Отчет по курсовой работе по курсу “Алгоритмы и структуры данных (углубленный курс)”

Саитов Ирек, группа М4138

# Постановка задачи

В данной курсовой работе необходимо было:

* Реализовать Красно-черное дерево, с поддержкой основных операций: поиска, нахождения минимума и максимума, вставки значения по ключу, также удаления ключа и связанного с ним значения.
* Провести тестирование корректности и провести анализ зависимости высоты получаемого дерева от количества значений.
* В качестве результата предоставить исходный код и краткий отчет с подробностями реализации, результатами тестов и графиками зависимости.

# Подробности реализации

Красно-черное дерево было реализовано на языке программирования *Python,* в классе *RBTree*, а узел в классе *RBNode,* также был задействован вспомогательный класс *Color* для удобства кодирования дерева. Исходный код размещен на [GitHub](https://github.com/CaptainFest/RBTree)[.](https://github.com/vanyaland/ifmo-b-tree) Выполняется поддержка следующих операций:

* Поиск:
  + Метод *find(key)* производит поиск ключа в дереве, и если такой ключ найден, то возвращает узел, в котором хранится этот ключ.
* Вставка:
  + Метод *add\_key(key)* вызывает инициализацию нового узла с ключом key и передаёт этот узел в метод вставки узла *add\_node(node)*.
  + Метод *add\_node(node)* осуществляет вставку нового узла в дерево.
  + Метод \_*add\_fix(node)* модифицирует структуру дерева путём поворотов \_right\_rotate(node) или \_left\_rotate(node) и окрашиванием узлов, чтобы все свойства красно-черного дерева выполнялись после вставки.
* Maximum:
* Метод *max\_key(node)* находит узел с максимальным значением ключа в поддереве с корнем в узле node.
* Minimum:
* Метод *min\_key(node)* находит узел с минимальным значением ключа в поддереве с корнем в узле node.
* Удаление:
  + Метод *delete\_key(key)* вызывает функцию поиска узла *find(key)* и передаёт найденный узел, как параметр для функции удаления узла *\_delete\_node*(*find(key))*
  + Метод *\_delete\_fix(node)* восстанавливает структуру красно-черного дерева после удаления узла путём поворотов *\_turn\_left(node)* или *\_turn\_right(node)* и перекрашиванием узлов.
* Height:
  + Метод *tree\_black\_height()* возвращает число равное черное высоте дерева. Основано на таком свойстве красно-черного дерева, что черная высота этого дерева одинакова для любого простого пути от корня к листку.
  + Метод *tree\_height(node, l\_height, r\_height)* находит максимальную высоту дерева. Эта функция основана на рекурсивном обходе дерева.

Вспомогательные функции и методы:

* Родственники:
* Метод *\_brother(node)* возвращает брата узла node
* Метод *\_uncle(node)* возвращает дядю узла node.
* Метод *\_grandfather(node)* возвращает дедушку узла node.

Эти методы используются для удобства кодирования. В особенности позволяют избежать дублирования кода в функциях *\_add\_fix(node)* и *\_delete\_fix(node).*

* Повороты:
* Метод *\_turn\_left(node)* осуществляет поворот узла node налево, при этом происходит переподвешивание ребёнков узлов node и node --> rightChild.
* Метод *\_turn\_right(node)* осуществляет поворот узла node направо, при этом происходит переподвешивание ребёнков узлов node и node --> leftChild.

Эти методы используются для поддержания свойств красно-черного дерева.

# Подробности тестирования

Работа с программой осуществляется через консоль. В качестве тестов корректности работы структуры данных были реализованы юнит-тесты для основных функций. Также функция вставки тестируется на рандомном множестве неповторяющихся ключей. А при тесте удаления после каждого удаленного узла в консоли выводится текущий список узлов в дереве. В конце тестов функций, которые меняют структуру дерева (т.е. вставка и удаление), результат записывается в файл f.txt и выполняется проверка корректности свойств измененного дерева. Результат тестов функций minimum, maximum, поиск можно посмотреть в консоли.

