Chika Quer Trapacear

Nome do Problema	Chika Quer Trapacear
Arquivo de Entrada	Tarefa interativa
Arquivo de Saída	Tarefa interativa
Limite de tempo	2 segundos
Limite de memória	512 megabytes

Chika tem um baralho com q cartas numeradas com vários inteiros positivos. Ela quer jogar alguns jogos com seus amigos do conselho estudantil da *Shuchi'in Academy* usando estas cartas, mas ela também quer ganhar, então ela decide marcar secretamente a parte de trás das cartas em seu baralho.

As cartas são todas quadrados de tamanho 2×2, onde o canto inferior esquerdo tem as coordenadas $(0, 0)$ e o canto superior direito tem as coordenadas $(2, 2)$. Chika desenha um certo padrão no verso de cada carta, para que mais tarde ela saiba, olhando o padrão, qual número está escrito na frente da carta. Ela desenha tal padrão usando o seguinte procedimento: Quantas vezes ela quiser (possivelmente 0 vezes), ela escolhe dois pontos distintos A e B que têm coordenadas inteiras relativas ao canto inferior esquerdo da carta e desenha um **segmento de linha reta** entre eles.

Chika só desenha segmentos **válidos**, ou seja, segmentos entre dois pontos A e B para os quais não há nenhum outro ponto C (diferente de A e B) com coordenadas inteiras que também se encontra no segmento. Por exemplo, o segmento entre $(0, 0)$ e $(2, 2)$ **não é válido** porque contém o ponto $(1, 1)$, mas segmentos entre $(0, 0)$ e $(1, 1)$ e entre $(1, 1)$ e $(2, 2)$ são ambos **válidos** e Chika pode até mesmo desenhar ambos no mesmo padrão. Além disso, observe que os segmentos não têm direção: Um segmento desenhado de A para B é **idêntico** tanto a si mesmo quanto ao segmento desenhado na direção inversa, de B a A.

O mais importante é que Chika quer ter certeza de que ela reconhecerá suas cartas independentemente de como elas são rotacionadas. Uma carta pode ser rotacionada 0, 90, 180 ou 270 graus no sentido anti-horário com relação à sua orientação original.

Sua tarefa é ajudar Chika a projetar os padrões para as q cartas em seu baralho e depois reconhecer essas cartas mais tarde.

Implementação

Esta é uma tarefa interativa com duas etapas, ***cada etapa envolve uma execução separada de seu programa***. Você precisa implementar duas funções:

- Uma função `BuildPattern` que retorna o padrão a ser desenhado no verso de uma determinada carta. Esta função será chamada q vezes na primeira etapa.
- Uma função `GetCardNumber` que retorna o número de uma carta (possivelmente rotacionada) que possui um determinado padrão desenhado na primeira etapa. Esta função será chamada q vezes na segunda etapa.

A primeira função

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

recebe um único parâmetro n, o número que está escrito na frente da carta. Você precisa retornar um `std::vector` contendo os segmentos que Chika desenha como um padrão no verso da carta para reconhecê-la mais tarde. Um segmento é representado como um `std::pair` de pontos, e um ponto é representado como um `std::pair` (x, y) de coordenadas inteiras relativas ao canto inferior esquerdo da carta, onde $0 \leq x, y \leq 2$. Todos os segmentos que a Chika desenha precisam ser válidos e não idênticos par a par. É garantido que todas as q chamadas a BuildPattern recebem valores diferentes para o parâmetro n.

Depois de receber todos os padrões para as q cartas, o corretor pode fazer qualquer uma das seguintes operações, qualquer número de vezes, em cada um dos padrões:

- Rotacionar o padrão inteiro em 0, 90, 180 ou 270 graus no sentido anti-horário.
- Modificar a ordem dos segmentos no `std::vector` que representa o padrão.
- Alterar a ordem das pontas de um segmento no padrão. (Um segmento desenhado de A para B pode tornar-se o segmento idêntico de B para A).

A segunda função,

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

recebe um único parâmetro p, o `std::vector` de segmentos descrevendo o padrão que é desenhado por Chika no verso da carta, com base no valor retornado por uma chamada anterior à sua função `BuildPattern`. A função deve retornar o número n escrito na frente da carta. Lembre-se que o padrão p não está necessariamente na forma original que é retornada por `BuildPattern`; ele pode ter sido submetido às três operações mencionadas acima. Também é possível que a ordem das cartas seja diferente da ordem em que foram dadas na primeira etapa, mas é garantido que cada carta será usada exatamente uma vez.

