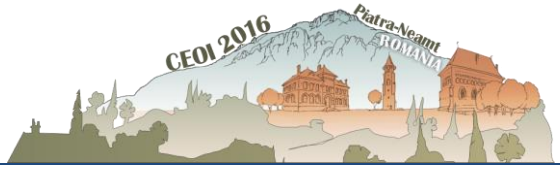


Астронавтът Астро току-що е започнал своето пребиваване на Международната Космическа Станция (ISS). Неговата задача е да изучава цивилизацията Терран от планетата Map Capa. Терранците вече са построили N града, номерирани с числата от 1 до N, но те все още не са построили нито едно шосе между тях.
Едновременно с началото на наблюдението на Астро, Терранците стартират строежа на шосета на Map Capa. Известно е, че Терранците искат да свържат всички градове с пътища колкото е възможно по-бързо и, поради тази причина, те не искат да строят пряко шосе между два града, които вече са свързани с път, състоящ се от едно или повече преки шосета. Всеки път, когато Терранците построяват ново шосе, Астро иска да знае кои градове свързва то преди да бъде построено следващото шосе.
За щастие на Астро, той има достъп до SETI Masterpieces – нов сателит, който може да наблюдава Map Capa и да вижда какви шосета са построени. При зададени две непресичащи се множества от градове, SETI може да каже дали съществува поне едно директно шосе, което свързва град от едното множество с град от другото.

Работата, която трябва да се извърши интерактивно е следната:

1. Терранците строят ново шосе.
2. Астро използва SETI в опит да открие новопостроеното шосе.
3. Той се явява в научния комитет на ISS и докладва своето мнение за това кое е новопостроеното шосе.
4. Ако има по-малко от N-1 построени нови шосета, се преминава към стъпка 1.

Вашата задача е да напишете програма, която ще реши проблемите на Астро.

Задача

Вие трябва да напишете функция *run()*, която ще се извика веднаж, в началото на програмата. От тази функция вие трябва да викате функцията *query()* всеки път, когато искате да използвате SETI. Всеки път, когато откриете най-новото построено шосе, вие трябва да извикате функцията *setRoad()*. Вие можете да викате *query()* няколко пъти преди да извикате *setRoad()*.

Функция на грейдера: `query()`

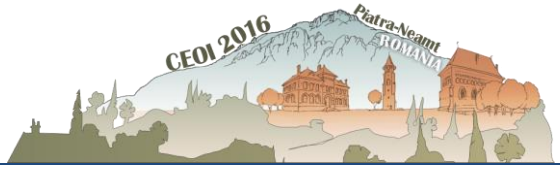
- C/C++: `int query(int size_a, int size_b, int a[], int b[]);`

Извиквайте тази функция за да отправите запитване към сателита SETI. Параметрите `size_a` и `size_b` задават размерите на двете множества от градове, за които се отнася запитването. Масивите `a[]` и `b[]` трябва да съдържат номерата на градовете в първото и второто множество съответно. ***Тези множества трябва да не се пресичат.*** Функцията връща **1**, ако съществува поне едно пряко шосе между град от първото множество и град от второто множество, и **0** в противен случай.

Функция на грейдера: `setRoad()`

- C/C++: `void setRoad(int a, int b);`
-

Извикайте тази функция, за да съобщите, че откритото от вас новопостроено шосе свързва градовете с номера *a* и *b*. Ако шосето е открито неправилно, то вие ще получите 0 точки за този тест и програмата прекъсва работата си. Ако това е `(N-1)`-то извикване на тази функция, програмата прекъсва работата си и връща резултат за този тест. В противен случай Террианците строят ново шосе и интерактивната работа продължава.



Следващите последователни извиквания на на *query()* ще се отнасят за шосето, построено след извикването на *setRoad()*.

Вашата функция: `run()`

- C/C++: `void run(int N);`

Файлът, който събмитвате, трябва да имплементира тази функция. Вие трябва алтернативно да викате *query()* и *setRoad()* от тази функция. Параметърът N, чийто стойност получавате, задава броя на градовете, които Терианците са построили. Ако тази функция завърши изпълнението си преди да бъдат открити всички пътища, вие няма да получите точки за този тест.

Бележки по реализацията

- C/C++: файлът, който събмитвате, трябва да включва `#include "icc.h"`

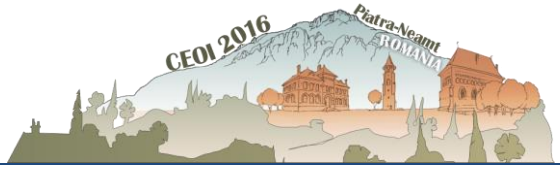
Оценяване

Тестовете за тази задача са групирани в 6 групи. Тестовете от една група имат една и съща стойност за N. Вие ще получите полагаемите се точки за групата, ако за всеки тест от нея откриете правилно всички шосета, използвайки най-много M извиквания на функцията *query()*. Стойностите на N и M, както и точките за всяка група, са дадени в следващата таблица.

Номер на група	N = Брой на градовете	M = Максимален брой извиквания на *query()*	Точки
1	15	1500	7
2	50	2500	11
3	100	2250	22
4	100	2000	21
5	100	1775	29
6	100	1625	10

Примерно изпълнение

Действие на състезателя	Действие на грейдера	Бележка
-	run(4)	- Грейдерът стартира функция run за 4 града. Първото шосе е построено между градове 2 и 4 и не е известно на състезателя (трябва да бъде открито от него).
query(1, 3, {1}, {2, 3, 4})	return 0	- Запитване за множества от градове {1} и {2, 3, 4}. Отговорът е "false": няма шосе от 1 до никой от градовете от второто множество.
query(1, 2, {2}, {3, 4})	return 1	- Запитване за множества от градове {2} и {3, 4}. Отговорът е "true": има шосе от град 2 до град 4.
query(1, 1, {2}, {3})	return 0	
setRoad(2, 4)	-	- Състезателят определя правилно шосе (2, 4). Грейдерът генерира ново шосе между 1 и 3.
query(2, 2, {2, 4}, {1, 3})	return 0	- Запитване за множества от градове {2, 4} и {1, 3}. Отговорът е "false".



setRoad(1, 3)	-	- Състезателят определя правилно шосе (1, 3). Грейдерът генерира ново шосе между 1 и 4.
query(2, 2, {2, 4}, {1, 3})	return 1	- Същото запитване като предното. Сега отговорът е „true”, поради наличието на последното построено шосе.
query(1, 2, {2}, {1, 3})	return 0	
query(1, 1, {4}, {3})	return 0	
setRoad(4, 1)	exit	- Състезателят е определил коректно последното построено шосе (4,1). Грейдерът приема последното откритие и дава полагаемите се точки за теста.