Projektowanie zabawek

Nazwa zadania	Projektowanie zabawek
Wejście	Zadanie interaktywne
Wyjście	Zadanie interaktywne
Limit czasu	1 sekunda
Limit pamięci	256 MB

Staś pracuje dla firmy, która projektuje zabawki. Proces tworzenia nowej zabawki wygląda następująco: dane jest n szpilek, ponumerowanych od 1 do n, wystających z pudełka. Pewne pary szpilek są połączone kabelkami wewnątrz pudełka. Innymi słowy, szpilki i kabelki tworzą graf nieskierowany, w którym szpilki są wierzchołkami, a kabelki są krawędziami. Kabelki nie są widoczne z zewnątrz i jedynym sposobem, żeby się czegoś o nich dowiedzieć, jest użycie **próbnika** na szpilkach: Staś może wybrać dwie szpilki i i j, takie że $i \neq j$, a próbnik zasygnalizuje mu, czy te szpilki są połączone wewnątrz pudełka - pośrednio lub bezpośrednio (czyli próbnik informuje, czy pomiędzy danymi szpilkami istnieje ścieżka w grafie).

Projektem zabawki nazwiemy zbiór połączeń wewnątrz pudełka.

Staś używa wyspecjalizowanego oprogramowania, żeby tworzyć projekty. To oprogramowanie działa następująco: zaczyna od pewnego projektu, który określamy jako projekt 0. Nie informuje Stasia, jakie są połączenia w pudełku dla tego projektu. Staś może wielokrotnie wykonywać następującą trzystopniową operację:

1. Wybiera numer projektu a i dwa numery szpilek i i j, takie że $i \neq j$.
2. Oprogramowanie sygnalizuje mu, co by się stało, gdyby użył próbnika na tych dwóch szpilkach. Innymi słowy, informuje go, czy szpilki i oraz j są (pośrednio lub bezpośrednio) połączone w projekcie a.
3. Ponadto jeśli szpilki nie były ani pośrednio, ani bezpośrednio połączone w projekcie a, to oprogramowanie tworzy nowy projekt, który zawiera wszystkie połączenia z projektu a oraz jedno dodatkowe bezpośrednie połączenie pomiędzy i i j. Ten projekt otrzymuje najmniejszy dostępny numer. Dokładniej, pierwszy stworzony w ten sposób projekt dostanie numer 1, potem numer 2 i tak dalej. Zauważ, że projekt a nie ulega zmianie, tylko tworzy się nowy projekt posiadający dodatkowe połączenie.

Twoim celem jest pomóc Stasiowi dowiedzieć się jak najwięcej o projekcie 0 poprzez używanie tej operacji.

Zauważ, że nie zawsze jest możliwe wyznaczenie dokładnego zbioru połączeń dla projektu 0, ponieważ nie można rozróżnić pośrednich i bezpośrednich połączeń. Rozważmy na przykład następujące dwa projekty dla $n = 3$:

Próbnik zasygnalizuje dowolną parę szpilek jako połączoną dla obu projektów, więc nie będziemy w stanie odróżnić ich, używając oprogramowania opisanego powyżej.

Twoim zadaniem jest wyznaczyć dowolny projekt równoważny projektowi 0. Dwa projekty są równoważne, gdy próbnik sygnalizuje takie same wyniki dla każdej pary szpilek w obu projektach.

Implementacja

To zadanie jest interaktywne. Twoim zadaniem jest zaimplementować funkcję

```
void ToyDesign(int n, int max_ops);
```

która wyznacza projekt *równoważny* do projektu 0. Twoja implementacja powinna osiągnąć ten cel poprzez wywoływanie dwóch poniższych funkcji zero lub więcej razy. Pierwsza z nich to

```
int Connected(int a, int i, int j);
```

gdzie $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, oraz a nie może przekroczyć liczby projektów stworzonych do tej pory. Jeśli szpilki i oraz j są (pośrednio lub bezpośrednio) połączone w projekcie a, to funkcja zwróci a. W przeciwnym przypadku zwróci liczbę projektów stworzonych dotychczas plus jeden, czyli numer przypisany do nowego projektu, który zawiera wszystkie połączenia z projektu a, a także połączenie pomiędzy i oraz j. Funkcja `Connected` może być wywołana co najwyżej `max_ops` razy.

