Identificare quei nodi da cui si possa raggiungere un ciclo dispari (`odd_cycle_reach`)

Dato un grafo non-orientato $G = (V, E)$, il tuo obiettivo è individuare da quali nodi $S \subseteq V$ sia possibile raggiungere un ciclo di lunghezza dispari.

+ **Precisiamo le nozioni coinvolte:** Un *grafo* è una coppia $G = (V, E)$ dove $V = V(G)$ è l'insieme finito dei suoi *nodi* ed $E = E(G)$ è l'insieme dei suoi archi. Ogni arco E è una coppia non-ordinata di nodi in V; lavoriamo cioè su grafi *non-orientati*, dove ogni arco può essere percorso in entrambi i versi. Pertanto, la presenza di un cammino $P_{u,v}$ che porti da un nodo u a un nodo v implica che esista anche un cammino da v ad u (basta percorrere $P_{u,v}$ a ritroso). Inoltre non ammettiamo *loops*, ossia $u \neq v$ per ogni arco $\{u, v\} \in E$. Un grafo G si dice *connesso* se esiste un cammino da u a v per ogni $u, v \in V(G)$. Un *sottografo* di un grafo $G = (V, E)$ è un qualsiasi grafo $G' = (V', E')$ con $V' \subseteq V$ e $U' \subseteq U$. Un *ciclo* C di G è un qualsiasi sottografo connesso di G tale che per ogni $v \in V(C)$ esistano precisamente 2 archi in $E(C)$ che contengono v. In pratica un ciclo è un cammino il cui primo ed ultimo nodo coincidano. Qualsiasi arco si tolga da un ciclo restiamo con un cammino. Un ciclo è detto *dispari* se ha un numero dispari di archi. Diremo che *un nodo* $v \in V(G)$ *raggiunge un sottografo* (V', E') *di* G se e solo se $v \in V'$ oppure esiste un $v' \in V'$ tale che G contenga un cammino da v a v'.

Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Seguono T istanze del problema, dove ogni istanza è un diverso grafo $G = (V, E)$. Per ogni istanza, la prima riga contiene due numeri interi separati da uno spazio: il numero di nodi $n = |V|$, e il numero di archi $m = |E|$. Seguono m righe ciascuna delle quali riporta un diverso arco di G. Ciascun arco viene specificato fornendo i nomi dei due nodi che collega (due numeri interi nell'intervallo $[0, n - 1]$, separati da uno spazio).

Output

Per ciascuna istanza, prima di leggere l'istanza successiva, scrivi su `stdout` il tuo output così strutturato:

+ la prima riga contiene un numero intero s, il numero di quei nodi $S \subseteq V$ che raggiungono un ciclo dispari in G.
+ la riga seguente contiene s numeri interi separati da spazio. Tali interi, tutti contenuti nell'intervallo $[0, n - 1]$, sono i nomi dei nodi contenuti in S.

Esempio

Input

```
2
10 11
0 1
2 4
3 4
2 3
0 2
1 3
5 6
6 7

```

```
7 8
8 9
5 9
7 7
0 1
4 6
2 3
0 2
4 5
5 6
1 3
```

Output

```
10
0 1 2 3 4 5 6 7 8 9
3
4 5 6
```

Spiegazione: il primo grafo contiene un ciclo dispari sui nodi $5, 6, 7, 8, 9$ ed un ciclo dispari sui nodi $2, 3, 4$ (un triangolo) ed ogni suo altro nodo raggiunge tale triangolo. Nel caso del secondo grafo abbiamo un triangolo sui nodi $4, 5, 6$ ma nessun altro nodo raggiunge cicli che siano dispari.

Subtask

Il tempo limite per istanza (ossia per ciascun testcase) è sempre di 1 secondo.

I testcase sono raggruppati nei seguenti subtask.

1. **[2 istanze] `esempi_testo`:** i due esempi del testo
2. **[25 istanze] `small`:** $N \leq 10$, $M \leq 20$
3. **[25 istanze] `medium`:** $N \leq 100$, $M \leq 500$
4. **[20 istanze] `big`:** $N \leq 5,000$, $M \leq 20,000$

In generale, quando si richiede la valutazione di un subtask vengono valutati anche i subtask che li precedono, ma si evita di avventurarsi in subtask successivi fuori dalla portata del tuo programma che potrebbe andare in crash o comportare tempi lunghi per ottenere la valutazione completa della sottomissione. Ad esempio, chiamando[1, 2] :

```
rtal -s <URL>  connect -x <token> -a size=medium
   odd_cycle_reach  -- python my_solution.py
```

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `small`, `medium`.

Il valore di default per l'argomento `size` è `big` che include tutti i testcase.

[1] `<URL>` server esame: `wss://ta.di.univr.it/esame`

[2] `<URL>` server esercitazioni e simula-prove: `wss://ta.di.univr.it/algo`