Задание к лабораторной работе №2 (семестр 2). Двоичные деревья поиска

В данной лабораторной работе изучается новая структура данных, двоичные (бинарные) деревья поиска и сбалансированные деревья поиска: AVL и Splay. Аналогично предыдущим лабораторным работам, есть **два** способа ее выполнения и защиты, однако присутствуют изменения:

1 **Базовый уровень.** ***Решается 4 задачи: три по вариантам и одну выбираете произвольно***. Варианты в табличке внизу, номер вашего варианта соответствует вашему номеру в списке группы. **Посмотреть свой номер нужно, например, в журнале успеваемости по дисциплине**. Вариант в течении семестра менять нальзя!

Условия базового уровня.

- **Всего баллов.** За базовый уровень максимум за защиту и отчет **в сумме** можно получить 9 баллов, если сдаете в срок (1 месяц, до **20 апреля 2022** года включительно).
- **Замена.** ***Одну*** задачу можно заменить без потерь **внутри** уровня сложности, или на более сложную. На уровень ниже - нельзя заменить.
- **Сложность.** У каждой задачи подписано количество баллов, которые можно за нее получить, и по этим баллам задачи распределены на 4 уровня сложности:
  * 1 уровень, задачи по 1-1,5 баллов: с 1 по 6;
  * 2 уровень, задачи по 2-2,5 баллов: с 7 по 12;
  * 3 уровень, задачи по 3 балла: с 13 по 17;
  * Задача №18 не входит ни в один уровень и ни в один вариант.
- **Замена на задачу ниже уровнем**. Вы можете заменить одну задачу на уровень ниже, но тогда вы потеряете разницу в стоимости между ними.
- Замена второй задачи влечет за собой штраф в дополнительные 1.5 балла.
- **Отчет.** На защиту ***без отчета***, оформленного по правилам и загруженного в гитхаб, можно ***не приходить***!
- Правила оформления отчета:
  * Формат файла - pdf;
  * Титульный лист с названием учреждения и факультета, в котором вы учитесь, курса, названия работы, а также ваши ФИО, группа; ФИО преподавателя; город, дата выполнения;
  * Описание задания к задаче;
  * Описание вашего решения и исходный код;
  * Описание проведенных тестов;
  * Выводы по каждой задаче и в целом по проделанной работе.
- Хорошо оформленный отчет может добавить до 1 балла к защите, на усмотрение преподавателя. Т.е. можно получить 10 баллов в итоге.
- За плохо оформленный отчет может сниматься до 2 баллов, также по усмотрению преподавателя. Т.е. можно получить только 7 баллов даже с учетом правильности всех задач.
- **Дедлайн**. Если вы сдаете лабораторную работу *позже* срока (начиная с 20.04.2022), то суммарная базовая стоимость понижается на 2 балла, т.е. до 7 баллов. Эти 2 балла можно компенсировать решением дополнительных задач.

2 **Продвинутый уровень.** Решаются ваши задачи по вариантам, причем принципы, описанные выше для базового случая тоже учитываются. До максимальных 15 баллов вы можете набрать, решая другие задачи не из вашего варинта, выбирая произвольно с учетом стоимости задач. На ваше усмотрение.

Условие на дедлайн так же действует для продвинутого уровня (-2 балла).

P.s. Решать все задачи вообще не обязательно. Но если хочется, то каждая решенная **до дедлайна** задача 3 уровня сложности (а также 18), которая превышает лимит в максимальные 15 баллов, при условии, что на 15 баллов уже сданы задачи, – может быть дополнительно учтена по количеству баллов в конце семестра в счет, например, экзамена.

Вариант	Номера задач	Вариант	Номера задач
1	1,12,17	16	1,11,16
2	2,7,13	17	2,8,17
3	3,8,14	18	3,9,13
4	4,9,15	19	4,9,14
5	5,10,16	20	6,8,15
6	6,11,15	21	1,12,16
7	1,12,14	22	2,7,16
8	3,10,13	23	3,11,15
9	4,9,17	24	4,10,14
10	5,8,11	25	5,11,17
11	2,7,16	26	6,9,13
12	6,11,12	27	2,8,11
13	3,12,13	28	3,7,16
14	4,7,14	29	4,12,15
15	5,10,15	30	5,9,14

Содержание

1 Задача. Обход двоичного дерева [5 s, 512 Mb, 1 балл]

В этой задаче вы реализуете три основных способа обхода двоичного дерева «в глубину»: центрированный (in-order), прямой (pre-order) и обратный (post-order). Очень полезно попрактиковаться в их реализации, чтобы лучше понять бинарные деревья поиска.

Вам дано корневое двоичное дерево. Выведите центрированный (in-order), прямой (pre-order) и обратный (post-order) обходы в глубину.

- **Формат ввода: стандартный ввод или input.txt.** В первой строке входного файла содержится количество узлов n. Узлы дерева пронумерованы от 0 до $n-1$. Узел 0 является корнем.

  Следующие n строк содержат информацию об узлах $0, 1, ..., n-1$ по порядку. Каждая из этих строк содержит три целых числа K_i, L_i и R_i. K_i – ключ i-го узла, L_i - индекс левого ребенка i-го узла, а R_i - индекс правого ребенка i-го узла. Если у i-го узла нет левого или правого ребенка (или обоих), соответствующие числа L_i или R_i (или оба) будут равны -1.

