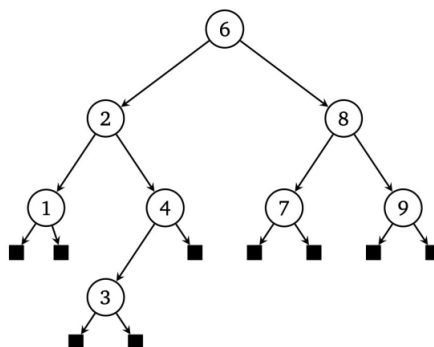


TODO:

1. Сравните конфигурации AVL-дерева, красно-черного дерева, 2-3(-4) дерева, которые возникают в результате последовательной вставки упорядоченной последовательности значений $A = \{1, 2, \dots, n\}$. При создании какого из деревьев поиска было выполнено наименьшее число преобразований?
2. Найдите две различные правильные красно-черные раскраски вершин этого дерева



3. 1) Верно ли, что любое поддерево красно-черного дерева также само является правильным красно-черным деревом?
2) Верно ли, что длина самого длинного пути от корня до листа в красно-черном дереве не более, чем в два раза превосходит длину самого короткого?
4. По набору ключей $A = \{10, 85, 15, 70, 20, 60, 30, 50\}$ построить 2-3-4 дерево

Task 1: (Сравнение конфигураций трех деревьев)

AVL-дерево: при вставке каждого следующего элемента каждый новый элемент вставляется как правый потомок предыдущего, после чего

происходят операции вращения для сохранения свойств дерева (одно или два), в противном случае дерево потеряет свойства. Для каждого нового элемента выполняется следующее: для каждого узла от корня до нового узла проверяем баланс, и при необходимости совершаем вращения. Так как последовательность упорядоченная, то дисбаланс возникает на каждом уровне, а значит каждый уровень требует вращения. Таким образом, каждый новый элемент требует $O(\log n)$ операций, а всего n элементов, то общее количество преобразований равно $O(n \log n)$

RB-дерево: при вставке каждого следующего элемента каждый новый элемент вставляется как в обычное BST дерево и окрашивается в красный цвет. При вставке упорядоченной последовательности дисбаланс возникает в двух случаях: 1. у красного узла красный родитель; 2. от корня до любого листа должно быть одинаковое число черных узлов. Практически во всех случаях дисбаланс устраняется перекраской узла без вращения. Тем не менее, вращение требуется в случае, если идут два красных узла подряд. Количество подобных исходов для n элементов упорядоченной последовательности можно оценить как $O(\log n)$.

2-3(-4) дерево: каждый новый элемент вставляется в листья, а значит дисбаланс возникает только при переполнении листа. Но переполнение узла влияет лишь на потомков, не влияя на остальную часть дерева. У этого дерева есть лишь две операции для восстановления баланса: разделение переполненного узла и подъем элемента на уровень выше. Переполнение и подъем элемента происходит раз в 3 добавления нового элемента в лист, тем не менее высота дерева наиболее близка к $O(\log n)$, а значит этому же значению пропорционально количество преобразований.

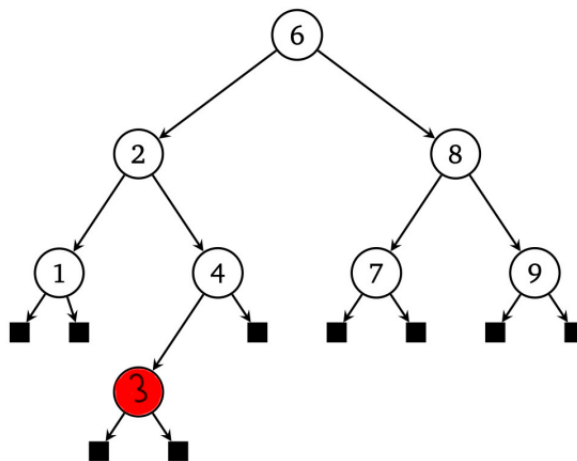
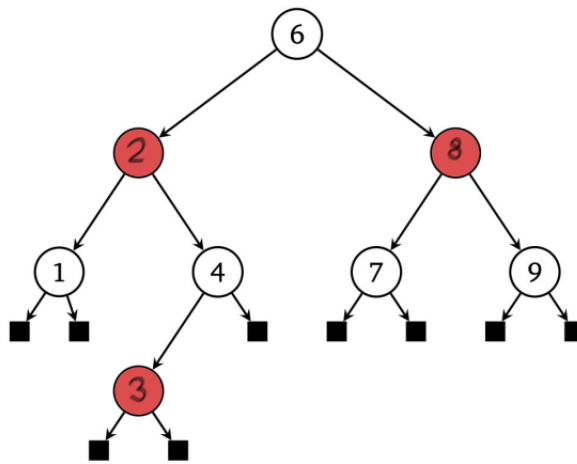
Таким образом, AVL требует наибольшее количество преобразований, а RB и 2-3(-4) дерево примерно равны, но при этом скрытая константа 2-3(-4) дерева меньше, чем у RB дерева, а значит 2-3(-4) дерево требует наименьшее число преобразований.

Task 2: (Различные раскраски)

У RB-дерева есть следующие ограничения:

1. Каждый узел красный или черный
2. Корень дерева всегда черный

3. Все листья дерева черные
4. У красного узла не может быть красных потомков
5. На пути от корня до любого из листьев должно быть одинаковое число черных узлов



Task 3.1: (RB поддерево)

Неверно, так как в поддереве корень может быть красным, а в RB дереве корень обязательно черный.

Task 3.2: (Длина пути в RB дереве)

Верно, это достигается в том случае, если длина самого короткого пути от корня до листа равна n , а длина самого длинного пути равна $2n$, на пути которого половина узлов - красные, что не нарушает балансировку RB дерева.

Task 4: (2-3-4 дерево по A)

Этапы:

1. Вставляем 10, он становится корнем 2-узлом

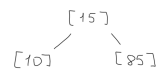
[10]

2. Вставляем 85 в тот же узел, тк как $85 > 10$. Узел становится 3-узлом

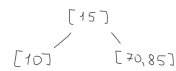
[10, 85]

3. Вставляем 15 в тот же узел, узел становится 4-узлом, после чего узел 15 поднимается вверх и образует новый корень, а 10 и 85 образуют два новых узла: левый и правый соответственно

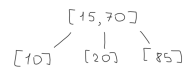
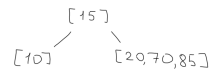
[10, 15, 85]



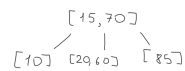
4. Вставляем 70 в правый узел, таким образом узел сейчас [70, 85]



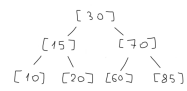
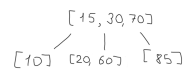
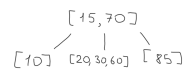
5. Вставляем 20 в правый узел, он становится [20, 70, 85], после 70 поднимается вверх, а 20 и 85 образуют два новых узла



6. Вставляем 60 в средний узел и он становится [20, 60]



7. Вставляем 30 в средний узел и он становится [20, 30, 60], 30 уходит наверх, 20 и 60 образуют два новых узла, а корень становится [20, 30, 60] и снова делится.



8. Вставляем 50 в правый под-узел от корня