Gdy Twój program zakończy wywołania funkcji `Connected`, powinien opisać projekt równoważny projektowi 0 i potem poprawnie zakończyć swoje wykonywanie. Aby opisać projekt, program powinien wywołać funkcję

```
void DescribeDesign(std::vector<std::pair<int,int>> result);
```

Argument `result` jest wektorem par liczb całkowitych opisujących bezpośrednie połączenia pomiędzy szpilkami. Każda para odpowiada numerom szpilek, pomiędzy którymi istnieje połączenie. Pomiędzy każdą (nieuporządkowaną) parą szpilek powinno istnieć co najwyżej jedno bezpośrednie połączenie i nie powinno być żadnego połączenia pomiędzy pewną szpilką a nią samą. Wywołanie tej funkcji kończy wykonywanie programu.

Ograniczenia

- $2 \leq n \leq 200$

Podzadania

- Podzadanie 1 (10 punktów): $n \leq 200$, *max_ops* $= 20\,000$
- Podzadanie 2 (20 punktów): $n \leq 8$, *max_ops* $= 20$
- Podzadanie 3 (35 punktów): $n \leq 200$, *max_ops* $= 2\,000$
- Podzadanie 4 (35 punktów): $n \leq 200$, *max_ops* $= 1\,350$

Przykład

Akcja zawodniczki	Akcja sprawdzaczki	Wyjaśnienie
	`ToyDesign(4, 5)`	W zabawce są 4 szpilki. Powinnaś wyznaczyć dowolny projekt równoważny projektowi 0 poprzez wywołanie `Connected` co najwyżej 5 razy.
`Connected(0, 1, 2)`	`Zwraca 1.`	Szpilki 1 i 2 nie są połączone ani bezpośrednio, ani pośrednio w projekcie 0. Tworzy się nowy projekt o numerze 1.
`Connected(1, 3, 2)`	`Zwraca 2.`	Szpilki 3 i 2 nie są połączone ani bezpośrednio, ani pośrednio w projekcie 1. Tworzy się nowy projekt o numerze 2.
`Connected(0, 3, 4)`	`Zwraca 0.`	Szpilki 3 i 4 są połączone bezpośrednio lub pośrednio w projekcie 0. Nie tworzy się żaden nowy projekt.
`DescribeDesign({{3, 4}})`	–	Opisujemy projekt, który ma tylko jedno połączenie:

		pomiędzy szpilkami o numerach 3 i 4.

Sprawdzaczka pomocnicza

Sprawdzaczka pomocnicza `grader.cpp`, znajdująca się w załączniku `ToyDesign.zip`, wczytuje dane ze standardowego wejścia w następującym formacie:

- Pierwsza linia zawiera liczbę szpilek n, liczbę kabelków m i *max_ops*.
- Kolejne m linii zawiera po dwie liczby a i b reprezentujące istnienie kabelka łączącego szpilki a i b.

Sprawdzaczka pomocnicza wczytuje dane ze standardowego wejścia i wywołuje funkcję `ToyDesign` z rozwiązania zawodniczki. Sprawdzaczka wypisze jedną z następujących wiadomości na podstawie Twojego rozwiązania:

- "Wrong answer: Number of operations exceeds the limit.", jeśli liczba wywołań `Connected` przekracza *max_ops*.
- "Wrong answer: Wrong design id.", jeśli argument a przekazany do funkcji `Connected` jest numerem projektu nieistniejącego w momencie wywołania funkcji.
- "Wrong answer: Incorrect design.", jeśli projekt opisany przez `DescribeDesign` nie jest równoważny projektowi 0.
- "OK!", jeśli projekt opisany przez `DescribeDesign` jest równoważny projektowi 0.

Aby skompilować pomocniczą sprawdzaczkę ze swoim rozwiązaniem, możesz użyć następującej komendy w terminalu:

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

gdzie `solution.cpp` jest plikiem z rozwiązaniem, który chcesz wysłać do CMS. Aby uruchomić program z przykładowym wejściem z załącznika, wpisz następującą komendę w terminalu:

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