- **Ограничения на входные данные.** $1 \le n \le 10^5, 0 \le K_i \le 10^9, -1 \le L_i, R_i \le n-1$. Гарантируется, что данное дерево является двоичным деревом. В частности, если $L_i \ne -1$ и $R_i \ne -1$, то $L_i \ne R_i$. Кроме того, узел не может быть ребенком двух разных узлов. Кроме того, каждый узел является потомком корневого узла.

- **Формат вывода / выходного файла (output.txt).** Выведите три строки. Первая строка должна содержать ключи узлов при центрированном обходе дерева (in-order). Вторая строка должна содержать ключи узлов при прямом обходе дерева (pre-order). Третья строка должна содержать ключи узлов при обратном обходе дерева (post-order).

- Ограничение по времени. 5 сек.

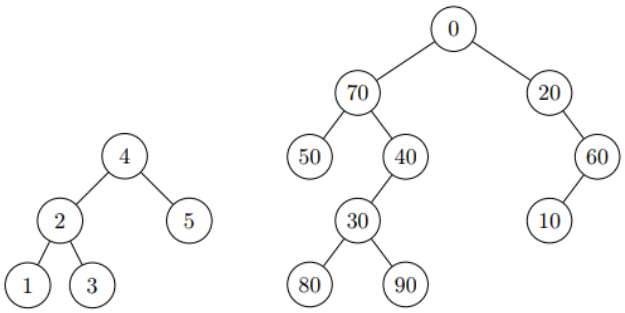
- Ограничение по памяти. 512 мб.

- Примеры:

input	output.txt
5 4 1 2 2 3 4 5 -1 -1 1 -1 -1 3 -1 -1	1 2 3 4 5 4 2 1 3 5 1 3 2 5 4

input	output.txt
10 0 7 2 10 -1 -1 20 -1 6 30 8 9 40 3 -1 50 -1 -1 60 1 -1 70 5 4 80 -1 -1 90 -1 -1	50 70 80 30 90 40 0 20 10 60 0 70 50 40 30 80 90 20 60 10 50 80 90 30 40 70 10 60 20 0

- Иллюстрации к обоим примерам:



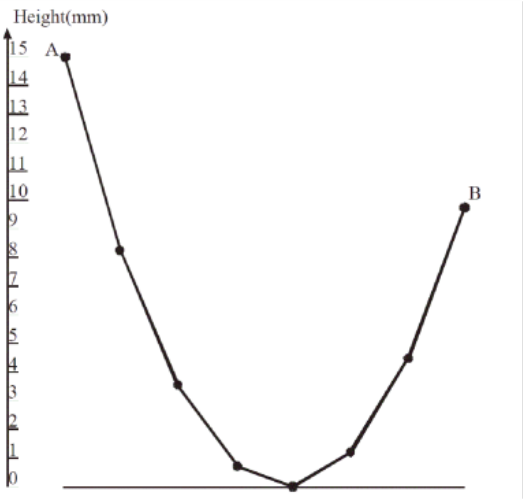
- Что делать. Реализовать алгоритмы обхода из лекции. Обратите внимание, что в этой задаче дерево может быть очень глубоким, поэтому будьте осторожны, избегайте проблем с переполнением стека, если используете рекурсию, и обязательно протестируйте свое решение на дереве максимально возможной высоты.

2 Задача. Гирлянда [2 s, 256 Mb, 1 балл]

Гирлянда состоит из n лампочек на общем проводе. Один её конец закреплён на заданной высоте A мм ($h_1 = A$). Благодаря силе тяжести гирлянда прогибается: высота каждой неконцевой лампы на 1 мм меньше, чем средняя высота ближайших соседей ($h_i = \frac{h_{i-1} + h_{i+1}}{2} - 1$ для $1 < i < N$).

Требуется найти минимальное значение высоты второго конца B ($B = h_n$), такое что для любого $\epsilon > 0$ при высоте второго конца $B + \epsilon$ для всех лампочек выполняется условие $h_i > 0$. Обратите внимание на то, что при данном значении высоты либо ровно одна, либо две соседних лампочки будут иметь нулевую высоту.

Подсказка: для решения этой задачи можно использовать двоичный поиск.



- **Формат ввода / входного файла (input.txt).** В первой строке входного файла содержится два числа n и A.
- **Ограничения на входные данные.** $3 \le n \le 1000$, n – целое, $10 \le A \le 1000$, A – вещественное и дано не более чем с тремя знаками после десятичной точки.
- **Формат вывода / выходного файла (output.txt).** Выведите одно вещественное число B – минимальную высоту второго конца. Ваш ответ будет засчитан, если он будет отличаться от правильного не более, чем на 10^{-6}.
- **Ограничение по времени. 2 сек.**
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt
8 15	9.75
692 532.81	446113.34434782615

- Проверить можно по ссылке, OpenEdu, курс "Алгоритмы программирования и структуры данных 6 неделя, 2 задача.

3 Задача. Простейшее BST [2 s, 256 Mb, 1 балл]

В этой задаче вам нужно написать простейшее BST по явному ключу и отвечать им на запросы:

- «+ x» – добавить в дерево x (если x уже есть, ничего не делать).
- «> x» – вернуть минимальный элемент больше x или 0, если таких нет.

