**Вопросы по третьему модулю.**

**х - значение(узел), к - ключ**

1. Сколько ребер содержит дерево из n вершин? Докажите.

n-1 доказывается по индукции)

1. Опишите обход двоичного дерева в глубину (pre-order, post-order, in-order).

(pre: узел, левое, правое поддер; post: левое, правое поддер, узел; in: левое, узел, правое поддер)

1. Опишите обход двоичного дерева в ширину. (используется очередь извлекается вершина, посещается, в очередь помещаются дочерние)
2. Опишите операцию поиска в двоичном дереве поиска. (рекурсивно: если к равен х - нашли, если меньше - ищи в левом подд, иначе - в правом)
3. Опишите операцию поиска вершины с минимальным (максимальным) ключом в двоичном дереве поиска. (переходить в левый (правый) дочерний узел, пока таковой существует)
4. Опишите наивный способ добавления элемента в двоичное дерево поиска. (если пусто заменить его на дерево с одним узлом с данным к, остановиться, иначе если к меньше х рекурсивно добавить в левое, иначе в правое)
5. Опишите наивный способ удаления элемента из двоичного дерева поиска. (если пусто остановиться, если к меньше х рекурсивно удалить из левого, больше - из правого, х==к рассмотреть 3 случая: нет дочерних - удаляем х обнуляем ссылку, нет одного дочернего - переносим дочерний в х удаляем узел дочернего, оба дочерние есть: минимальный к из правого подд в х удаляем мин)
6. Опишите операцию Split в декартовом дереве.(пусть к > корня, тогда если пусто то оба разреза = 0, иначе левое поддерево совпадет с левым поддеревом исходного, а правое Для нахождения правого поддерева, нужно разрезать правое поддерево исходного на по к и взять левое, а полученное правое будет отрезанным правым, симметрично для к < корня)
7. Опишите операцию Merge в декартовом дереве. ( у результирующего дерева корнем станет корень одного из данных деревьем с большим приоритетом. пусть у левого корень с большим приоритетом. Тогда левое поддерево результата совпадёт с левым поддеревом левого исходного. Справа же нужно подвесить объединение правого поддерева левого исходного и правого исходного.)
8. Опишите операцию вставки в декартово дерево без помощи слияния.(спускаемся как обычно останавливаемся на первом эл-те с меньшим приоритетом режем поддерево найденного эл-та полученные деревья пишем как сыновей вставляемого все это дерево ставим на место эл-та найденного в начале)
9. Опишите операцию удаления из декартова дерева без помощи разрезания.(спускаемся до удаляемого эл-та сливаем его левое и правое поддерево записываем результат вместо удаляемого элемента)
10. При каком условии следует выполнять малый левый поворот в АВЛ-дереве? (когда высота правого равна выс левого+2 и выс центра <=выс правого)
11. При каком условии следует выполнять большой левый поворот в АВЛ-дереве? (когда высота правого равна выс левого+1 и выс центров равна выс левого+2)
12. Опишите операцию добавления вершины в АВЛ-дерево.(проходим поиском убеждаемся что ключа в дереве нет, выполняем наивно вставку элемента, проверяем разницу высот поддеревьев каждой вершины, если она = 2 выполняем соотв вращение)
13. Опишите операцию удаления вершины из АВЛ-дерева.(ищем вершину для удаления, удаляем по алгоритму для дерева поиска, проверяем баланс см п14)
14. Постройте АВЛ-дерево по следующей последовательности команд: +1, +4, +7, + 2, +5,+3, -4, +8, + 9, -8, - 2. Команда +k означает добавление узла с ключом k, команда -k означает удаление узла с ключом k.(3(1,7(5,9))
15. Оценка высоты красно-черного дерева.(o(log n))
16. Операция вставки в красно-черное дерево.(идем наивным способ до нужного листа создаем вставляем эл-т делаем его красным, восстанавливаем св-ва кр-чер дерева, если отец эл-та черный, то все норм, если красный смотрим 2 случая: если дядя тоже красный то отца и дядю в черный деда в красный и так рекурсией до корня корень по-любому делаем черным; если дядя черный и правый и добавили эл-т справа то делаем левый поворот - превращаем его в левого потомка и делаем правый поворот для деда и его сыновей тогда дед и вставка становятся оба рядом и красные перекрашиваем их как в 1 случае)
17. Сплей-дерево. Операция Splay.(это обычное дерево поиска, в котором фигурирует операция сплей для операций find, add, remove, сплей от дерева и ключа делает ключ корнем дерева и далее все как в бинарном дп)
18. Поиск, вставка и удаление в сплей-дереве.(find: для узла вызовем сплей и он окажется в корне, add: запускаем сплей от дерева и х и подвешиваем два полученных дерева (если корень >= ключу) это левое поддерево исходного и корень и правое дерево исходного, remove: делаем сплей для х и сливаем его правое и левое поддеревья так, что запустим сплей от максимума левого поддерева этот макс - новый корень, у него есть левое подд но нет правого и правым станет правое поддерево исходного)
19. B-дерево. Определение. Ограничения на количество ключей и дочерних узлов.( в-дерево - это идеально сбалансированное дерево, все листья на одной глубине

* Каждый узел, кроме корня, содержит не менее t - 1 ключей, и каждый внутренний
* узел имеет по меньшей мере t дочерних узлов. Если дерево не является пустым,корень должен содержать как минимум один ключ.
* Каждый узел, кроме корня, содержит не более 2t - 1 ключей и не более чем 2t сыновей во внутренних узлах
* Корень содержит от 1 до 2t - 1 ключей, если дерево не пусто и от 2 до 2t детей при высоте большей 0.
* Каждый узел, кроме листьев, содержащий ключи k\_1, ..., k\_n, имеет n+1 сына. i-й сын содержит ключи из отрезка [k\_{i - 1}; k\_i],\: k\_0 = -\infty,\: k\_{n + 1}= \infty.
* Ключи в каждом узле упорядочены по неубыванию.
* Все листья находятся на одном уровне.)