Once upon a time, in a kingdom far, far away, there lived a king Byteasar II. There was nothing special neither about him, nor about his kingdom. As a mediocre ruler, he did nothing about his kingdom and preferred hunting and feasting over doing anything about his kingdom prosperity.

Luckily, his adviser, wise magician Bitlin, was working for the kingdom welfare daily and nightly. However, since there was no one to advise him, he completely forgot about one important thing: the roads. Over the years most of the two-way roads built by Byteasar II predecessors were forgotten and no longer traversable. Only a few `roads` can still be used.

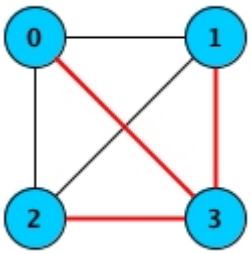
Bitlin wanted each pair of cities to be connected, but couldn't find a way to figure out which roads are missing. Desperate, he turned to his magic crystal, seeking help. The crystal showed him a programmer from the distant future: you! Now you're the only one who can save the kingdom. Given the existing `roads` and the number of `cities` in the kingdom, you should use the most modern technologies and find out what roads should be built again to make each pair of cities connected. Since the magic crystal is quite old and meticulous, it will only transfer the information that is sorted properly.

The roads to be built should be returned in an array sorted lexicographically, with each road stored as $[city_i, city_j]$, where $city_i < city_j$.

Example

For `cities = 4` and `roads = [[0, 1], [1, 2], [2, 0]]`,
the output should be
`roadsBuilding(cities, roads) = [[0, 3], [1, 3], [2, 3]]`.

See the image below: the existing roads are colored black, and the ones to be built are colored red.



Input/Output

- **[time limit] 4000ms (js)**
- **[input] integer cities**

  The number of cities in the kingdom.

  Constraints:
  $1 \le cities \le 100$.

- **[input] array.array.integer roads**

  Array of roads in the kingdom. Each road is given as a pair $[city_i, city_j]$, where $0 \le city_i, city_j < cities$ and $city_i \neq city_j$. It is guaranteed that no road is given twice.

  Constraints:
  $0 \le roads.length \le 5000$,

`roads[i].length = 2`,
`0 ≤ roads[i][j] < cities`.

- **[output] array.array.integer**

  A unique array of roads that should be built sorted as described above. There's no need to build loop roads, i.e. roads from a city to itself.