- **Формат ввода / входного файла (input.txt).** В каждой строке содержится один запрос. Все x - целые числа, количество запросов N не указано в начале, не более 300 000. Гарантируется, что все x выбраны равномерным распределением.
- Случайные данные! Не нужно ничего специально балансировать.
- **Ограничения на входные данные.** $1 \leq x \leq 10^9$, $1 \leq N \leq 300000$
- **Формат вывода / выходного файла (output.txt).** Для каждого запроса вида «> x» выведите в отдельной строке ответ.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
+ 1	3
+ 3	3
+ 3	0
> 1	2
> 2	
> 3	
+ 2	
> 1	

4 Задача. Простейший неявный ключ [2 s, 256 Mb, 1 балл]

В этой задаче вам нужно написать BST по **неявному** ключу и отвечать им на запросы:

- «+ x» – добавить в дерево x (если x уже есть, ничего не делать).
- «? k» – вернуть k-й по возрастанию элемент.

- **Формат ввода / входного файла (input.txt).** В каждой строке содержится один запрос. Все x - целые числа, количество запросов N не указано в начале, не более 300 000. Гарантируется, что все x выбраны равномерным распределением.
- Случайные данные! Не нужно ничего специально балансировать.
- **Ограничения на входные данные.** $1 \le x \le 10^9$, $1 \le N \le 300000$, в запросах «? k», число k от 1 до количества элементов в дереве.
- **Формат вывода / выходного файла (output.txt).** Для каждого запроса вида «? k» выведите в отдельной строке ответ.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
+ 1	1
+ 4	3
+ 3	4
+ 3	3
? 1	
? 2	
? 3	
+ 2	
? 3	

5 Задача. Простое двоичное дерево поиска [2 s, 512 Mb, 1 балл]

Реализуйте простое двоичное дерево поиска.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание операций с деревом, их количество N не превышает 100. В каждой строке находится одна из следующих операций:
  - insert x – добавить в дерево ключ x. Если ключ x есть в дереве, то ничего делать не надо;
  - delete x – удалить из дерева ключ x. Если ключа x в дереве нет, то ничего делать не надо;
  - exists x – если ключ x есть в дереве выведите «true», если нет – «false»;
  - next x – выведите минимальный элемент в дереве, строго больший x, или «none», если такого нет;
  - prev x – выведите максимальный элемент в дереве, строго меньший x, или «none», если такого нет.

  В дерево помещаются и извлекаются только целые числа, не превышающие по модулю 10^9.
- **Ограничения на входные данные.** $0 \le N \le 100$, $|x_i| \le 10^9$.
- **Формат вывода / выходного файла (output.txt).** Выведите последовательно результат выполнения всех операций exists, next, prev. Следуйте формату выходного файла из примера.
- **Ограничение по времени. 2 сек.**
- Ограничение по памяти. 512 мб.
- Пример:

input.txt	output.txt
insert 2	true
insert 5	false
insert 3	5
exists 2	3
exists 4	none
next 4	3
prev 4	
delete 5	
next 4	
prev 4	

6 Задача. Опознание двоичного дерева поиска [10 s, 512 Mb, 1.5 балла]

В этой задаче вы собираетесь проверить, правильно ли реализована структура данных бинарного дерева поиска. Другими словами, вы хотите убедиться, что вы можете находить целые числа в этом двоичном дереве, используя бинарный поиск по дереву, и вы всегда получите правильный результат: если целое число есть в дереве, вы его найдете, иначе – нет.

Вам дано двоичное дерево с ключами - целыми числами. Вам нужно проверить, является ли это правильным двоичным деревом поиска. Для каждой вершины дерева V выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева больше ключа вершины V.

Другими словами, узлы с меньшими ключами находятся слева, а узлы с большими ключами – справа. Вам необходимо проверить, удовлетворяет ли данная структура двоичного дерева этому условию. Вам гарантируется, что входные данные содержат допустимое двоичное дерево. То есть это дерево, и каждый узел имеет не более двух ребенков.

- **Формат ввода / входного файла (input.txt).** В первой строке входного файла содержится количество узлов n. Узлы дерева пронумерованы от 0 до $n-1$. Узел 0 является корнем.

  Следующие n строк содержат информацию об узлах $0, 1, ..., n-1$ по порядку. Каждая из этих строк содержит три целых числа K_i, L_i и R_i. K_i – ключ i-го узла, L_i - индекс левого ребенка i-го узла, а R_i - индекс правого ребенка i-го узла. Если у i-го узла нет левого или правого ребенка (или обоих), соответствующие числа L_i или R_i (или оба) будут равны -1.

- **Ограничения на входные данные.** $0 \le n \le 10^5$, $-2^{31} \le K_i \le 2^{31}-1$, $-1 \le L_i, R_i \le n-1$. Гарантируется, что данное дерево является двоичным деревом. В частности, если $L_i \ne -1$ и $R_i \ne -1$, то $L_i \ne R_i$. Кроме того, узел не может быть ребенком двух разных узлов. Кроме того, каждый узел является потомком корневого узла.

  Все ключи во входных данных различны.

- **Формат вывода / выходного файла (output.txt).** Если заданное двоичное дерево является правильным двоичным деревом поиска, выведите одно слово «CORRECT» (без кавычек). В противном случае выведите одно слово «INCORRECT» (без кавычек).
- Ограничение по времени. 10 сек.
- Ограничение по памяти. 512 мб.
- Примеры:

input.txt	output.txt
3 2 1 2 1 -1 -1 3 -1 -1	CORRECT

input.txt	output.txt
3 1 1 2 2 -1 -1 3 -1 -1	INCORRECT

input.txt	output.txt
0	CORRECT

input.txt	output.txt
5 1 -1 1 2 -1 2 3 -1 3 4 -1 4 5 -1 -1	CORRECT

input.txt	output.txt
7 4 1 2 2 3 4 6 5 6 1 -1 -1 3 -1 -1 5 -1 -1 7 -1 -1	CORRECT

input.txt	output.txt
4 4 1 -1 2 2 3 1 -1 -1 5 -1 -1	INCORRECT

- Примечание. Пустое дерево считается правильным двоичным деревом поиска. Дерево не обязательно должно быть сбалансировано.

