Задача 1. Сортировка кучей

Источник:	базовая*
Имя входного файла:	`input.bin`
Имя выходного файла:	`output.bin`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В первых четырёх байтах входного файла задано целое число N — количество чисел в массиве A. Далее идут N четырёхбайтовых целых чисел — содержимое массива A. Размер массива лежит в диапазоне: $0 \leqslant N \leqslant 500\,000$.

Требуется отсортировать массив A по неубыванию, используя **алгоритм сортировки кучей**. В простейшем случае алгоритм состоит в том, чтобы добавить все элементы массива в двоичную кучу, а затем последовательно извлекать из кучи минимальный/максимальный элемент и записывать обратно в массив.

В выходной файл нужно вывести ровно N четырёхбайтовых целых чисел: содержимое массива A после сортировки.

Пример

input.bin
0A 00 00 00 1F 00 00 00 F2 FF FF FF 06 00 00 00 04 00 00 00 26 00 00 00 FD FF FF FF 1E 00 00 00 F6 FF FF FF 0A 00 00 00 F4 FF FF FF
output.bin
F2 FF FF FF F4 FF FF FF F6 FF FF FF FD FF FF FF 04 00 00 00 06 00 00 00 0A 00 00 00 1E 00 00 00 1F 00 00 00 26 00 00 00

Задача 2. Сортировка деревом поиска

Источник:	базовая∗
Имя входного файла:	`input.bin`
Имя выходного файла:	`output.bin`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

В первых четырёх байтах входного файла задано целое число N — количество чисел в массиве A. Далее идут N четырёхбайтовых целых чисел — содержимое массива A. Размер массива лежит в диапазоне: $0 \leqslant N \leqslant 500\,000$.

Требуется отсортировать массив A по неубыванию, используя **дерево поиска**. Учтите, что в исходном массиве может быть много одинаковых элементов. Кроме того, элементы массива могут быть изначально выстроены в каком-то фиксированном порядке.

В выходной файл нужно вывести ровно N четырёхбайтовых целых чисел: содержимое массива A после сортировки.

Пример

input.bin
0A 00 00 00 1F 00 00 00 F2 FF FF FF 06 00 00 00 04 00 00 00 26 00 00 00 FD FF FF FF 1E 00 00 00 F6 FF FF FF 0A 00 00 00 F4 FF FF FF
output.bin
F2 FF FF FF F4 FF FF FF F6 FF FF FF FD FF FF FF 04 00 00 00 06 00 00 00 0A 00 00 00 1E 00 00 00 1F 00 00 00 26 00 00 00

Задача 3. sql join

Источник:	базовая
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Многие приложения используют базы данных для постоянного хранения информации.

На сегодняшний день наиболее популярны реляционные базы данных. В реляционной базе данные представляются в виде таблиц. Каждая таблица состоит из произвольного набора записей, каждая запись занимает одну строку таблицы. Запись можно воспринимать как структуру языка C: в ней есть фиксированный набор полей фиксированного типа. Каждый столбец таблицы содержит значение одного конкретного поля для всех записей. Таким образом, в таблице строки задают отдельные записи, а столбцы — поля этих записей.

Для извлечения и фильтрации данных из реляционной базы данных чаще всего используют язык SQL. При этом нередко пользователю базы данных нужно составить сборный отчёт по нескольким таблицам. В таком случае можно использовать операции соединения (join) таблиц. В данной задаче предлагается реализовать операцию Inner Join для двух конкретных таблиц.

Первая таблица содержит биографии известных актёров кино. Она была создана следующей командой SQL:

```
CREATE TABLE ActorBio (
  Name varchar(30),
  BirthYear int,
  Country varchar(10)
);
```

Первый столбец называется `Name` и хранит имя актёра. Во втором записан год рождения как целое число. А третьем столбце записана страна, в которой жил актёр.

Вторая таблица содержит информацию о том, какой актёр в каких фильмах играл. Она была создана командой:

```
CREATE TABLE ActorInMovie (
  ActorName varchar(30),
  MovieName varchar(20)
);
```

Первый столбец содержит имя актёра `ActorName`, а второй — название кино, в котором этот актёр играл роль.

От вас требуется реализовать следующий SQL-запрос:

```
SELECT *
FROM ActorBio INNER JOIN ActorInMovie
ON ActorBio.Name = ActorInMovie.ActorName
```

Результатом этой операции является одна таблица, в которой пять полей: первые три поля взяты из таблицы `ActorBio`, а последние два — из `ActorInMovie`.

