Сдать задание нужно до 28 ноября 2016г. (9:00).

Контест: <https://contest.yandex.ru/contest/3082/enter/>

Ведомость: <https://drive.google.com/open?id=17mbUAdW-KCbcHwErDdBd4VHZwTIP28aAsOaKuZzVAjI>

# Задача 1. Порядок обхода (3 балла)

Дано число N < 106 и последовательность целых чисел из [-231..231] длиной N.

Требуется построить бинарное дерево, заданное наивным порядком вставки.

Т.е., при добавлении очередного числа K в дерево с корнем root, если root→Key ≤ K, то узел K добавляется в правое поддерево root; иначе в левое поддерево root.

**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 < 106 и последовательность пар целых чисел из [-231..231] длиной N.

Построить декартово дерево из N узлов, характеризующихся парами чисел {Xi, Yi}.

Каждая пара чисел {Xi, Yi} определяет ключ Xi и приоритет Yi 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 Xi методом из задачи 2.

**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 АВЛ-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 приходящего солдата указывать, перед каким солдатом в строе он должен становится. Требуемая скорость выполнения команды - O(log n).

Формат входных данных.

Первая строка содержит число N – количество команд (1 ≤ N ≤ 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 строе на месте Y надо удалить из строя. Солдаты в строе 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  1 100  1 200  1 50  2 1  1 150 | 0  0  2  1 |

**3\_2. Порядковые статистики.** Дано число N и N строк. Каждая строка содержащит 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 0  10 1  4 1  -10 0  50 2 | 40  40  10  4  50 |

# Задача 4. Алгоритм сжатия данных Хаффмана (8 баллов)

Напишите две функции для создания архива из одного файла и извлечения файла из архива.

// Метод архивирует данные из потока original

void Encode(IInputStream& original, IOutputStream& compressed);

// Метод восстанавливает оригинальные данные

void Decode(IInputStream& compressed, IOutputStream& original);

где:

typedef char byte;

interface IInputStream {

// Возвращает false, если поток закончился

virtual bool Read(byte& value) = 0;

};

interface IOutputStream {

virtual void Write(byte value) = 0;

};

В архиве сохраняйте дерево Хаффмана и код Хаффмана от исходных данных.  
Дерево Хаффмана требуется хранить эффективно - не более 10 бит на каждый 8-битный символ.

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

Лучшие 3 решения с Kc <= 50% из каждой группы оцениваются в 15, 10 и 5 баллов соответственно.

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);

}