7 Задача. Опознание двоичного дерева поиска (усложненная версия) [10 s, 512 Mb, 2.5 балла]

Эта задача отличается от предыдущей тем, что двоичное дерева поиска может содержать равные ключи.

Вам дано двоичное дерево с ключами - целыми числами, которые могут повторяться. Вам нужно проверить, является ли это правильным двоичным деревом поиска. Теперь, для каждой вершины дерева V выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева **больше или равны** ключу вершины V.

Другими словами, узлы с меньшими ключами находятся слева, а узлы с большими ключами – справа, дубликаты всегда справа. Вам необходимо проверить, удовлетворяет ли данная структура двоичного дерева этому условию.

- **Формат ввода / входного файла (input.txt).** В первой строке входного файла содержится количество узлов n. Узлы дерева пронумерованы от 0 до $n-1$. Узел 0 является корнем.

  Следующие n строк содержат информацию об узлах $0, 1, ..., n-1$ по порядку. Каждая из этих строк содержит три целых числа K_i, L_i и R_i. K_i – ключ i-го узла, L_i - индекс левого ребенка i-го узла, а R_i - индекс правого ребенка i-го узла. Если у i-го узла нет левого или правого ребенка (или обоих), соответствующие числа L_i или R_i (или оба) будут равны -1.

- **Ограничения на входные данные.** $0 \le n \le 10^5$, $-2^{31} \le K_i \le 2^{31}-1$, $-1 \le L_i, R_i \le n-1$. Гарантируется, что данное дерево является двоичным деревом. В частности, если $L_i \ne -1$ и $R_i \ne -1$, то $L_i \ne R_i$. Кроме того, узел не может быть ребенком двух разных узлов. Кроме того, каждый узел является потомком корневого узла. Обратите внимание, что минимальное и максимальное возможные значения 32-битного целочисленного типа могут быть ключами в дереве.

- **Формат вывода / выходного файла (output.txt).** Если заданное двоичное дерево является правильным двоичным деревом поиска, выведите одно слово «CORRECT» (без кавычек). В противном случае выведите одно слово «INCORRECT» (без кавычек).

- Ограничение по времени. 10 сек.

- Ограничение по памяти. 512 мб.

- Примеры:

input.txt	output.txt
3 2 1 2 1 -1 -1 3 -1 -1	CORRECT

input.txt	output.txt
3 1 1 2 2 -1 -1 3 -1 -1	INCORRECT

input.txt	output.txt
3 2 1 2 1 -1 -1 2 -1 -1	CORRECT

input.txt	output.txt
3 2 1 2 2 -1 -1 3 -1 -1	INCORRECT

input.txt	output.txt
5 1 -1 1 2 -1 2 3 -1 3 4 -1 4 5 -1 -1	CORRECT

input.txt	output.txt
7 4 1 2 2 3 4 6 5 6 1 -1 -1 3 -1 -1 5 -1 -1 7 -1 -1	CORRECT

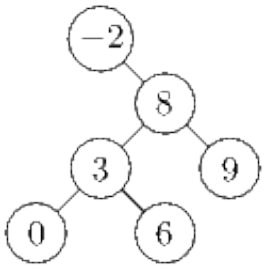
input.txt	output.txt
1 2147483647 -1 -1	CORRECT

- Примечание. Пустое дерево считается правильным двоичным деревом поиска. Дерево не обязательно должно быть сбалансировано. Попробуйте адаптировать алгоритм из предыдущей задачи к случаю, когда допускаются повторяющиеся ключи, и остерегайтесь целочисленного переполнения!

8 Задача. Высота дерева возвращается [2 s, 256 Mb, 2 балла]

Высотой дерева называется максимальное число вершин дерева в цепочке, начинающейся в корне дерева, заканчивающейся в одном из его листьев, и не содержащей никакую вершину дважды.

Так, высота дерева, состоящего из единственной вершины, равна единице. Высота пустого дерева равна нулю. Высота дерева, изображенного на рисунке, равна четырем.



Дано **двоичное дерево поиска**. В вершинах этого дерева записаны ключи – целые числа, по модулю не превышающие 10^9. Для каждой вершины дерева V выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева больше ключа вершины V.

Найдите высоту данного дерева.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание двоичного дерева. В первой строке файла находится число N – число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла $(1 \le i \le N)$ находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами – ключа K_i в i-ой вершине, номера левого L_i ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого R_i ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет).
- **Ограничения на входные данные.** $0 \le N \le 2 \cdot 10^5$, $|K_i| \le 10^9$. Все ключи различны. Гарантируется, что данное дерево является деревом поиска.
- **Формат вывода / выходного файла (output.txt).** Выведите одно целое число – высоту дерева.
- **Ограничение по времени. 2 сек.**
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
6 -2 0 2 8 4 3 9 0 0 3 6 5 6 0 0 0 0 0	4

- Во входном файле задано то же дерево, что и изображено на рисунке.
- Проверить можно по ссылке, OpenEdu, курс "Алгоритмы программирования и структуры данных 6 неделя, 3 задача.

