Чика иска да измами

Име на задачата	Измама
Вход	Интерактивна задача
Изход	Интерактивна задача
Ограничение по време	2 секунди
Ограничение по памет	512 MB

Чика има тесте от q карти за игра, номерирани с различни естествени числа. Тя иска да играе различни игри с нейните приятели от ученическия съвет на академията на Shuchi'in, позната на някои от състезанието преди 2 седмици :). Понеже Чика иска да печели, то тя е решила да маркира гърба на картите в тестето ѝ.

Всички карти са във формата на квадрат с размер 2×2, като долния ляв ъгъл има координати $(0, 0)$, а горния десен - $(2, 2)$. Чика чертае определен шаблон на гърба на всяка карта, така че по-късно да знае кое е числото, написано на картата само като види съответния шаблон. Тя рисува шаблоните като използва следната процедура: произволен брой пъти (може и 0) избира две различни точки A и B, които имат целочислени координати спрямо долния ляв ъгъл на картата и чертае **отсечка** между тях.

Имайте предвид, че Чика ще чертае само **валидни** отсечки, т.е. такива които свързват две точки A и B и за които няма друга точка C (различни от A и B) с цели координати, която лежи на отсечката. Например, отсечката между точките $(0, 0)$ и $(2, 2)$ **не е валидна**, защото съдържа точката $(1, 1)$. Докато отсечките между точките $(0, 0)$ и $(1, 1)$ и между точките $(1, 1)$ и $(2, 2)$ са **валидни** и дори Чика може да ги нарисува двете заедно на един шаблон. Вземете под внимание, че отсечките нямат посока: отсечка, начертана от точка A към B е **същата** като отсечката, начертана в обратна посока - от B към A.

Много е важно, че Чика иска да разпознае нейните карти без значение как са завъртяни. Всяка карта може да бъде завъртяна на 0, 90, 180 или 270 градуса в посока обратна на часовниковата стрелка спрямо началната ориентация.

Вашата задача е да помогнете на Чика да начертае шаблоните на q-те карти в нейното тесте и да разпознае тези карти по-късно.

Детайли по имплементацията

*Това е задача тип комуникация в две стъпки, **<u>всяка от които включва отделно изпълнение на вашата програма</u>***. Трябва да имплементирате две функции:

- Функция `BuildPattern`, която връща шаблона, начертан на гърба на дадена карта. Тази функция ще бъде извикана q пъти по време на първата стъпка.
- Функция `GetCardNumber`, която връща числото на карта по шаблона, начертан на първата стъпка (възможно и завъртян). Тази функция ще бъде извикана q пъти по време на втората стъпка.

Първата функция трябва да има следния формат

```
std::vector<std::pair<std::pair<int, int>, std::pair<int, int>>> BuildPattern(int n);
```

и да приема като параметър единствено n, което е числото записано на картата. Тази функция трябва да върне `std::vector`, съдържащ отсечките, които Чика ще начертае на гърба на съответната карта и ще разпознава по-късно. Всяка отсечка се представя като `std::pair` от две точки и всяка точка се представя като `std::pair` (x, y) от цели числа - координати спрямо долния ляв ъгъл на картата (разбира се $0 \leq x, y \leq 2$). Всички отсечки, които Чика чертае, за текущия шаблон са валидни и различни. Гарантирано е, че всички q извиквания са за различни стойности на n.

След като са получени всички шаблони за q карти, грейдърът може да извърши една от следните операции, произволен брой пъти, за всеки шаблон:

- Шаблонът се завърта на 0, 90, 180 или 270 градуса по посока обратна на часовниковата стрелка.
- Редът на отсечките във `std::vector`, представящ шаблона, се променя.
- Редът на крайните точки на отсечка в шаблона се променя. (отсечка, която е начертана от точка A до B може да стане отсечка от B до A.)

Втората функция трябва да има следния формат

```
int GetCardNumber(std::vector<std::pair<std::pair<int, int>, std::pair<int, int>>> p);
```

и да приема като параметър единствено p, което е `std::vector` от отсечки, описващи шаблона на Чика на гърба на картата (а този шаблон съответства на върнатата стойност от по-ранното извикване на вашата функция `BuildPattern`). Тази функция трябва да върне числото n, записано на картата. Имайте предвид, че шаблонът p може да не е точно по начина, по който сте го задали като върната стойност на функцията `BuildPattern`; може да е претърпял някоя от трите операции, описани по-горе. Също е възможно реда на картите

на тази стъпка да е различен от реда на първата стъпка. Гарантирано е, че всяка карта ще бъде използвана точно веднъж.