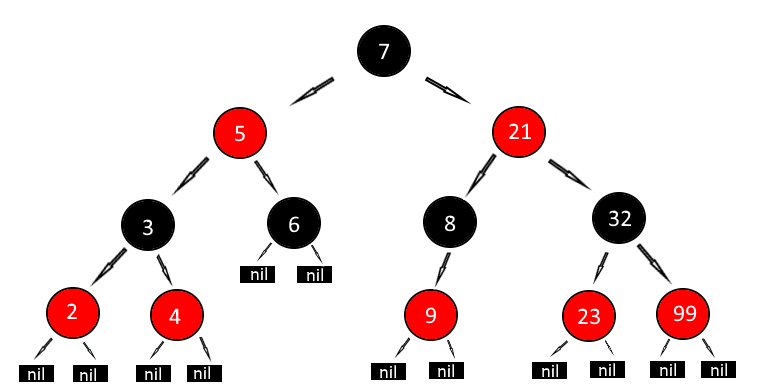
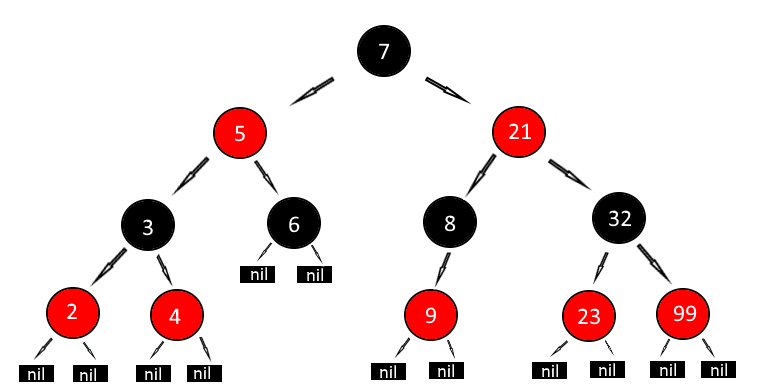
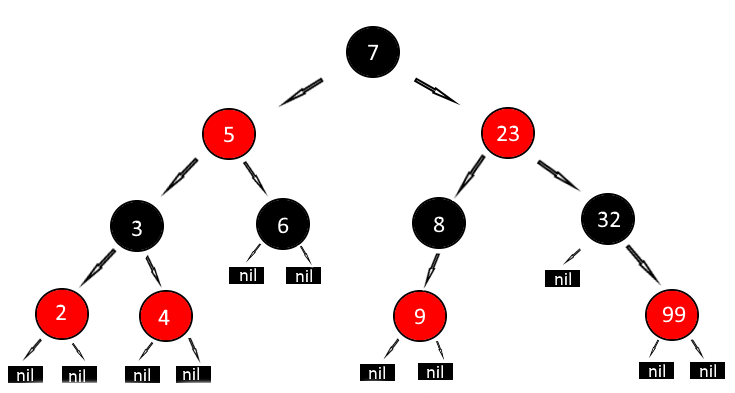


