5. Бинарные деревья поиска:

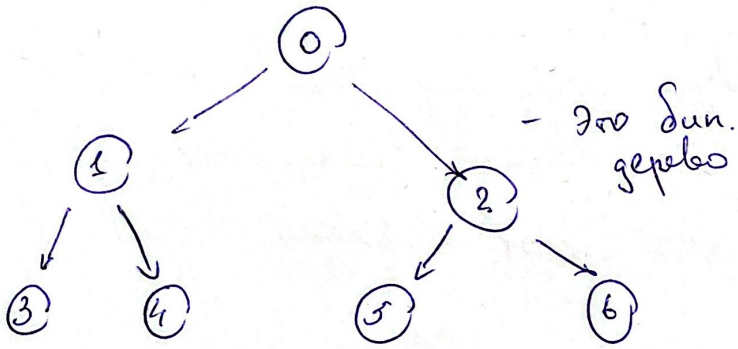
Основные операции (поиск, вставка, удаление, повороты) Оценка алгоритмической сложности выполнения основных операций.

$h_{min} = \log n$

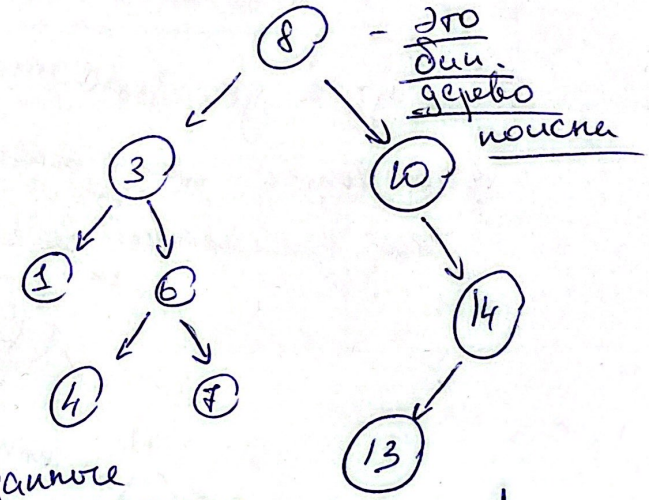
$h_{max} = n$

Бин. дерево поиска - структура данных для работы с упорядоченными множествами.

Свойство: если x - узел бин. дерева с ключом k, то все узлы в левом поддереве ~~больше~~ должны иметь ключи, меньше, чем k, а в правом больше, чем k.



- это бин. дерево



- это бин. дерево поиска

данные хранятся в отсортированном виде

Поиск узла - $O(\log n)$ - балансир.

1. Берем корень, сравниваем с искомым элементом;

a.) Если корень > искомого → направо, если ≤ налево и сравниваем след. узел

делаем так, пока не найдем дерева искомый элемент

Каждый дочерний узел бин. дерева поиска сам является деревом и благодаря такой структуре достигается $O(\log n)$ - меньше, чем в списке $O(n)$. Также стоит заметить, что такая эффективность достигается при сохранении его в сбалансированном состоянии - все уровни, кроме последнего заполнены.

Вставка узла $O(\log n) + O(1)$

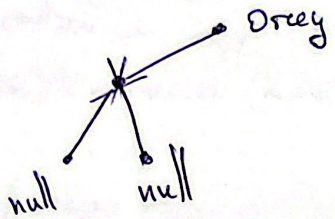
1.) Делаем проверку алгоритмом поиска на наличие листа, если его нет выполняем создание нового листа и вставляем туда элемент.

Удаление элемента $O(\log n) + O(1)$

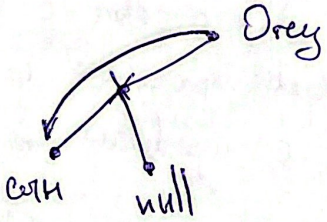
1.) Находим элемент с помощью поиска:

а.) Узел - лист без потомков → Просто удаляем узел

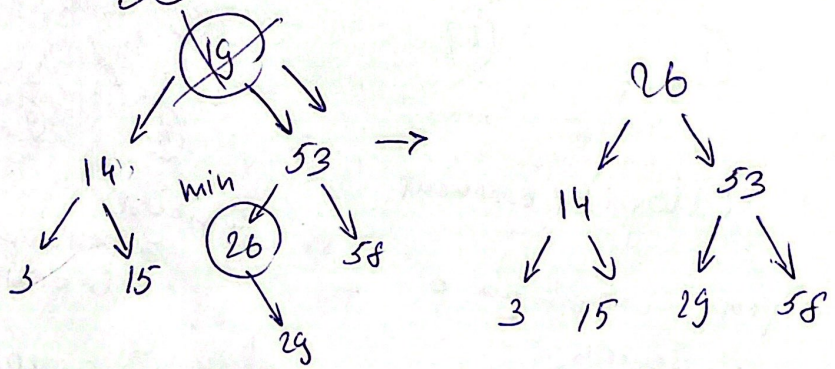
б



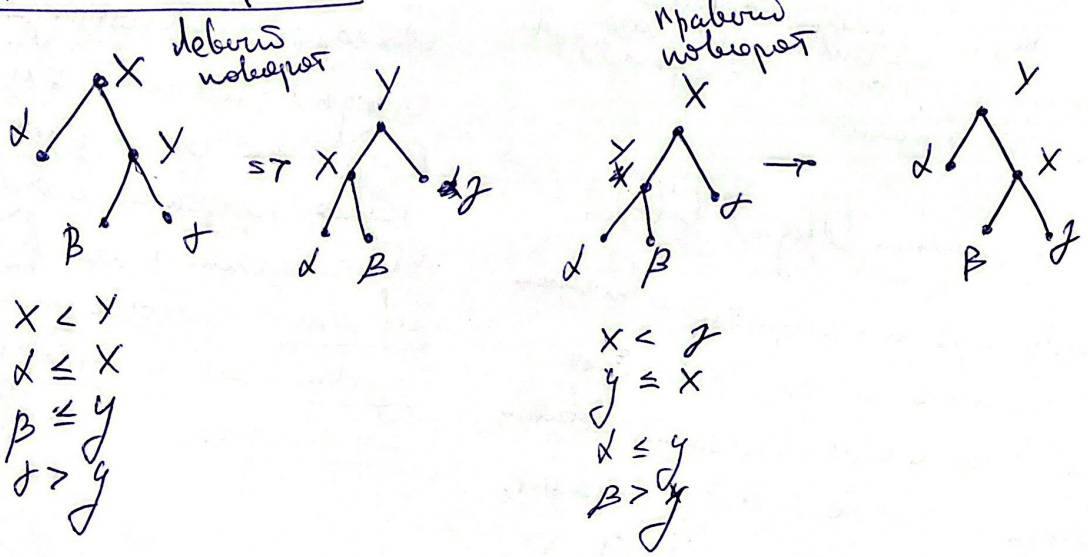
б.) У узла есть потомок → делаем ссылку из отца на сына



в.) У узла два потомка → Идем один раз направо, а далее ищем мин элемент в этом дереве. Заменяем наш удаляемый узел на этот мин. элемент.



Алгоритм поворота $O(1)$



$x < y$
$\alpha \le x$
$\beta \le y$
$\gamma > y$

$x < z$
$y \le x$
$\alpha \le y$
$\beta > y$