Ограничения

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ за всички извиквания на функцията `BuildPattern`.
- Имайте предвид, че съществува алгоритъм, който да направи подходящи шаблони за разпознаване на $67\,000\,000$ различни числа на картите.

Подзадачи

- Подзадача 1 (2 точки): $n \leq 2$.
- Подзадача 2 (9 точки): $n \leq 25$.
- Подзадача 3 (15 точки): $n \leq 1\,000$ и грейдърът **няма да завърта** шаблоните между първа и втора стъпка. (но той **може** да извършва някоя от другите две операции.)
- Подзадача 4 (3 точки): $n \leq 16\,000\,000$ и грейдърът **няма да завърта** шаблоните между първа и втора стъпка. (но той **може** да извършва някоя от другите две операции.)
- Подзадача 5 (24 точки): $n \leq 16\,000\,000$.
- Подзадача 6 (18 точки): $n \leq 40\,000\,000$.
- Подзадача 7 (29 точки): Няма допълнителни ограничения.

Примерно взаимодействие

Извикване на функция	Върната стойност	Обяснение
Първата стъпка започва.	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	Трябва да направим шаблон за числото 3 на карта с размери 2×2. Решаваме да начертаем следните 2 отсечки: - между точките $(0,0)$ и $(2,1)$, - между точките $(1,1)$ и $(2,0)$.
`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	Трябва да направим шаблон за числото 1 на карта с размери 2×2. Решаваме да начертаем следната отсечка: - между точките $(0,1)$ и $(0,0)$.
Първата стъпка завършва.	-	-
Втората стъпка започва.	-	-
`GetCardNumber(`	`1`	Получаваме шаблон, съставен от

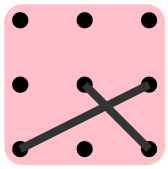
`{{{0, 0}, {0, 1}}})`		една отсечка: - между точките $(0,0)$ и $(0,1)$. Това е същият шаблон, който бихме получили от начертаването на отсечка: - между точките $(0,1)$ и $(0,0)$ което е точно шаблонът, който направихме при второто извикване на функцията `BuildPattern`. Затова връщаме числото 1.

`GetCardNumber(`
`{{{1, 1}, {2, 2}},`
`{{1, 2}, {2, 0}}})`
	`3`	Получаваме шаблон, съставен от две отсечки:
		- между точките $(1,1)$ и $(2,2)$,
		- между точките $(1,2)$ и $(2,0)$.
		Това е шаблонът, който направихме при първото извикване на функцията `BuildPattern`, завъртян на 90 градуса по посока обратна на часовниковата стрелка. Затова връщаме числото 3.

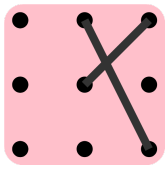
Втората стъпка завършва.	-	-

Следващите три фигури описван следното:

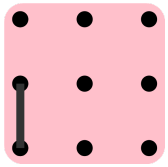
- Шаблонът върнат от първото извикване на функцията `BuildPattern`:



- Шаблонът получен, като параметър на второто извикане на `GetCardNumber` (който всъщност е първият направен шаблон, завъртян на 90 градуса по посока обратна на часовниковата стрелка).



- Шаблонът върнат от второто извикване на функцията `BuildPattern`, който представлява същата фигура, която описва параметърът на първото извикване на `GetCardNumber`.



Примерен грейдър

Предоставен е примерен грейдър, `grader.cpp`, в прикачения файл `Cheat.zip`. Той прочита от стандартния вход цялото число q, след което изпълнява следните q стъпки:

- Прочита цяло число n от стандартния вход.
- Изпълнява `BuildPattern(n)` и съхранява върната стойност в променлива p.
- Изпълнява `GetCardNumber(p)` и отпечатва върнатата стойност на стандартния изход.

Можете да променяте примерния грейдър, както пожелаете.

За да компилирате примерния грейдър с вашето решение, може да използвате следната команда от конзолата:

```
g++ -std=gnu++11 -O2 -o solution grader.cpp cheat.h solution.cpp
```

където `solution.cpp` е името на сорс файла с вашето решение (който трябва да бъде пратен на системата CMS). След като компилирате, за да изпълните изпълнимия файл с примерния вход, прикачен в архива, използвайте следната команда:

```
./solution < input.txt
```

Имайте предвид, че примерният грейдър е различен от истинския грейдър, който ще бъде използван на CMS и съответно ще извършва първата и втората стъпка в различни изпълнения на вашата програма.