Разработка и анализ алгоритмов. Кучи

A. Операции с кучей

1 секунда, 256 мегабайт

Недавно Петя начал изучать структуру данных «Двоичная куча».

Куча, с которой он работает, позволяет выполнять следующие операции:

- добавить число в кучу;
- узнать минимальное число в куче;
- удалить минимальное число из кучи.

Таким образом, в любой момент куча содержит в себе набор чисел (возможно, пустой), среди которых, возможно, есть одинаковые.

Чтобы лучше запомнить материал, Петя взял пустую кучу и стал применять к ней различные операции. Он не хотел ничего упустить, поэтому все выполненные операции он аккуратно записывал в журнал в порядке их выполнения, соблюдая следующий формат:

- `insert` x — в кучу было добавлено число x;
- `getMin` x — был выполнен запрос минимума, результатом запроса было число x;
- `removeMin` — из кучи было удалено текущее минимальное число (только одно вхождение).

Последовательность операций была корректной, то есть на момент выполнения операций `getMin` и `removeMin` в куче находился хотя бы один элемент.

Пока Петя обедал, его младший брат Вова забежал к нему в комнату и вырвал из журнала несколько случайных страниц, так как ему не из чего было строить бумажные кораблики.

В какой-то момент Вова понял, что Петина последовательность операций могла стать некорректной. Например, если выполнять оставшиеся операции в том же порядке, в каком они были записаны, то результат операций `getMin` может не совпадать с результатом, записанным в журнале, а какие-то из операций `getMin` и `removeMin` вообще не могут быть выполнены, так как на момент их выполнения куча пуста.

Теперь Вове нужно дописать в произвольные места в журнале какие-нибудь операции так, чтобы последовательность снова стала корректной, то есть чтобы результат всех операций `getMin` совпадал с результатом, записанным в журнале, и все операции могли быть корректно выполнены. Вова хочет сделать это как можно быстрее, пока Петя не вернулся с обеда, то есть дописать в журнал минимально возможное количество операций. Любую операцию разрешается дописывать в начало последовательности, между любыми двумя операциями или в конец последовательности операций.

Помогите Вове решить эту проблему.

Входные данные

В первой строке входных данных записано число n ($1 \le n \le 100\,000$) — количество записей, оставшихся в журнале Пети.

В каждой из следующих n строк записана одна из оставшихся в журнале записей об операциях с кучей в формате, описанном выше. Все числа во входных данных целые и не превышают 10^9 по абсолютной величине.

Выходные данные

В первой строке выведите целое число m — минимальное количество операций в исправленной последовательности.

В следующих m строках выведите эту последовательность операций по одной операции в каждой строке в формате, описанном выше. Все числа должны быть целыми и не превышать 10^9 по абсолютной величине.

Обратите внимание, что исправленная последовательность должна содержать в себе последовательность операций из входных данных в качестве **подпоследовательности**.

Гарантируется, что существует корректная последовательность операций, длина которой не превосходит $1\,000\,000$.

входные данные
2 insert 3 getMin 4
выходные данные
4 insert 3 removeMin insert 4 getMin 4

входные данные
4 insert 1 insert 1 removeMin getMin 2

выходные данные
6 insert 1 insert 1 removeMin removeMin insert 2 getMin 2

В первом примере после добавления в кучу числа 3 минимумом будет являться единственное число 3, и для того, чтобы результатом операции `getMin` было число 4, нужно удалить из кучи 3 (как минимум) и затем добавить 4.

Во втором примере число 1 добавляется дважды, и удалить его нужно также дважды.

B. Менеджер памяти-1

1 секунда, 512 мегабайт

Пете поручили написать менеджер памяти для новой стандартной библиотеки языка φ++. В распоряжении у менеджера находится массив из N последовательных ячеек памяти, пронумерованных от 1 до N. Задача менеджера – обрабатывать запросы приложений на выделение и освобождение памяти. Запрос на выделение памяти имеет один параметр K. Такой запрос означает, что приложение просит выделить ему K последовательных ячеек памяти. Если в распоряжении менеджера есть хотя бы один свободный блок из K последовательных ячеек, то он обязан в ответ на запрос выделить такой блок. При этом непосредственно перед самой первой ячейкой памяти выделяемого блока не должно располагаться свободной ячейки памяти. После этого выделенные ячейки становятся занятыми и не могут быть использованы для выделения памяти, пока не будут освобождены. Если блока из K последовательных свободных ячеек нет, то запрос отклоняется. Запрос на освобождение памяти имеет один параметр T. Такой запрос означает, что менеджер должен освободить память, выделенную ранее при обработке запроса с порядковым номером T. Запросы нумеруются, начиная с единицы. Гарантируется, что запрос с номером T – запрос на выделение, причем к нему еще не применялось освобождение памяти. Освобожденные ячейки могут снова быть использованы для выделения памяти. Если запрос с номером T был отклонен, то текущий запрос на освобождение памяти игнорируется. Требуется написать менеджер памяти, удовлетворяющий приведенным критериям.

Входные данные

Первая строка входного файла содержит числа N и M – количество ячеек памяти и количество запросов соответственно ($1 \le N \le 2^{31} - 1$; $1 \le M \le 10^5$). Каждая из следующих M строк содержит по одному числу: $(i+1)$-я строка входного файла ($1 \le i \le M$) содержит либо положительное число K, если i-й запрос – запрос на выделение с параметром K ($1 \le K \le N$), либо отрицательное число $-T$, если i-й запрос – запрос на освобождение с параметром T ($1 \le T < i$).

Выходные данные

Для каждого запроса на выделение памяти выведите в выходной файл результат обработки этого запроса: для успешных запросов выведите номер первой ячейки памяти в выделенном блоке, для отклоненных запросов выведите число -1. Результаты нужно выводить в порядке следования запросов во входном файле.

входные данные
42 9 7 3 8 -2 6 5 -5 9 4

выходные данные
1 8 11 19 25 30 19

входные данные
128 12 1 2 4 -2 8 -3 16 -5 32 -7 64 -1

выходные данные
1 2 4 8 16 32 64