Restrições

- $1 \leq q \leq 10\,000$.
- $1 \leq n \leq 67\,000\,000$ para todas as chamadas à função `BuildPattern`.
- Note que existem algoritmos para construir padrões tais que é possível reconhecer $67\,000\,000$ cartas distintas.

Pontuação

- Subtarefa 1 (2 pontos): $n \leq 2$.
- Subtarefa 2 (9 pontos): $n \leq 25$.
- Subtarefa 3 (15 pontos): $n \leq 1\,000$ e o corretor **não rotacionará** os padrões entre as etapas 1 e 2. (O corretor **pode** executar as outras duas operações.)
- Subtarefa 4 (3 pontos): $n \leq 16\,000\,000$ e o corretor **não rotacionará** os padrões entre as etapas 1 e 2. (O corretor **pode** executar as outras duas operações.)
- Subtarefa 5 (24 pontos): $n \leq 16\,000\,000$.
- Subtarefa 6 (18 pontos): $n \leq 40\,000\,000$.
- Subtarefa 7 (29 pontos): Nenhuma restrição adicional.

Exemplo de Interação

Chamada da Função	Valor Retornado	Explicação
Primeira etapa começa.	-	-
`BuildPattern(3)`	`{{{0, 0}, {2, 1}}, {{1, 1}, {2, 0}}}`	Temos que criar um padrão para o número 3 na carta de tamanho 2×2. Decidimos desenhar 2 segmentos: - entre $(0, 0)$ e $(2, 1)$, - entre $(1, 1)$ e $(2, 0)$.
`BuildPattern(1)`	`{{{0, 1}, {0, 0}}}`	Temos que criar um padrão para o número 1 na carta de tamanho 2×2. Decidimos desenhar 1 segmento: - entre $(0, 1)$ e $(0, 0)$.
Primeira etapa termina.	-	-
Segunda etapa começa.	-	-
`GetCardNumber( {{{0, 0}, {0, 1}}})`	`1`	Obtemos um padrão composto por 1 segmento: - entre (0, 0) e (0, 1). Este é o mesmo padrão que obteríamos com o desenho do segmento: - entre (0, 1) e (0, 0) que é exatamente o mesmo padrão

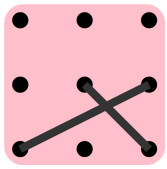
		com a mesma orientação (rotacionado em 0 graus) que retornamos na segunda chamada da função `BuildPattern`. Portanto, nós retornamos 1.

`GetCardNumber(`
`{{{1, 1}, {2, 2}},`
`{{1, 2}, {2, 0}}})`
	`3`	Obtemos um padrão composto por 2 segmentos:
		- entre $(1, 1)$ e $(2, 2)$,
		- entre $(1, 2)$ e $(2, 0)$.
		Este é o padrão que retornamos na primeira chamada à função `BuildPattern`, rotacionado em 90 graus no sentido anti-horário. Portanto, retornamos 3.

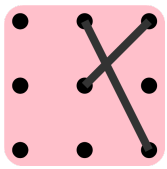
Segunda etapa termina.	-	-

As três imagens a seguir representam, nessa ordem:

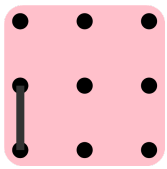
- O padrão retornado como saída pela primeira chamada de `BuildPattern`:



- O padrão recebido como parâmetro pela segunda chamada de `GetCardNumber`, que é o primeiro padrão após ser rotacionado em 90 graus no sentido anti-horário.



- O padrão retornado como saída da segunda chamada de `BuildPattern`, que é também o mesmo padrão recebido como argumento da primeira chamada de `GetCardNumber`.



Corretor Exemplo

O corretor exemplo fornecido, `grader.cpp`, no anexo da tarefa `Cheat.zip` lê um número inteiro q da entrada padrão e depois faz os seguintes passos q vezes:

- Lê um número inteiro n da entrada padrão.
- Chama `BuildPattern(n)` e armazena o valor retornado em uma variável p.
- Chama `GetCardNumber(p)` e imprime o valor retornado na saída padrão.

Você pode modificar seu corretor localmente, se desejar.

Para compilar o corretor exemplo com sua solução, você pode usar o seguinte comando no terminal de comando:

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

onde `solution.cpp` é seu arquivo de solução a ser submetido no CMS. Para executar o programa com a entrada de exemplo fornecida no anexo, digite o seguinte comando a partir do terminal de comando:

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

Por favor, note que, ao contrário do corretor exemplo, o corretor real no CMS executará a primeira etapa e a segunda etapa em execuções separadas de seu programa.