9 Задача. Удаление поддеревьев [2 s, 256 Mb, 2 балла]

Дано некоторое двоичное дерево поиска. Также даны запросы на удаление из него вершин, имеющих заданные ключи, причем вершины удаляются целиком вместе со своими поддеревьями.

После каждого запроса на удаление выведите число оставшихся вершин в дереве.

В вершинах данного дерева записаны ключи – целые числа, по модулю не превышающие 10^9. Гарантируется, что данное дерево является двоичным деревом поиска, в частности, для каждой вершины дерева V выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева больше ключа вершины V.

Высота дерева не превосходит 25, таким образом, можно считать, что оно сбалансировано.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание двоичного дерева и описание запросов на удаление.

  В первой строке файла находится число N – число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла ($1 \le i \le N$) находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами – ключа K_i в i-ой вершине, номера левого L_i ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого R_i ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет).

  Все ключи различны. Гарантируется, что данное дерево является деревом поиска.

  В следующей строке находится число M – число запросов на удаление. В следующей строке находятся M чисел, разделенных пробелами – ключи, вершины с которыми (вместе с их поддеревьями) необходимо удалить. Все эти числа не превосходят 10^9 по абсолютному значению. Вершина с таким ключом не обязана существовать в дереве – в этом случае дерево изменять не требуется. Гарантируется, что корень дерева никогда не будет удален.

- **Ограничения на входные данные.** $1 \le N \le 2 \cdot 10^5$, $|K_i| \le 10^9$, $1 \le M \le 2 \cdot 10^5$
- **Формат вывода / выходного файла (output.txt).** Выведите M строк. На i-ой строке требуется вывести число вершин, оставшихся в дереве после выполнения i-го запроса на удаление.
- **Ограничение по времени. 2 сек.**
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
6 -2 0 2 8 4 3 9 0 0 3 6 5 6 0 0 0 0 0 4 6 9 7 8	5 4 4 1

- Проверить можно по ссылке, OpenEdu, курс "Алгоритмы программирования и структуры данных 6 неделя, 4 задача.

10 Задача. Проверка корректности [2 s, 256 Mb, 2 балла]

Свойство двоичного дерева поиска можно сформулировать следующим образом: для каждой вершины дерева выполняется следующее условие:

- все ключи вершин из левого поддерева меньше ключа вершины V;
- все ключи вершин из правого поддерева больше ключа вершины V.

Дано двоичное дерево. Проверьте, выполняется ли для него свойство двоичного дерева поиска.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание двоичного дерева.

  В первой строке файла находится число N – число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла ($1 \le i \le N$) находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами – ключа K_i в i-ой вершине, номера левого L_i ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого R_i ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет).
- **Ограничения на входные данные.** $0 \le N \le 2 \cdot 10^5$, $|K_i| \le 10^9$.
- На 60% от при $0 \le N \le 2000$.
- **Формат вывода / выходного файла (output.txt).** Выведите «YES», если данное во входном файле дерево является двоичным деревом поиска, и «NO», если не является.
- **Ограничение по времени. 2 сек.**
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt	output.txt
6 -2 0 2 8 4 3 9 0 0 3 6 5 6 0 0 0 0 0	YES

input.txt	output.txt
0	YES

input.txt	output.txt
3 5 2 3 6 0 0 4 0 0	NO

- Проверить можно по ссылке, OpenEdu, курс "Алгоритмы программирования и структуры данных 6 неделя, наблюдаемая задача.

11 Задача. Сбалансированное двоичное дерево поиска [2 s, 512 Mb, 2 балла]

Реализуйте сбалансированное двоичное дерево поиска.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание операций с деревом, их количество N не превышает 10^5. В каждой строке находится одна из следующих операций:
  - insert x – добавить в дерево ключ x. Если ключ x есть в дереве, то ничего делать не надо;
  - delete x – удалить из дерева ключ x. Если ключа x в дереве нет, то ничего делать не надо;
  - exists x – если ключ x есть в дереве выведите «true», если нет – «false»;
  - next x – выведите минимальный элемент в дереве, строго больший x, или «none», если такого нет;
  - prev x – выведите максимальный элемент в дереве, строго меньший x, или «none», если такого нет.

  В дерево помещаются и извлекаются только целые числа, не превышающие по модулю 10^9.
- **Ограничения на входные данные.** $0 \le N \le 10^5$, $|x_i| \le 10^9$.
- **Формат вывода / выходного файла (output.txt).** Выведите последовательно результат выполнения всех операций exists, next, prev. Следуйте формату выходного файла из примера.
- **Ограничение по времени. 2 сек.**
- Ограничение по памяти. 512 мб.
- Пример:

input.txt	output.txt
insert 2	true
insert 5	false
insert 3	5
exists 2	3
exists 4	none
next 4	3
prev 4	
delete 5	
next 4	
prev 4	

12 Задача. Проверка сбалансированности [2 s, 256 Mb, 2 балла]

АВЛ-дерево является сбалансированным в следующем смысле: для любой вершины высота ее левого поддерева отличается от высоты ее правого поддерева не больше, чем на единицу.

Введем понятие баланса вершины: для вершины дерева V ее баланс $B(V)$ равен разности высоты правого поддерева и высоты левого поддерева. Таким образом, свойство АВЛ-дерева, приведенное выше, можно сформулировать следующим образом: для любой ее вершины V выполняется следующее неравенство:

$$-1 \le B(V) \le 1$$

Обратите внимание, что, по историческим причинам, определение баланса в этой и последующих задачах этой недели «зеркально отражено» по сравнению с определением баланса в лекциях! Надеемся, что этот факт не доставит Вам неудобств. В литературе по алгоритмам – как российской, так и мировой – ситуация, как правило, примерно та же.

