Rotaļlietu projektēšana

Uzdevuma nosaukums	Rotaļlietu projektēšana
Ievaddati	Interaktīvs uzdevums
Izvaddati	Interaktīvs uzdevums
Laika limits	1 sekunde
Atmiņas limits	256 megabaiti

Tu strādā kompānijā, kas projektē rotaļlietas. Tiek izstrādāta jauna rotaļlieta, kas darbojas šādi: No korpusa ir izvirzītas n kājiņas, kas ir sanumurētas no 1 līdz n. Daži kājiņu pāri ir ar vadiem savienoti korpusa iekšpusē. Citiem vārdiem sakot, kājiņas un vadi veido neorientētu grafu, kurā kājiņas ir virsotnes un vadi ir šķautnes. Vadi nav redzami no ārpuses, un vienīgais veids, kā par tiem kaut ko uzzināt, ir, kājiņām izmantojot **testeri**: var izvēlēties divas kājiņas i un j tā, lai $i \neq j$, un testeris paziņos, vai korpusā šīs divas kājiņas ir savienotas (vai nu tieši, vai netieši). Tādējādi testeris norāda, vai grafā starp šīm kājiņām ir ceļš.

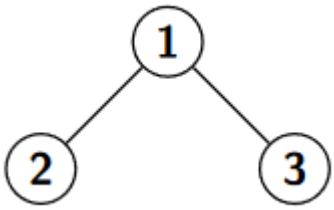
Korpusā esošo savienojumu kopu sauksim par rotaļlietas **konstrukciju**.

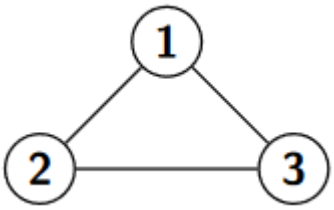
Tu izmanto īpašu programmatūru, lai veiktu vaicājumus par konstrukcijām un izveidotu jaunas konstrukcijas. Šī programmatūra darbojas šādi: Tā sāk ar kādu rotaļlietas konstrukciju, ko mēs apzīmējam kā "0-to konstrukciju". Programmatūra neparāda savienojumus korpusa iekšienē. Tā vietā Tu vari atkārtoti veikt šo triju soļu operāciju:

1. Tu izvēlies a-to konstrukciju un divus tādus kājiņu numurus i un j, ka $i \neq j$.
2. Programmatūra pasaka Tev, kas notiktu, ja izmantotu testeri šīm divām kājiņām. Citiem vārdiem, tā pasaka, vai kājiņas i un j ir (tieši vai netieši) savienotas a-tajā konstrukcijā.
3. Ja kājiņas a-tajā konstrukcijā nebija savienotas ne tieši, ne netieši, tad programmatūra izveido jaunu konstrukciju, kurā ir visi savienojumi, kas bija a-tajā konstrukcijā, plus tiešais savienojums starp i un j. Šai konstrukcijai tiek piešķirts nākamais brīvais konstrukcijas numurs (pirmajai konstrukcijai, kas tiek izveidota šādā veidā, būs 1. numurs, tad 2. numurs utt.). Jāņem vērā, ka šis nemaina a-to konstrukciju, bet tikai izveido jaunu konstrukciju, kurai ir papildu savienojums.

Mērķis ir, izmantojot šo operāciju, noskaidrot pēc iespējas vairāk par 0-to konstrukciju.

Jāņem vērā, ka ne vienmēr ir iespējams precīzi noteikt 0-tās konstrukcijas savienojumu kopu, jo nevar atšķirt tiešos un netiešos savienojumus. Piemēram, tas nav iespējams šādām divām konstrukcijām, kurām $n = 3$:





Testeris jebkuru kājiņu pāri abām konstrukcijām uzrādītu kā savienotu, tāpēc, izmantojot iepriekš aprakstīto programmatūru, tās nevarēs atšķirt.

Uzdevums ir atrast jebkuru konstrukciju, kas ir līdzvērtīga 0-tajai konstrukciju. Divas konstrukcijas ir **līdzvērtīgas**, ja testeris abās konstrukcijās uzrāda vienādus rezultātus visiem kājiņu pāriem.

