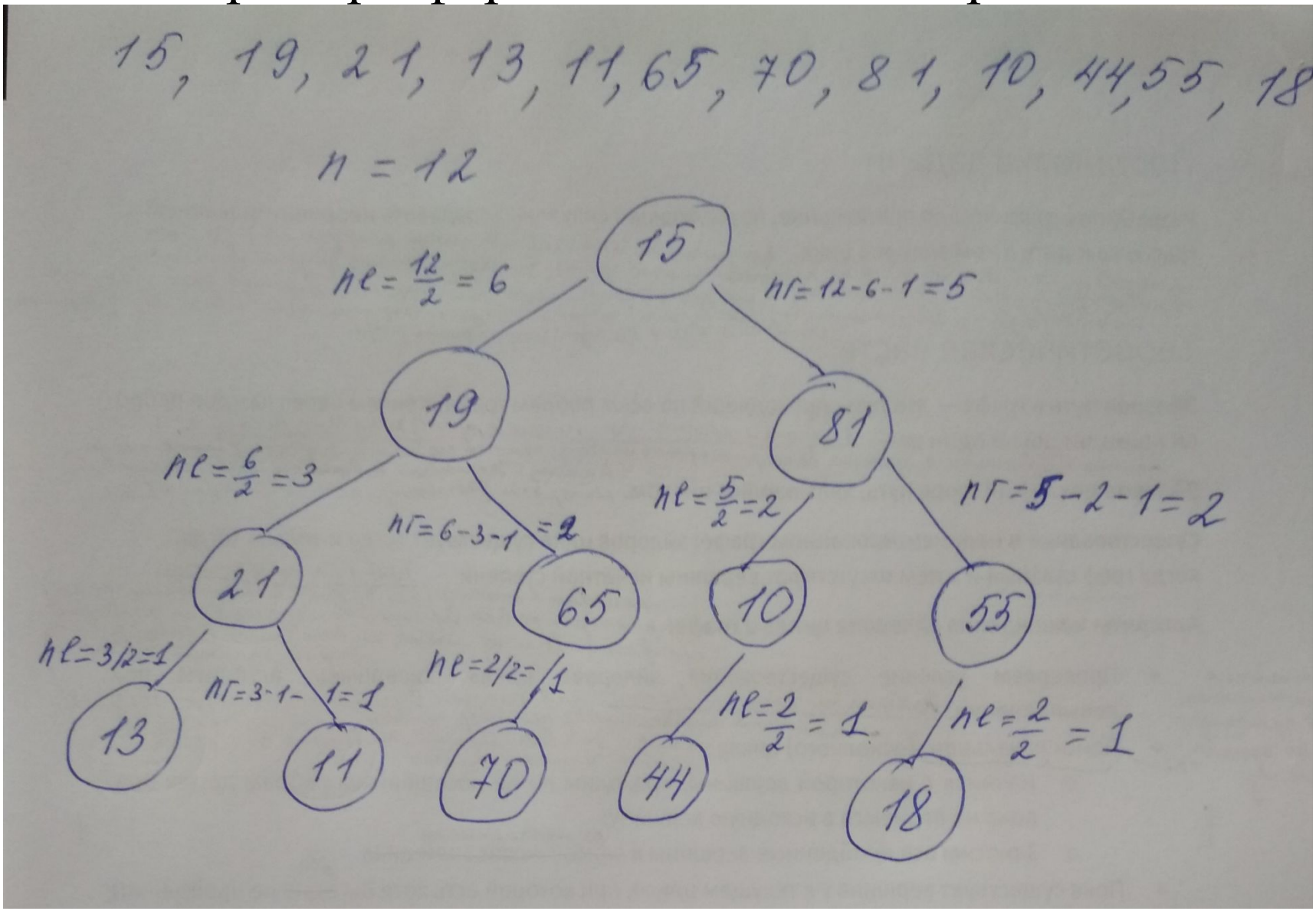
- Дерево называется ***идеально сбалансированным***, если ***число вершин в его левых и правых поддеревьях отличается не более чем на 1.***

Алгоритм построения идеально сбалансированного дерева

1. Взять одну вершину в качестве корня.
2. Построить тем же способом левое поддерево с ***nl = n /2*** вершинами.
3. Построить тем же способом правое поддерево с ***nr = n-nl-1*** вершинами.

