

Сдать задание нужно до 26 ноября 2018. (9:00).

Контекст: <https://contest.yandex.ru/contest/9657/enter/>

Ведомость:

Задача 1. Порядок обхода (3 балла)

Дано число $N < 10^6$ и последовательность целых чисел из $[-2^{31}..2^{31}]$ длиной N .

Требуется построить бинарное дерево, заданное наивным порядком вставки.

Т.е., при добавлении очередного числа K в дерево с корнем $root$, если $root \rightarrow Key \leq K$, то узел K добавляется в правое поддерево $root$; иначе в левое поддерево $root$. Балансировку выполнять не требуется.

1_1. Выведите элементы в порядке in-order (слева направо).

in	out
3 2 1 3	1 2 3
3 1 2 3	1 2 3
3 3 1 2	1 2 3

1_2. Выведите элементы в порядке pre-order (сверху вниз).

in	out
3 2 1 3	2 1 3
3 1 2 3	1 2 3
3 3 1 2	3 1 2
4 3 1 4 2	3 1 2 4

1_3. Выведите элементы в порядке post-order (снизу вверх).

in	out
3 2 1 3	1 3 2
3 1 2 3	3 2 1

3 3 1 2	2 1 3
------------	-------

1_4. Выведите элементы в порядке level-order (по слоям, “в ширину”).

in	out
3 2 1 3	2 1 3
3 1 2 3	1 2 3
3 3 1 2	3 1 2
4 3 1 4 2	3 1 4 2

Задача 2. Декартово дерево (4 балла)

Дано число $N < 10^6$ и последовательность пар целых чисел из $[-2^{31}..2^{31}]$ длиной N .

Построить декартово дерево из N узлов, характеризующихся парами чисел $\{X_i, Y_i\}$.

Каждая пара чисел $\{X_i, Y_i\}$ определяет ключ X_i и приоритет Y_i в декартовом дереве.

Добавление узла в декартово дерево выполняйте второй версией алгоритма, рассказанного на лекции:

- При добавлении узла (x, y) выполняйте спуск по ключу до узла P с меньшим приоритетом. Затем разбейте найденное поддерево по ключу x так, чтобы в первом поддереве все ключи меньше x , а во втором больше или равны x . Получившиеся два дерева сделайте дочерними для нового узла (x, y) . Новый узел вставьте на место узла P .

Построить также наивное дерево поиска по ключам X_i методом из задачи 1.

2_1. Вычислить разницу глубин наивного дерева поиска и декартового дерева. Разница может быть отрицательна.

in	out
10 5 11 18 8 25 7 50 12 30 30 15 15 20 10 22 5 40 20 45 9	2

10 38 19 37 5 47 15 35 0 12 3 0 42 31 37 21 45 30 26 41 6	2
---	---

2_2. Вычислить количество узлов в самом широком слое декартового дерева и количество узлов в самом широком слое наивного дерева поиска. Вывести их разницу. Разница может быть отрицательна.

in	out
10 5 11 18 8 25 7 50 12 30 30 15 15 20 10 22 5 40 20 45 9	1
10 38 19 37 5 47 15 35 0 12 3 0 42 31 37 21 45 30 26 41 6	1

Задача 3. Использование AVL-дерева (5 баллов)

3_1. Солдаты. В одной военной части решили построить в одну шеренгу по росту. Т.к. часть была далеко не образцовая, то солдаты часто приходили не вовремя, а то их и вовсе приходилось выгонять из шеренги за плохо начищенные сапоги. Однако солдаты в процессе прихода и ухода должны были всегда быть выстроены по росту – сначала самые высокие, а в конце – самые низкие. За расстановку солдат отвечал прапорщик, который заметил интересную особенность – все солдаты в части разного роста. Ваша задача состоит в том, чтобы помочь прапорщику правильно расставлять солдат, а именно для каждого проходящего солдата указывать, перед каким солдатом в строю он должен

становится. Требуемая скорость выполнения команды - $O(\log n)$.

Формат входных данных.

Первая строка содержит число N – количество команд ($1 \leq N \leq 30\,000$). В каждой следующей строке содержится описание команды: число 1 и X если солдат приходит в строй (X – рост солдата, натуральное число до 100 000 включительно) и число 2 и Y если солдата, стоящим в строю на месте Y надо удалить из строя. Солдаты в строю нумеруются с нуля.

Формат выходных данных.

На каждую команду 1 (добавление в строй) вы должны выводить число K – номер позиции, на которую должен встать этот солдат (все стоящие за ним двигаются назад).

in	out
5	0
1 100	0
1 200	2
1 50	1
2 1	
1 150	

3_2. Порядковые статистики. Дано число N и N строк. Каждая строка содержит команду добавления или удаления натуральных чисел, а также запрос на получение k -ой порядковой статистики. Команда добавления числа A задается положительным числом A , команда удаления числа A задается отрицательным числом “- A ”. Запрос на получение k -ой порядковой статистики задается числом k . Требуемая скорость выполнения запроса - $O(\log n)$.

in	out
5	40
40 0	40
10 1	10
4 1	4
-10 0	50
50 2	

Задача 4. Адаптивный алгоритм сжатия данных Хаффмана (8 баллов)

Напишите две функции для создания архива из одного файла и извлечения файла из архива. Дополнительная память $O(1)$.

```
// Метод архивирует данные из потока original
void Encode(IInputStream& original, IOutputStream& compressed);
// Метод восстанавливает оригинальные данные
void Decode(IInputStream& compressed, IOutputStream& original);
```

где:

```
typedef unsigned char byte;
```

```
struct IInputStream {  
    // Возвращает false, если поток закончился  
    bool Read(byte& value) = 0;  
};
```

```
struct IOutputStream {  
    void Write(byte value) = 0;  
};
```

В контекст необходимо отправить .cpp файл содержащий функции Encode, Decode, а также включающий файл Huffman.h. Тестирующая программа выводит размер сжатого файла в процентах от исходного.

Пример минимального решения:

```
#include "Huffman.h"
```

```
static void copyStream(IInputStream& input, IOutputStream& output)  
{  
    byte value;  
    while (input.Read(value))  
    {  
        output.Write(value);  
    }  
}
```

```
void Encode(IInputStream& original, IOutputStream& compressed)  
{  
    copyStream(original, compressed);  
}
```

```
void Decode(IInputStream& compressed, IOutputStream& original)  
{  
    copyStream(compressed, original);  
}
```

Соревнование

На базе задачи 4 предлагается разработать оптимальный алгоритм сжатия данных. Это может быть как усовершенствованный алгоритм Хаффмана, так и любой другой алгоритм.

Лучшие 3 решения из каждой группы оцениваются в 15, 10 и 5 баллов соответственно.

Качество алгоритма оценивается по степени сжатия данных.

В качестве данных для сжатия выступают: текстовые документы, картинка в формате bmp, картинка в формате jpg.