Механизм выполнения операции следующий:

1. Перебираем все пары записей A и B, где A взята из таблицы `ActorBio`, а B — из таблицы `ActorInMovie`.

2. Для каждой пары проверяем условие соединения: что имя актёра `Name` в записи A совпадает с именем актёра `ActorName` в записи B.
3. Если условие выполнено, то конкатенируем записи A и B и полученную запись с пятью полями добавляем в таблицу-результат.

Чтобы было проще понять, как работает операция, крайне рекомендуется посмотреть первый пример к задаче.

Формат входных данных

В первой строке входного файла записано целое число N — количество записей в таблице `ActorBio` ($1 \leqslant N \leqslant 10^5$). Далее идёт N строк, в которых записаны записи таблицы `ActorBio`. Затем записано целое число M — количество записей в таблице `ActorInMovie` ($1 \leqslant M \leqslant 10^5$). Далее идёт M строк, в которых записаны записи таблицы `ActorInMovie`.

Для каждой записи в файле записываются все её поля через пробел, в том порядке, в котором они определены. Все записи кроме года рождения строковые: они окружены символом двойной кавычки (ASCII 34) с обеих сторон. Каждое строковое значение непустое и может содержать в себе любые печатные символы ASCII (коды от 32 до 126 включительно) кроме символа двойной кавычки. Год рождения записан как целое число. Имена актёров не длиннее 30 символов, названия стран не длиннее 10 символов, названия фильмов не длиннее 20 символов.

Формат выходных данных

Требуется вывести ровно K строк: записи, получившиеся в таблице-результате после соединения. Каждая строка должна описывать одну запись с пятью полями, в том же формате, в котором эти записи записаны во входных данных. Порядок записей в выходном файле может быть любой.

Гарантируется, что $N \cdot M \leqslant 10^5$.

Пример

input.txt

```
8
"Peter Falk" 1927 "USA"
"Oleg Tabakov" 1935 "USSR"
"Andrei Mironov" 1941 "USSR"
"Arnold Schwarzenegger" 1947 "USA"
"Jean Reno" 1948 "France"
"Sharon Stone" 1958 "USA"
"Tom Cruise" 1962 "USA"
"Ryoko Hirosue" 1980 "Japan"
12
"Sharon Stone" "Basic Instinct"
"Jean Reno" "Mission: Impossible"
"Arnold Schwarzenegger" "Total Recall"
"Tom Cruise" "Mission: Impossible"
"Andrei Mironov" "Twelve Chairs"
"Sharon Stone" "Total Recall"
"Ryoko Hirosue" "Wasabi"
"Arnold Schwarzenegger" "Terminator"
"Jean Reno" "Wasabi"
"Peter Falk" "Colombo"
"Anatoli Papanov" "Twelve Chairs"
"Jean Reno" "Leon"
```

output.txt

```
"Andrei Mironov" 1941 "USSR" "Andrei Mironov" "Twelve Chairs"
"Arnold Schwarzenegger" 1947 "USA" "Arnold Schwarzenegger" "Total
Recall"
"Arnold Schwarzenegger" 1947 "USA" "Arnold Schwarzenegger" "Terminator"
"Jean Reno" 1948 "France" "Jean Reno" "Leon"
"Jean Reno" 1948 "France" "Jean Reno" "Mission: Impossible"
"Jean Reno" 1948 "France" "Jean Reno" "Wasabi"
"Peter Falk" 1927 "USA" "Peter Falk" "Colombo"
"Ryoko Hirosue" 1980 "Japan" "Ryoko Hirosue" "Wasabi"
"Sharon Stone" 1958 "USA" "Sharon Stone" "Basic Instinct"
"Sharon Stone" 1958 "USA" "Sharon Stone" "Total Recall"
"Tom Cruise" 1962 "USA" "Tom Cruise" "Mission: Impossible"
```

input.txt

```
5
"O]V| c -(SZ9mY ~‘/{8" 1950 "bo"
" Q G*u4 T:Eqy’d" 1979 "9"
"0" 2005 "jMsB"
" w cQ" 1982 "&"
" Q G*u4 T:Eqy’d" 2003 " J|hL"
7
" Q G*u4 T:Eqy’d" "6q*EDh!"
"0" "s"
" Q G*u4 T:Eqy’d" "CQMG::dw{"
":D h%$  W^cr" "%‘De!Si"
" Q G*u4 T:Eqy’d" "]Zo"
" Q G*u4 T:Eqy’d" "t"
"0" "Uxb/.& "
```