Дано двоичное дерево поиска. Для каждой его вершины требуется определить ее баланс.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание двоичного дерева.

  В первой строке файла находится число N – число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла $(1 \le i \le N)$ находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами – ключа K_i в i-ой вершине, номера левого L_i ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого R_i ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет). Все ключи различны. Гарантируется, что данное дерево является деревом поиска.

- **Ограничения на входные данные.** $0 \le N \le 2 \cdot 10^5$, $|K_i| \le 10^9$.

- **Формат вывода / выходного файла (output.txt).** Для i-ой вершины в i-ой строке выведите одно число – баланс данной вершины.

- **Ограничение по времени. 2 сек.**

- Ограничение по памяти. 256 мб.

- Пример:

input.txt	output.txt
6	3
-2 0 2	-1
8 4 3	0
9 0 0	0
3 6 5	0
6 0 0	0
0 0 0	

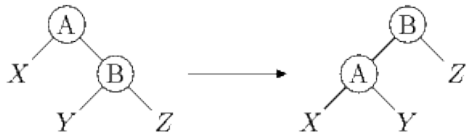
- Проверить можно по ссылке, OpenEdu, курс "Алгоритмы программирования и структуры данных 7 неделя, 1 задача.

13 Задача. Делаю я левый поворот... [2 s, 256 Mb, 3 балла]

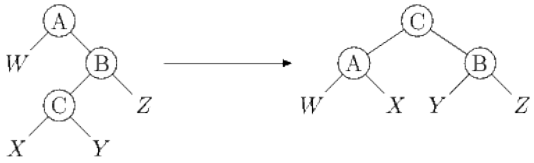
Для балансировки АВЛ-дерева при операциях вставки и удаления производятся *левые* и *правые* повороты. Левый поворот в вершине производится, когда баланс этой вершины больше 1, аналогично, правый поворот производится при балансе, меньшем -1.

Существует два разных левых (как, разумеется, и правых) поворота: *большой* и *малый* левый поворот.

Малый левый поворот осуществляется следующим образом:



Заметим, что если до выполнения малого левого поворота был нарушен баланс только корня дерева, то после его выполнения все вершины становятся сбалансированными, за исключением случая, когда у правого ребенка корня баланс до поворота равен -1. В этом случае вместо малого левого поворота выполняется большой левый поворот, который осуществляется так:



Дано дерево, в котором баланс корня равен 2. Сделайте левый поворот.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание двоичного дерева.

  В первой строке файла находится число N – число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла $(1 \le i \le N)$ находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами – ключа K_i в i-ой вершине, номера левого L_i ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого R_i ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет). Все ключи различны. Гарантируется, что данное дерево является деревом поиска. Баланс корня дерева (вершины с номером 1) равен 2, баланс всех остальных вершин находится в пределах от -1 до 1.

- **Ограничения на входные данные.** $3 \le N \le 2 \cdot 10^5$, $|K_i| \le 10^9$.

- **Формат вывода / выходного файла (output.txt).** Выведите в том же формате дерево после осуществления левого поворота. Нумерация вершин может быть произвольной при условии соблюдения формата. Так, номер вершины должен быть меньше номера ее детей.

- **Ограничение по времени. 2 сек.**

- Ограничение по памяти. 256 мб.

- Пример:

input.txt	output.txt
7	7
-2 7 2	3 2 3
8 4 3	-2 4 5
9 0 0	8 6 7
3 6 5	-7 0 0
6 0 0	0 0 0
0 0 0	6 0 0
-7 0 0	9 0 0

- Проверить можно по ссылке, OpenEdu, курс "Алгоритмы программирования и структуры данных 7 неделя, 2 задача.

14 Задача. Вставка в АВЛ-дерево [2 s, 256 Mb, 3 балла]

Вставка в АВЛ-дерево вершины V с ключом X при условии, что такой вершины в этом дереве нет, осуществляется следующим образом:

- находится вершина W, ребенком которой должна стать вершина V;
- вершина V делается ребенком вершины W;
- производится подъем от вершины W к корню, при этом, если какая-то из вершин несбалансирована, производится, в зависимости от значения баланса, левый или правый поворот.

Первый этап нуждается в пояснении. Спуск до будущего родителя вершины V осуществляется, начиная от корня, следующим образом:

- Пусть ключ текущей вершины равен Y.
- Если $X < Y$ и у текущей вершины есть левый ребенок, переходим к левому ребенку.
- Если $X < Y$ и у текущей вершины нет левого ребенка, то останавливаемся, текущая вершина будет родителем новой вершины.
- Если $X > Y$ и у текущей вершины есть правый ребенок, переходим к правому ребенку.
- Если $X > Y$ и у текущей вершины нет правого ребенка, то останавливаемся, текущая вершина будет родителем новой вершины.

Отдельно рассматривается следующий крайний случай – если до вставки дерево было пустым, то вставка новой вершины осуществляется проще: новая вершина становится корнем дерева.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание двоичного дерева, а также ключа вершины, которую требуется вставить в дерево.

  В первой строке файла находится число N – число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла ($1 \le i \le N$) находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами – ключа K_i в i-ой вершине, номера левого L_i ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого R_i ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет).