Realizācija

Šis ir interaktīvs uzdevums. Ir jārealizē funkcija

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

kas nosaka konstrukciju, kas ir *līdzvērtīga* 0-tajai konstrukcijai. Šai funkcijas realizācijai jāsasniedz mērķis, izsaucot divas zemāk aprakstītās funkcijas. Pirmā funkcija, kuru var izsaukt, ir:

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

kurā $1 \leq i, j \leq n$, $i \neq j$, $a \geq 0$, un a nedrīkst pārsniegt līdz šim radīto konstrukciju skaitu. Ja kājiņas i un j konstrukcijā a ir (tieši vai netieši) savienotas, tad funkcija atgriež a. Pretējā gadījumā funkcija atgriež līdz šim radīto konstrukciju skaitu plus viens, kas kļūst par numuru, kas ir piešķirts jaunajai konstrukcijai. Jaunajai konstrukcijai ir visi konstrukcijas a savienojumi plus savienojums starp i un j.Funkcija `Connected` var tikt izsaukta ne vairāk kā `max_ops` reizes.

Kad programma ir pabeigusi funkcijas `Connected` darbības, tai ir jāapraksta konstrukcija, kas ir līdzvērtīga 0-tajai konstrukcijai. Lai aprakstītu konstrukciju, programmai jāizsauc:

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

Parametrs `result` ir veselu skaitļu pāru vektors, kas apraksta kājiņu tiešos savienojumus. Katrs pāris atbilst vienam savienojumam. Pārī jābūt savienojuma abu kājiņu numuriem. Ir jābūt vismaz vienam tiešajam savienojumam starp katru (nesakārtotu) kājiņu pāri, un nedrīkst būt tiešo savienojumu starp kājiņu un šo pašu kājiņu. Šīs funkcijas izsaukums apstādina programmas darbību.

Ierobežojumi

- $2 \leq n \leq 200$

Vērtēšana

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

Interakcijas paraugs

Funkcijas izsaukums	Atgriežamā vērtība	Paskaidrojums
	`ToyDesign(4, 20)`	Rotaļlietai ir 4 kājiņas. Ir nepieciešams noteikt jebkuru konstrukciju, kas ir līdzvērtīga 0-tajai konstrukcijai. Tas jāpaveic, ne vairāk kā 20 reizes izsaucot funkciju `Connected`.
`Connected(0, 1, 2)`	`Returns 1.`	0-tajā konstrukcijā 1. un 2. kājiņa nav savienotas ne tieši, ne netieši. Ir radīta jauna konstrukcija Nr. 1.
`Connected(1, 3, 2)`	`Returns 2.`	Pirmajā konstrukcijā 3. un 2. kājiņa nav savienotas ne tieši, ne netieši. Ir radīta jauna konstrukcija Nr. 2.
`Connected(0, 3, 4)`	`Returns 0.`	0-tajā konstrukcijā 3. un 4. kājiņa ir tieši savienotas. Jauna konstrukcija nav radīta.
`DescribeDesign({{3, 4}})`	-	Tiek aprakstīta konstrukcija, kurai ir tikai viens savienojums: 3. un 4. kājiņa.

Paraugvērtētājs

Uzdevuma pielikumā `ToyDesign.zip` esošais paraugvērtētājs `grader.cpp` ielasa ievaddatus no standarta ievades šādā formātā:

- Pirmajā rindā dots kājiņu skaits n, tiešo savienojumu skaits m un *max_ops*
- Nākamajās m rindās doti tiešie savienojumi, kā kājiņu korteži.

Paraugvērtētājs ielasa ievaddatus un no risinājuma izsauc funkciju `ToyDesign`. Paraugvērtētājs atkarībā no risinājuma uzvedības izvada vienu no šiem paziņojumiem:

- "Wrong answer: Number of operations exceeds the limit.", ja funkcijas `Connected` izsaukumu skaits pārsniedz *max_ops*
- "Wrong answer: Wrong design id.", ja funkcija `Connected` tiek izsaukta ar tādu parametru a, kurš funkcijas izsaukšanas brīdī nav nevienas konstrukcijas numurs.
- "Wrong answer: Incorrect design.", ja funkcijas `DescribeDesign` aprakstītā konstrukcija nav līdzvērtīga ar 0-to konstrukciju.
- "OK!", ja funkcijas `DescribeDesign` aprakstītā konstrukcija ir līdzvērtīga ar 0-to konstrukciju.

Lai kompilētu paraugvērtētāju ar savu risinājumu, komandrindā jāizmanto šāda komanda:

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

kur `solution.cpp` ir risinājuma fails, kas būs jāiesniedz sacensību sistēmā (CMS). Lai darbinātu programmu ar pielikumā esošo ievaddatu paraugu, komandrindā jāizmanto šāda komanda:

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