output.txt

```
" Q G*u4 T:Eqy’d" 1979 "9" " Q G*u4 T:Eqy’d" "t"
" Q G*u4 T:Eqy’d" 1979 "9" " Q G*u4 T:Eqy’d" "CQMG::dw{"
" Q G*u4 T:Eqy’d" 1979 "9" " Q G*u4 T:Eqy’d" "]Zo"
" Q G*u4 T:Eqy’d" 1979 "9" " Q G*u4 T:Eqy’d" "6q*EDh!"
" Q G*u4 T:Eqy’d" 2003 " J|hL" " Q G*u4 T:Eqy’d" "t"
" Q G*u4 T:Eqy’d" 2003 " J|hL" " Q G*u4 T:Eqy’d" "CQMG::dw{"
" Q G*u4 T:Eqy’d" 2003 " J|hL" " Q G*u4 T:Eqy’d" "]Zo"
" Q G*u4 T:Eqy’d" 2003 " J|hL" " Q G*u4 T:Eqy’d" "6q*EDh!"
"0" 2005 "jMsB" "0" "Uxb/.& "
"0" 2005 "jMsB" "0" "s"
```

Задача 4. sql join: хеширование

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Данная задача является продолжением задачи «sql join». Прочитайте условие оригинальной задачи перед тем, как читать дальше!

Допустим, в первой таблице N записей, во второй — M записей, а в таблице-результате R записей. Легко видеть, что если имя актёра одинаковое во всех записях обеих таблиц, то в результате соединения в таблице окажется $R = NM$ записей. На практике такой случай довольно бессмысленный, обычно таблица-результат по количеству записей примерно такая же, как таблицы-аргументы: то есть $R = O(M + N)$. В таком случае для ускорения операции соединения следует использовать такие структуры данных и алгоритмы, чтобы выполнить соединение быстрее чем за $O(MN)$.

В данной задаче необходимо использовать **хеш-таблицу** для ускорения соединения. Запишите все записи одной таблицы в хеш-таблицу, в которой ключом является имя актёра. Затем переберите все записи другой таблицы: для каждой записи хеш-таблица позволяет быстро найти все совпадающие записи из первой таблицы.

Входные/выходные данные в этой задаче такие же, как в задаче «sql join». Ограничения такие же ($1 \leqslant N, M \leqslant 10^5$), за одним важным исключением:

В этой задаче **не** гарантируется, что $N \cdot M \leqslant 10^5$. Вместо этого гарантируется, что после соединения количество записей R не превышает 10^5.

Задача 5. sql join: бинарный поиск

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Данная задача является продолжением задачи «sql join». Прочитайте условие оригинальной задачи перед тем, как читать дальше!

В данной задаче необходимо использовать сортировку и **бинарный поиск** для ускорения соединения. Отсортируйте все записи одной таблицы по имени актёра. Далее переберите все записи другой таблицы: для каждой записи можно бинарным поиском найти первую запись с тем же именем, после чего легко обнаружить их все.

Входные/выходные данные в этой задаче такие же, как в задаче «sql join». Ограничения такие же ($1 \leqslant N, M \leqslant 10^5$), за одним важным исключением:

В этой задаче **не** гарантируется, что $N \cdot M \leqslant 10^5$. Вместо этого гарантируется, что после соединения количество записей R не превышает 10^5.

Задача 6. sql join: слияние

Источник:	основная*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Данная задача является продолжением задачи «sql join». Прочитайте условие оригинальной задачи перед тем, как читать дальше!

В данной задаче необходимо использовать сортировку и **слияние** для ускорения соединения. Отсортируйте обе таблицы по имени актёра. Теперь вы можете двигать два указателя, каждый по своей таблице, точно так же, как в алгоритме слияния. Если по указателям имена совпадают, надо найти полностью совпадающие группы и добавить попарно их конкатенации в ответ, а если не совпадают, то надо указатель на запись с меньшим именем сдвинуть вправо.

Входные/выходные данные в этой задаче такие же, как в задаче «sql join». Ограничения такие же ($1 \leqslant N, M \leqslant 10^5$), за одним важным исключением:

В этой задаче **не** гарантируется, что $N \cdot M \leqslant 10^5$. Вместо этого гарантируется, что после соединения количество записей R не превышает 10^5.

