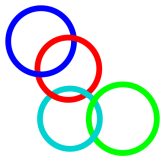
Парашютные кольца

Ранняя и довольно сложная версия того, что мы сегодня называем парашютом описана в работе Леонардо *Codex Atlanticus* (ca. 1485). Парашют Леонардо состоял из ткани, натянутой на деревянный каркас в форме пирамиды.

Связанные кольца

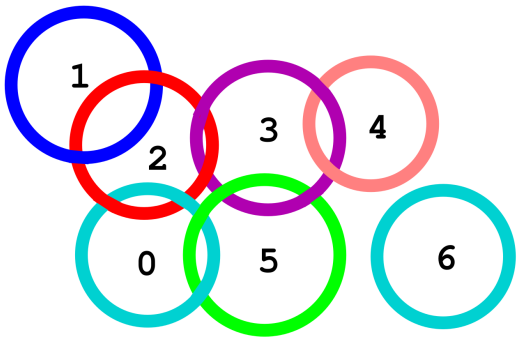
Парашютист Адриан Николас испытал проект Леонардо через 500 лет. Для этого, современная облегченная конструкция соединила парашют Леонардо с телом человека. Мы хотим использовать соединенные кольца, у которых еще предусмотрены крюки для закрепления колец к ткани одежды. Каждое кольцо сделано из гибкого и прочного материала. Кольца легко соединяются вместе, так как каждое кольцо может быть открыто и снова закрыто. Особая конфигурация соединенных колец - это "цепь". "Цепь" - это последовательность одного или более колец в которой каждое кольцо соединено только с двумя соседними кольцами, кроме первого и последнего, которые соединены только с одним другим кольцом, как показано ниже. Обратите внимание, что одно кольцо - тоже цепь.



Другие конфигурации также возможны, поскольку кольцо может быть соединено с тремя или более другими кольцами. Мы говорим что кольцо "критическое", если после его открытия и удаления, все оставшиеся кольца образуют множество цепей(или не остается других колец). Другими словами, после удаления этого кольца, должны остаться только цепи.

Пример

На следующем рисунке представлены 7 колец, пронумерованных от 0 до 6. Имеется 2 критических кольца. Одно критическое кольцо - это кольцо номер 2, так как после его удаления оставшиеся кольца образуют цепи [1], [0, 5, 3, 4] и [6]. Другое критическое кольцо с номером 3, так как после его удаления оставшиеся кольца образуют цепи [1, 2, 0, 5], [4] и [6]. Если мы удалим любое другое кольцо, мы не получим множество несвязанных цепей. Например, после удаления кольца номер 5, хотя мы и имеем цепь [6], но связанные кольца 0, 1, 2, 3 и 4 не образуют цепь.



Постановка задачи

Ваша задача - вычислять количество критических колец в конфигурации, полученной в результате взаимодействия с вашей программой.

В начале, имеется некоторое количество попарно не связанных колец. Затем, кольца связываются вместе. В любой момент времени у вас могут запросить вернуть количество критических колец в текущей конфигурации. А именно, вы должны реализовать три процедуры.

- `Init`(N) — эта процедура вызывается ровно один раз в начале, чтобы сообщить, что в начальной конфигурации имеется N попарно не связанных колец, пронумерованных от 0 до N-1 включительно.
- `Link`(A, B) — два кольца с номерами A и B соединяются. Гарантируется, что A и B различные, и еще не соединены непосредственно; кроме этого, нет дополнительных условий на A и B, в частности, нет условий, возникающих из физических ограничений. Ясно, что `Link`(A, B) и `Link`(B, A) эквивалентны.
- `CountCritical`() — возвращает количество критических колец для текущей конфигурации связанных колец.

Пример

Рассмотрим наш рисунок с N = 7 колец и предположим, что они сначала не связаны. Мы покажем возможную последовательность вызовов процедур, где после последнего вызова мы получаем ситуацию, изображенную на рисунке.

Вызовы	Возвращаемые значения
`Init`(7)	
`CountCritical`()	7
`Link`(1, 2)	
`CountCritical`()	7
`Link`(0, 5)	
`CountCritical`()	7
`Link`(2, 0)	
`CountCritical`()	7
`Link`(3, 2)	
`CountCritical`()	4
`Link`(3, 5)	
`CountCritical`()	3
`Link`(4, 3)	
`CountCritical`()	2

Подзадача 1 [20 баллов]

- $N \leq 5\,000$.
- Функция `CountCritical` вызывается только один раз, после всех других вызовов; Функция `Link` вызывается не более 5 000 раз.

Подзадача 2 [17 баллов]

- $N \leq 1\,000\,000$.
- Функция `CountCritical` вызывается только один раз, после всех других вызовов; Функция `Link` вызывается не более 1 000 000 раз.

Подзадача 3 [18 баллов]

- $N \leq 20\,000$.
- Функция `CountCritical` вызывается не более 100 раз; Функция `Link` вызывается не более 10 000 раз.

Подзадача 4 [14 баллов]

- $N \leq 100\,000$.
- Функция `CountCritical` и `Link` вызываются в сумме не более 100 000 раз.

Подзадача 5 [31 балл]

- $N \leq 1\,000\,000$.

- Функция `CountCritical` и `Link` вызываются в сумме не более 1 000 000 раз.

Детали реализации

Вы должны отправить на проверку ровно один файл, с именем `rings.c`, `rings.cpp` или `rings.pas`. Этот файл должен содержать реализацию подпрограмм используя следующие описания (сигнатуры):

Программы на C/C++

```
void Init(int N);
void Link(int A, int B);
int CountCritical();
```

Программы на Pascal

```
procedure Init(N : LongInt);
procedure Link(A, B : LongInt);
function CountCritical() : LongInt;
```

Эти подпрограммы должны вести себя как описано выше. Конечно, вы можете реализовывать любые другие подпрограммы для вашего внутреннего использования. Отсылаемые вами на проверку решения не должны никаким образом использовать стандартный поток ввода/вывода или любые другие файлы.

Пример проверяющего модуля (grader)

Пример проверяющего модуля(grader) читает входные данные в следующем формате:

- Строка 1 содержат два числа N, L;
- Строки со 2 по (L + 1) должны содержать:
  - -1 для вызова процедуры `CountCritical`;
  - A, B для вызова процедуры `Link` с параметрами A,B.

Пример проверяющего модуля (grader) будет выводить все результаты вызовов функции `CountCritical`.