  Все ключи различны. Гарантируется, что данное дерево является корректным АВЛ-деревом.

  В последней строке содержится число X – ключ вершины, которую требуется вставить в дерево. Гарантируется, что такой вершины в дереве нет.

- **Ограничения на входные данные.** $0 \le N \le 2 \cdot 10^5$, $|K_i| \le 10^9$, $|X| \le 10^9$.
- **Формат вывода / выходного файла (output.txt).** Выведите в том же формате дерево после осуществления операции вставки. Нумерация вершин может быть произвольной при условии соблюдения формата.
- **Ограничение по времени. 2 сек.**
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
2 3 0 2 4 0 0 5	3 4 2 3 3 0 0 5 0 0

- Проверить можно по ссылке, OpenEdu, курс "Алгоритмы программирования и структуры данных 7 неделя, 3 задача.

15 Задача. Удаление из АВЛ-дерева [2 s, 256 Mb, 3 балла]

Удаление из АВЛ-дерева вершины с ключом X, при условии ее наличия, осуществляется следующим образом:

- путем спуска от корня и проверки ключей находится V – удаляемая вершина;
- если вершина V – лист (то есть, у нее нет детей):
  - удаляем вершину;
  - поднимаемся к корню, начиная с бывшего родителя вершины V, при этом если встречается несбалансированная вершина, то производим поворот.
- если у вершины V не существует левого ребенка:
  - следовательно, баланс вершины равен единице и ее правый ребенок – лист;
  - заменяем вершину V ее правым ребенком;
  - поднимаемся к корню, производя, где необходимо, балансировку.
- иначе:
  - находим R – самую правую вершину в левом поддереве;
  - переносим ключ вершины R в вершину V;
  - удаляем вершину R (у нее нет правого ребенка, поэтому она либо лист, либо имеет левого ребенка, являющегося листом);
  - поднимаемся к корню, начиная с бывшего родителя вершины R, производя балансировку.

Исключением является случай, когда производится удаление из дерева, состоящего из одной вершины - корня. Результатом удаления в этом случае будет пустое дерево.

Указанный алгоритм не является единственно возможным, но мы просим Вас реализовать именно его, так как тестирующая система проверяет точное равенство получающихся деревьев.

- **Формат ввода / входного файла (input.txt).** Входной файл содержит описание двоичного дерева, а также ключа вершины, которую требуется удалить из дерева.

  В первой строке файла находится число N – число вершин в дереве. В последующих N строках файла находятся описания вершин дерева. В $(i+1)$-ой строке файла $(1 \le i \le N)$ находится описание i-ой вершины, состоящее из трех чисел K_i, L_i, R_i, разделенных пробелами – ключа K_i в i-ой вершине, номера левого L_i ребенка i-ой вершины ($i < L_i \le N$ или $L_i = 0$, если левого ребенка нет) и номера правого R_i ребенка i-ой вершины ($i < R_i \le N$ или $R_i = 0$, если правого ребенка нет). Все ключи различны. Гарантируется, что данное дерево является деревом поиска.

  В последней строке содержится число X – ключ вершины, которую требуется удалить из дерева. Гарантируется, что такая вершина в дереве существует.
- **Ограничения на входные данные.** $1 \le N \le 2 \cdot 10^5$, $|K_i| \le 10^9$, $|X| \le 10^9$.
- **Формат вывода / выходного файла (output.txt).** Выведите в том же формате дерево после осуществления операции удаления. Нумерация вершин может быть произвольной при условии соблюдения формата.
- **Ограничение по времени. 2 сек.**
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
3 4 2 3 3 0 0 5 0 0 4	2 3 0 2 5 0 0

- Проверить можно по ссылке, OpenEdu, курс "Алгоритмы программирования и структуры данных 7 неделя, 4 задача.

16 Задача. K-й максимум [2 s, 512 Mb, 3 балла]

Напишите программу, реализующую структуру данных, позволяющую добавлять и удалять элементы, а также находить k-й максимум.

- **Формат ввода / входного файла (input.txt).** Первая строка входного файла содержит натуральное число n – количество команд. Последующие n строк содержат по одной команде каждая. Команда записывается в виде двух чисел c_i и k_i – тип и аргумент команды соответственно. Поддерживаемые команды:
  - +1 (или просто 1): Добавить элемент с ключом k_i.
  - 0 : Найти и вывести k_i-й максимум.
  - -1 : Удалить элемент с ключом k_i.

  Гарантируется, что в процессе работы в структуре не требуется хранить элементы с равными ключами или удалять несуществующие элементы. Также гарантируется, что при запросе ki-го мак- симума, он существует.
- **Ограничения на входные данные.** $n \leq 100000$, $|k_i| \leq 10^9$.
- **Формат вывода / выходного файла (output.txt).** Для каждой команды нулевого типа в выходной файл должна быть выведена строка, содержащая единственное число – k_i-й максимум.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 512 мб.
- Пример:

input.txt	output.txt
11	7
+1 5	5
+1 3	3
+1 7	10
0 1	7
0 2	3
0 3	
-1 5	
+1 10	
0 1	
0 2	
0 3	

17 Задача. Множество с суммой [120 s, 512 Mb, 3 балла]

В этой задаче ваша цель – реализовать структуру данных для хранения набора целых чисел и быстрого вычисления суммы элементов в заданном диапазоне.