Задача 7. Максимум в окне

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда∗
Ограничение по памяти:	разумное

Дан массив A, в котором записано N целых чисел. По этому массиву перемещается окно. Окно — это подотрезок в массиве, начинающийся с L-ого элемента массива, и заканчивающийся на (R-1)-ом элементе (всего в окне $R - L$ элементов). При этом счётчики L и R постоянно изменяются. Нужно после каждого изменения счётчиков найти и вывести максимальное число в окне.

Изначально оба счётчика равны 0. Далее нужно выполнить $2N - 1$ операций, каждая операция имеет один из двух типов:

- `L` — увеличить счётчик L на 1, тем самым сдвинув начало окна.
- `R` — увеличить счётчик R на 1, тем самым сдвинув конец окна.

После выполнения каждой операции нужно вывести максимум среди чисел в окне, то есть максимум среди элементов массива, индекс которых попадает в диапазон $[L, R)$.

Формат входных данных

В первой строке содержится целое число N — количество элементов в массиве ($1 \leqslant N \leqslant 2 \cdot 10^5$). Во второй строке записано N целых чисел через пробел — содержимое массива. Все числа по абсолютной величине не превышают 10^9. В третьей и последней строке записано подряд $(2N - 1)$ символов — команды, которые требуется выполнить.

Гарантируется, что:

1. первая команда имеет тип `R`,
2. после выполнения каждой команды верно $R > L$,
3. окно всегда входит в массив, то есть $R \leqslant N$.

Формат выходных данных

Нужно вывести $2N - 1$ строк, в каждой из которых требуется записать максимум в текущем окне. Максимум нужно выводить после обработки каждой команды.

Пример

input.txt	output.txt
14 1 8 3 2 5 2 7 3 7 4 9 1 3 2 RRRRLLRLRRLLRRLLLRRRRLRLLLL	1 8 8 8 8 3 5 5 5 7 7 7 7 7 7 7 7 7 7 9 9 9 9 9 9 3 3 2

Задача 8. Топологическая сортировка++

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда*
Ограничение по памяти:	разумное

Имеется N лекционных тем, пронумерованных числами от 1 до N. За одну лекцию можно целиком рассказать одну тему, смешивать темы на лекции нельзя. Некоторые темы зависят от других тем. Если тема A зависит от темы B, то это означает, что нельзя рассказывать тему A, не рассказав предварительно тему B. Требуется определить, в каком порядке нужно рассказывать темы на лекции, чтобы соблюсти все зависимости.

Если есть несколько подходящих порядков, нужно выбрать лексикографически наименьший из них. То есть номер первой рассказываемой темы должен быть минимально возможный, среди таких порядков нужно выбрать такой, у которого номер темы на второй лекции минимально возможный, далее нужно минимизировать номер темы на третьей лекции, и так далее.

Указание: Зависимости нужно хранить не в матрице, а в виде массива списков (связных списков или растущих массивов). Кроме этого, нужно ускорить алгоритм Кана до асимптотики $O(M + NlogN)$.

Формат входных данных

В первой строке записано два целых числа: N — количество тем и M — количество зависимостей ($1 \leqslant N \leqslant 100\,000$, $1 \leqslant M \leqslant 200\,000$).

В оставшихся M строках описаны зависимости. Каждая зависимость описывается двумя целыми числами B и A, что означает, что тема A зависит от темы B ($1 \leqslant A \neq B \leqslant N$).

Формат выходных данных

Выведите искомый порядок в единственную строку выходного файла: N целых чисел, обозначающих номера тем. Если в темах имеется циклическая зависимость и искомого порядка не существует, выведите слова “`bad course`” вместо порядка.

Пример

input.txt	output.txt
5 7 1 5 1 3 5 2 5 4 3 4 1 2 4 2	1 3 5 4 2
3 3 1 2 2 3 3 2	bad course

Задача 9. sql join: дерево поиска

Источник:	повышенной сложности*
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Данная задача является продолжением задачи «sql join». Прочитайте условие оригинальной задачи перед тем, как читать дальше!

В данной задаче необходимо использовать **дерево поиска** для ускорения соединения. Занесите все записи одной таблицы в бинарное дерево поиска (при этом нужно решить, что делать с равными ключами). Далее переберите все записи второй таблицы: для каждой из них можно найти в дереве поиска список всех записей с совпадающим именем.

Входные/выходные данные в этой задаче такие же, как в задаче «sql join». Ограничения такие же ($1 \leqslant N, M \leqslant 10^5$), за одним важным исключением:

В этой задаче **не** гарантируется, что $N \cdot M \leqslant 10^5$. Вместо этого гарантируется, что после соединения количество записей R не превышает 10^5.