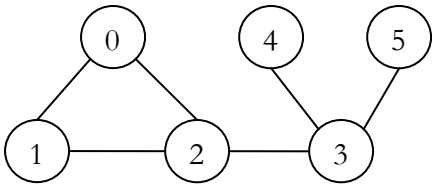
1300 – Odd Personality

Being an odd person, Jim always liked odd numbers. One day he was visiting his home town which consists of **n** places and **m** bidirectional roads. A road always connects two different places and there is at most one road between two places. The places are numbered from **0** to **n-1**.

Jim wants to find a tour which starts from a place **p** and each time it goes to a new **road** and finally at last step it returns back to **p**. As Jim likes odd numbers, he wants the length of the tour to be odd. And the length of a tour is defined by the number of roads used in the tour.



For the city map given above, **0 - 1 - 2 - 0** is such a tour, so, **0** is one of the results, since from **0**, a tour of odd length is found, similarly, **1 - 2 - 0 - 1** is also a valid tour. But **3 - 2 - 0 - 1 - 2 - 3** is not. Since the road **2 - 3** is used twice. Now given the city map, Jim wants to find the number of places where he can start his journey for such a tour. As you are the best programmer in town, he asks you for help. Jim can use a place more than once, but a road can be visited at most once in the tour.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers: **n (3 ≤ n ≤ 10000)** and **m (0 ≤ m ≤ 20000)**. Each of the next **m** lines contains two integers **u v (0 ≤ u, v < n, u ≠ v)** meaning that there is a bidirectional road between place **u** and **v**. The input follows the above constraints. And no road is reported more than once.

Output

For each case, print the case number and the total number places where Jim can start his journey.

Sample Input	Output for Sample Input
1 6 6 0 1 1 2 2 0 3 2 3 4 3 5	Case 1: 3

Note

Dataset is huge, user faster I/O methods.