Заполнение идеально сбалансированного дерева осуществляется согласно прямого метода прохождения.

Пример оформления домашней работы



РЕКУРСИВНЫЕ МЕТОДЫ ПРОХОЖДЕНИЯ ДЕРЕВЬЕВ

Прямой метод прохождения определяется посещением узла в первую очередь и последующим прохождением сначала левого, а потом правого поддеревьев.

Порядок операций дает так называемое ***NLR*** (node, left, right,) сканирование.

1. Посещение узла (N).
2. Прохождение левого поддерева (L).
3. Прохождение правого поддерева (R).

Сначала осуществлялось прохождение по левому поддереву, а уже потом по правому. Существует алгоритм, который выбирает сначала правое поддерево и потом левое.

Порядок операций дает так называемое ***NRL*** сканирование.

Порядок операций при ***симметричном методе*** следующий:

1. Прохождение левого поддерева (L).
2. Посещение узла (N).
3. Прохождение правого поддерева (R).

Назовем такое прохождение ***LNR.*** Второй алгоритм – ***RNL*** прохождение.

Порядок операций при ***обратном методе*** следующий:

1. Прохождение левого поддерева (L).
2. Прохождение правого поддерева (R).
3. Посещение узла (N).

Назовем такое прохождение ***LRN*** . Второй алгоритм – ***RLN*** прохождение.

Варианты прохождения дерева

NLR: 15, 19,21,13,11,65,70,81, 10,44,55,18

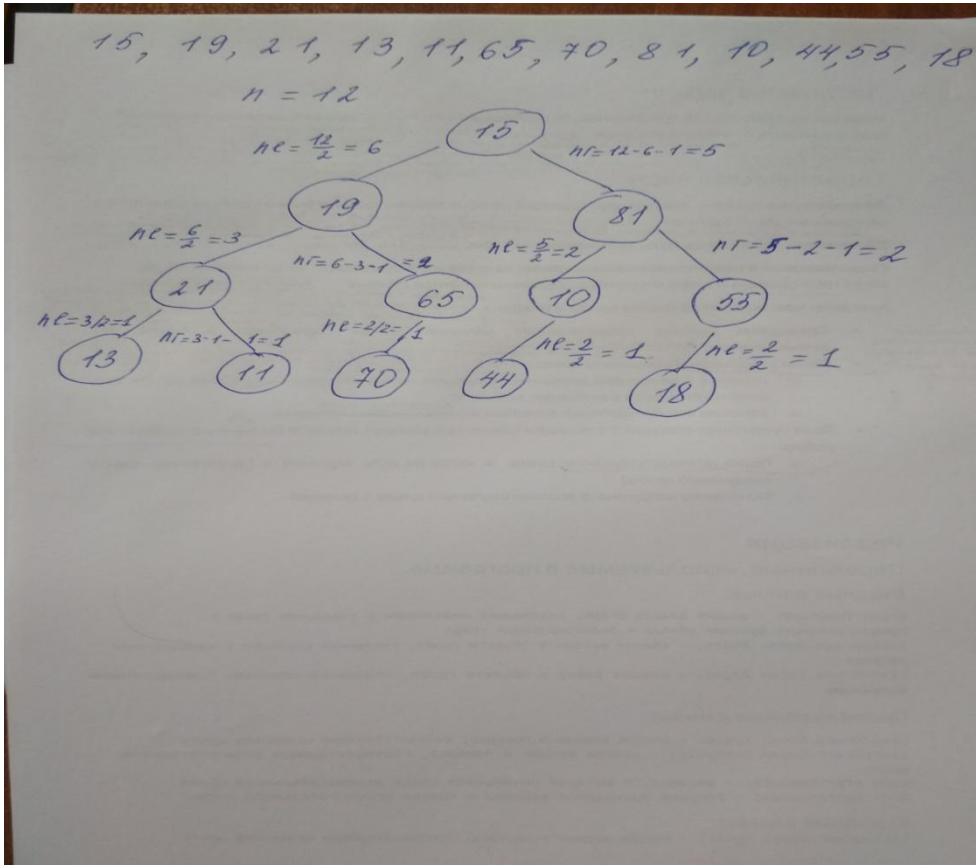
NRL: 15, 81,55, 18,10, 44, 19, 65,70, 21,11, 13

