

### Min3. Третий минимум

Имя входного файла: `min3.in`  
Имя выходного файла: `min3.out`

Реализуйте структуру данных для хранения целых чисел, поддерживающую следующие операции:

- добавить элемент;
- извлечь минимальный элемент;
- извлечь второй по величине элемент.
- извлечь третий по величине элемент.

(заметьте, что извлекаемый элемент удаляется)

#### Формат входного файла

Первая строка входного файла содержит количество запросов  $n$  ( $1 \leq n \leq 100\,000$ ). Следующие  $n$  строк содержат запросы в следующем формате:

- `Insert( $X$ )` — добавить число  $X$ ;
- `GetMin` — извлечь минимум;
- `GetMin2` — извлечь второй по величине элемент;
- `GetMin3` — извлечь третий по величине элемент.

Гарантируется, что при запросе на извлечение элемента, соответствующий элемент существует.

#### Формат выходного файла

Для каждого запроса `GetMin`, `GetMin2` и `GetMin3` требуется вывести соответствующий извлеченный элемент.

#### Пример

<code>min3.in</code>	<code>min3.out</code>
11	2
<code>Insert(1)</code>	3
<code>Insert(2)</code>	4
<code>Insert(3)</code>	1
<code>Insert(4)</code>	4
<code>Insert(1)</code>	
<code>GetMin3</code>	
<code>GetMin3</code>	
<code>Insert(4)</code>	
<code>GetMin3</code>	
<code>GetMin</code>	
<code>GetMin2</code>	

### Condense 2. Конденсация графа

Имя входного файла: `condense2.in`  
Имя выходного файла: `condense2.out`

Требуется найти количество ребер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных ребер.

#### Формат входного файла

Первая строка входного файла содержит два натуральных числа  $n$  и  $m$  — количество вершин и ребер графа соответственно ( $n \leq 10\,000$ ,  $m \leq 100\,000$ ). Следующие  $m$  строк содержат описание ребер, по одному на строке. Ребро номер  $i$  описывается двумя натуральными числами  $b_i$ ,  $e_i$  — началом и концом ребра соответственно ( $1 \leq b_i, e_i \leq n$ ). В графе могут присутствовать кратные ребра и петли.

#### Формат выходного файла

Первая строка выходного файла должна содержать одно число — количество ребер в конденсации графа.

#### Пример

<code>condense2.in</code>	<code>condense2.out</code>
4 4 2 1 3 2 2 3 4 3	2

## Post. Почтальон

Имя входного файла: `post.in`  
Имя выходного файла: `post.out`

В городе есть  $n$  площадей, соединенных улицами. При этом количество улиц не превышает ста тысяч и существует не более трех площадей, на которые выходит нечетное число улиц. Для каждой улицы известна ее длина. По улицам разрешено движение в обе стороны. В городе есть хотя бы одна улица. От любой площади до любой можно дойти по улицам.

Почтальону требуется пройти хотя бы один раз по каждой улице так, что бы длина его пути была наименьшей. Он может начать движение на любой площади и закончить также на любой (в том числе и на начальной).

## Формат входного файла

Первая строка входного файла содержит натуральное число  $n$  — количество площадей в городе ( $1 \leq n \leq 1000$ ). Далее следуют  $n$  строк, задающих улицы. В  $i$ -ой из этих строк находится число  $m_i$  — количество улиц, выходящих из площади  $i$ . Далее следуют  $m_i$  пар положительных чисел. В  $j$ -ой паре первое число — номер площади, в которую идет  $j$ -ая улица с  $i$ -ой площади, а второе число — длина этой улицы.

Между двумя площадями может быть несколько улиц, но не может быть улиц с площади на нее саму.

Все числа во входном файле не превосходят  $10^5$ .

## Формат выходного файла

Если решение существует, то в первую строку выходного файла выведите одно число — количество улиц в искомом маршруте (считая первую и последнюю), а во вторую — номера площадей в порядке их посещения.

Если решений нет, выведите в выходной файл одно число -1.

Если решений несколько, выведите любое.

## Пример

post.in	post.out
4	5
2 2 1 2 2	1 2 3 4 2 1
4 1 2 4 4 3 5 1 1	
2 2 5 4 8	
2 3 8 2 4	