Рис. 1 Визуальное представление результата test\_add



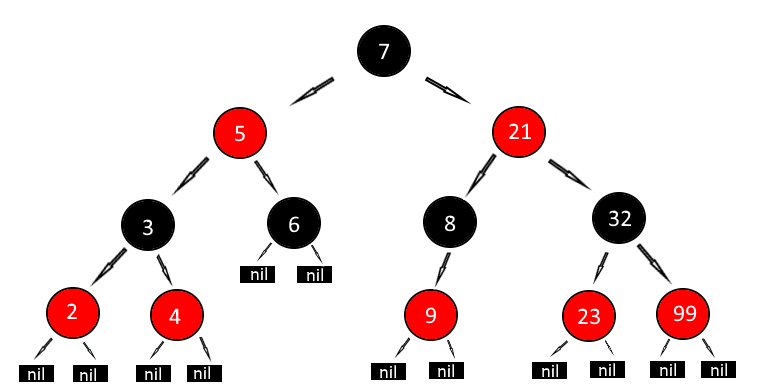
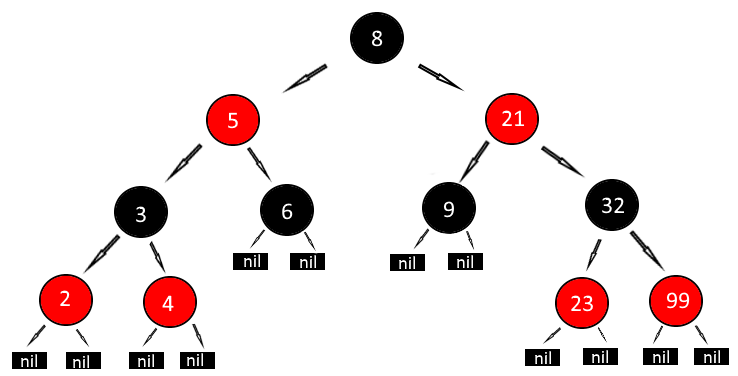
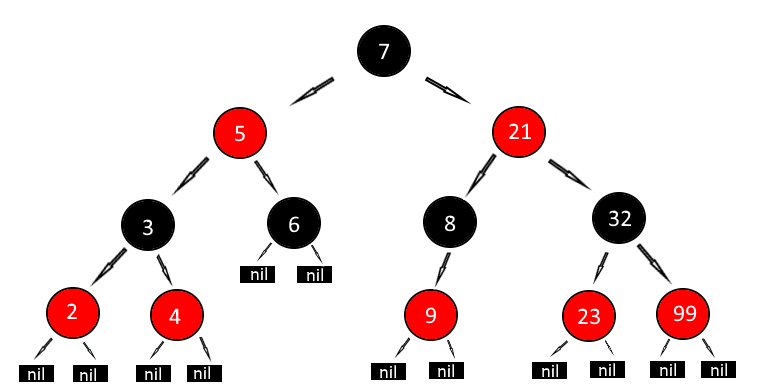
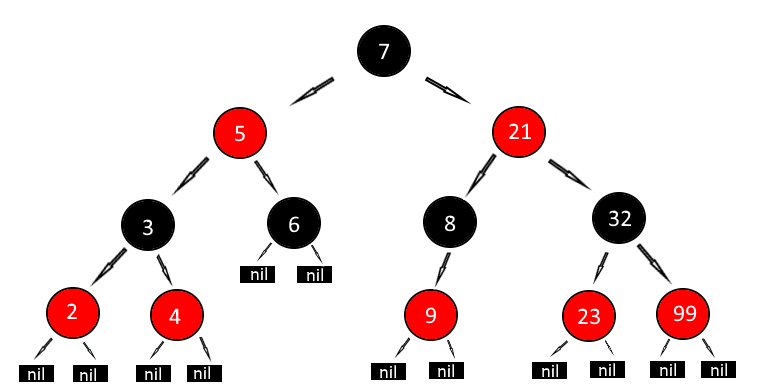
Рис. 2 Визуальное представление результата test\_delete удаления узла 21

Рис. 3 Визуальное представление результата test\_delete удаления узла 7



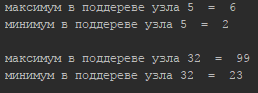


Рис. 4 Визуальное представление test\_min\_max для узлов 5 и 32

Рис. 5 Визуальное представление результата test\_find для ключей 24, 23 и 99

Для исследования зависимости высоты дерева от количества узлов в дереве были проведены 1000 опытов для каждого конкретного значения ключа в диапазоне от N = {0, 2000} с разным набором допустимых ключей. Затем взяты среднее арифметическое высоты и построены следующий график зависимости:

Исходя из результатов можно сделать вывод, что наблюдается логарифмическая зависимость высоты от количества ключей. Причем увеличение высоты дерева происходит скачкообразно в местах, которые являются степенью 2. Например, около N = 512 происходит рост вероятности построения более высокого дерева, это же можно увидеть у 1024.

Также можно вывести верхнюю границу для высоты красно-черного дерева:

где h – высота дерева, а N – кол-во узлов или ключей.

Сделанное предположение о получении более укомплектованного дерева (высота наименьшая) при динамическом расширении (например, для 40 узлов берутся значения ключей от 0 до 40; для 80 узлов – от 0 до 80) ключей неверно. Отсюда можно сделать вывод, что последовательность добавления узлов влияет на получаемую высоту красно-черного дерева, так как при каждом добавлении новый узел подвешивается к родителю листка, а уже затем происходит балансировка.