LNR: 13,21,11,19,70,65, 15, 44,10,81,18,55

RNL: 55,18,81,10,44, 15, 65,70,19,11,21,13

LRN: 13,11,21,70,65,19,44,10,18,55,81, 15

RLN: 18,55,44,10,81,70,65,11,13,21,19, 15

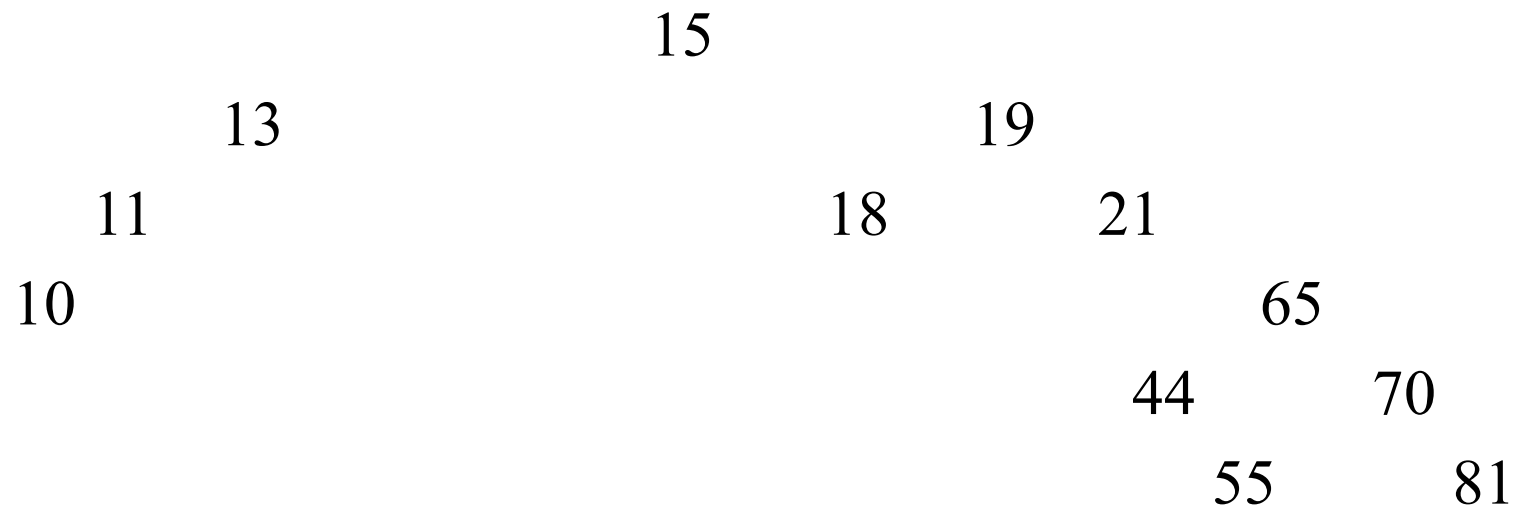


Бинарное поисковое дерево

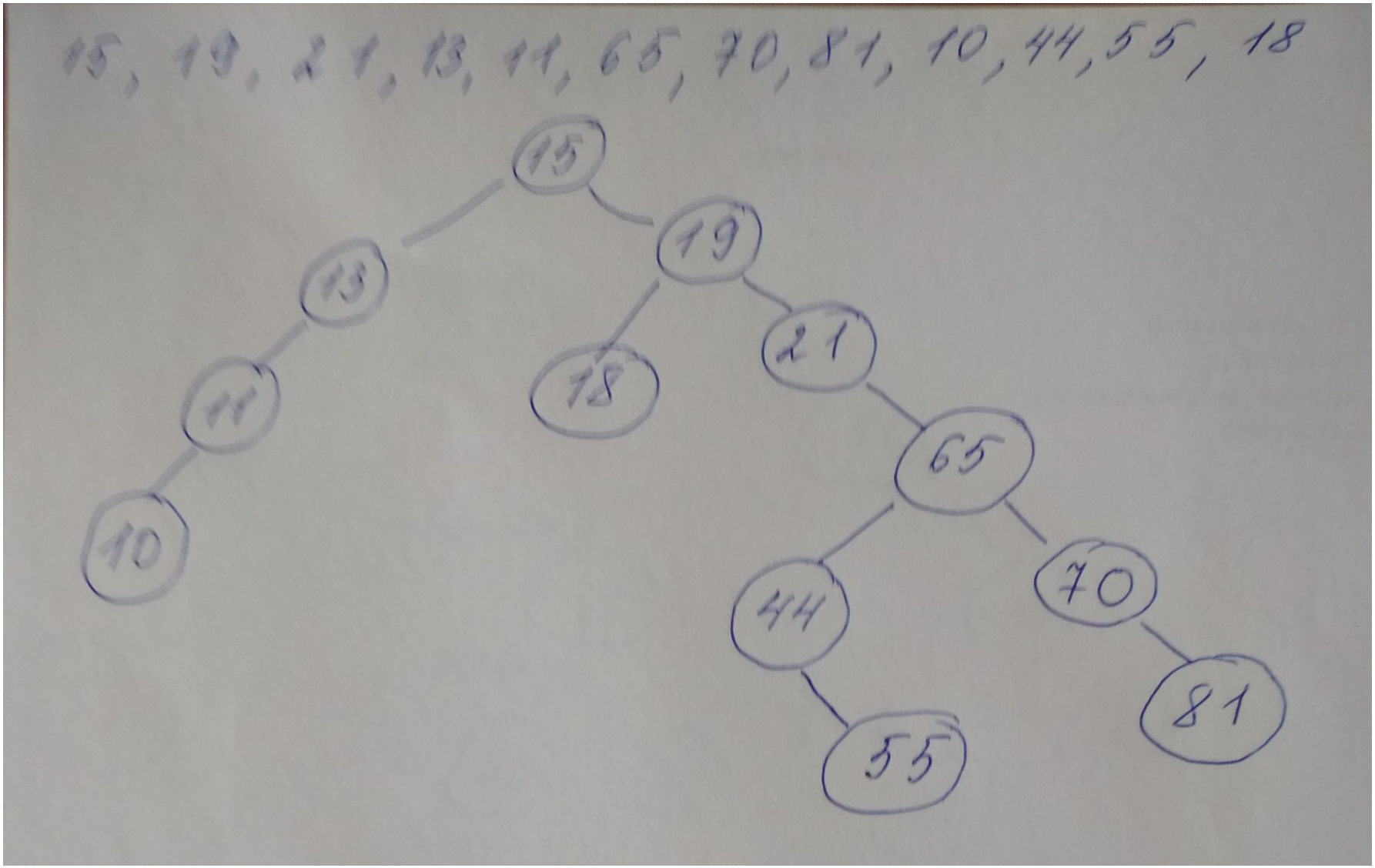
- Если дерево организовано так, что для каждой вершины t_i справедливо утверждение, что все ключи левого поддерева t_i меньше ключа t_i, а все ключи правого поддерева t_i больше его, то такое дерево будем называть ***деревом поиска*** или ***поисковым деревом***.
- Бинарное дерево поиска строится по следующему правилу: для каждого узла значения данных в левом поддереве меньше, чем в этом узле, а в правом поддереве - больше или равны.
- В таком дереве легко обнаружить произвольный ключ, для этого достаточно, начав с корня, двигаться к левому или правому поддереву на основании лишь одного сравнения с ключом текущей вершины.
- Причем структура поискового дерева заранее не определена, она меняется в ходе выполнения программы, дерево растет или сокращается.

Пример построения поискового дерева

15 19 21 13 11 65 70 81 10 44 55 18



Пример поискового дерева



Домашнее задание

1. Построить идеально сбалансированное дерево.
2. Обойти построенное дерево с помощью 6 способов прохождения.
3. Построить поисковое дерево.

Исходная последовательность:

1. **число, соответствующее дню рождения,**
2. **число, соответствующее месяцу рождения,**
3. **Номер школы,**
4. **Номер дома,**
5. **Номер квартиры**
6. **Любимое число**
7. **94 33 39 90 75 99 85 43 67**