Реализуйте такую структуру данных, в которой хранится набор целых чисел S и доступны следующие операции:

- add(i) – добавить число i в множество S. Если i уже есть в S, то ничего делать не надо;
- del(i) – удалить число i из множества S. Если i нет в S, то ничего делать не надо;
- find(i) – проверить, есть ли i во множестве S или нет;
- sum(l, r) – вывести сумму всех элементов v из S таких, что $l \le v \le r$.

- **Формат ввода / входного файла (input.txt).** Изначально множество S пусто. Первая строка содержит n – количество операций. Следующие n строк содержат операции. Однако, чтобы убедиться, что ваше решение может работать в режиме онлайн, каждый запрос фактически будет зависеть от результата последнего запроса суммы. Обозначим $M = 1\,000\,000\,001$. В любой момент пусть x будет результатом последней операции суммирования или просто 0, если до этого операций суммирования не было. Тогда каждая операция будет являться одной из следующих:
  - «+ i» – добавить некое число в множество S. Но не само число i, а число $((i + x) \bmod M)$.
  - «– i» – удалить из множества S, т.е. del$((i + x) \bmod M)$.
  - «? i» – find$((i + x) \bmod M)$.
  - «s l r» – вывести сумму всех элементов множества S из определенного диапазона, т.е. sum$((l+x) \bmod M, (r+x) \bmod M)$.
- **Ограничения на входные данные.** $1 \le n \le 100000$, $1 \le i \le 10^9$.
- **Формат вывода / выходного файла (output.txt).** Для каждого запроса «find», выведите только «Found» или «Not found» (без кавычек, первая буква заглавная) в зависимости от того, есть ли число $((i + x) \bmod M)$ в S или нет.

  Для каждого запроса суммы «sum» выведите сумму всех значений v из S из диапазона $(l + x) \bmod M \le v \le (r + x) \bmod M$, где x – результат подсчета прошлой суммы «sum», или 0, если еще не было таких операций.
- **Ограничение по времени. 120 сек. Python**
- Ограничение по памяти. 512 мб.
- Пример:

input	output
15 ? 1 + 1 ? 1 + 2 s 1 2 + 1000000000 ? 1000000000 - 1000000000 ? 1000000000 s 999999999 1000000000 - 2 ? 2 - 0 + 9 s 0 9	Not found Found 3 Found Not found 1 Not found 10

- Пояснение. В первом примере для первых 5 операций $x = 0$. Для следующих 5 операций $x = 3$, и для оставшихся 5 операций $x = 1$. Добавление повторяющегося числа дважды не меняет множество. Попытки удалить элемент, которого во множестве нет, игнорируются.
- Еще примеры:

input	output
5 ? 0 + 0 ? 0 - 0 ? 0	Not found Found Not found

input	output
5 + 491572259 ? 491572259 ? 899375874 s 310971296 877523306 + 352411209	Found Not found 491572259

- Что делать. Используйте splay дерево для эффективного хранения множества, и добавления, удаления и поиска элементов. Для каждого узла дерева дополнительно сохраните сумму всех элементов поддерева этого узла. Не забывайте обновлять эту сумму каждый раз при изменении дерева...

18 Задача. Веревка [120 s, 512 Mb, 5 баллов]

В этой задаче вы реализуете Веревку (или Rope) – структуру данных, которая может хранить строку и эффективно вырезать часть (подстроку) этой строки и вставлять ее в другое место. Эту структуру данных можно улучшить, чтобы она стала персистентной, то есть чтобы разрешить доступ к предыдущим версиям строки. Эти свойства делают ее подходящим выбором для хранения текста в текстовых редакторах.

Это очень сложная задача, более сложная, чем почти все предыдущие сложные задачи этого курса.

Вам дана строка S, и вы должны обработать n запросов. Каждый запрос описывается тремя целыми числами i, j, k и означает вырезание подстроки $S[i...j]$ (здесь индексы i и j в строке считаются от 0) из строки и вставка ее после k-го символа оставшейся строки (как бы символы в оставшейся строке нумеруются с 1). Если $k = 0$, $S[i...j]$ вставляется в начало. Дополнительные пояснения смотрите в примерах.

- **Формат ввода / входного файла (input.txt).** В первой строке ввода содержится строка S. Вторая строка содержит количество запросов n. Следующие n строк содержат по три целых числа i, j, k.
- **Ограничения на входные данные.** Строка S содержит только английские строчные буквы. $1 \le |S| \le 300000$, $1 \le n \le 100000$, $0 \le i \le j \le |S| - 1$, $0 \le k \le |S| - (j - i + 1)$.
- **Формат вывода / выходного файла (output.txt).** Выведите строку после выполнения n запросов.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input	output.txt
hlelowrold 2 1 1 2 6 6 7	helloworld

- Пояснение. *hlelowrold* $\to$ *hellowrold* $\to$ *helloworld*

  Здесь $i = j = 1$, $S[i..j] = l$, и эту букву надо вставить после $k = 2$ второго символа оставшейся строки *helowrold*, что в итоге дает *hellowrold*. Далее, $i = j = 6$, $S[i..j] = r$ и эту букву надо вставить после 7 символа оставшейся строки *hellowold*, получается *helloworld*.

- Пример:

input	output.txt
abcdef 2 0 1 1 4 5 0	efcabd

- Пояснение. *abcdef* $\to$ *cabdef* $\to$ *efcabd*.
- Что делать. Используйте splay дерево для хранения строки. Используйте методы разделения и слияния splay дерева, чтобы вырезать и вставлять подстроки. Подумайте, что должно храниться в качестве ключа в splay дереве.