Assignment 6

Time-outs:

- Python: 5 seconds on Q1 and Q2, 10 seconds on Q3
- C++ (w/ g++ -O3 flag): 2 seconds on Q1 and Q2, 4 seconds on Q3

You may solve the programming problems in either Python or C++. To use Python, name the file containing your code `main.py`. To use C++, name the file `main.cpp`. You are encouraged to test your code locally.

At any time, you can choose to run the tests that will be used to grade your solution. To do so, navigate to the Gradescope assignment for that particular homework question and upload **only** `main.py` or `main.cpp`. Gradescope will tell you how many test cases passed, and what error occurred on the ones that failed.

When you run the tests on Gradescope, debugging output will be printed to tell you which test cases you passed, which ones you failed, and what your total score currently is for that problem. If a test case failed, the output will specify a failure mode (e.g. timeout, compiler error, wrong answer, etc.).

The first few test cases for each problem are visible test cases. This means that if you fail them, Gradescope will tell you the the expected output and the output your code actually gave. Note that these outputs may not be listed in order. The rest of the test cases for each problem are hidden test cases, which means you will not be shown the input or expected output.

All assignments MUST be manually submitted to Gradescope in order to complete submission! There is no auto-submit feature enabled for any Gradescope coursework. There will be no exemptions granted if you forget to manually submit your assignment. If time permits, you can contact the course staff and they can check on the Gradescope platform and confirm your assignment submission status.

1. Rapid Vertex Cover
(200 pts)

Description

The scientists from Oganesson Dynamics have found an alien computer than can input a graph and rapidly determine whether it contains a clique of K vertices. The possibilities are endless, since they should now be able to solve any NP-Complete problem!

For example: they found a set of wormholes (that form a graph!) with space stations at each vertex. To activate a wormhole, they must first power at least one neighboring space station. Will K activated space stations be enough to get the entire network up and running?

That's the vertex cover problem!

Given that you have a K-Clique solver, can you convert instances of the vertex cover problem to K-Clique such that it always returns the same YES/NO answer?

The conversion works as follows. If you convert your K to $N - K$ (that is, looking for the opposite of the vertex cover), you will have converted to the independent set problem. Then if you toggle all edges (that is, remove all edges that were present and add in all of the edges that were originally missing) to produce the complement graph, you will have converted from the independent set problem to the clique problem. Once these conversions are done, you then need to output the resulting problem instance.

Input Format

- The first line contains three values, N, M, K.
  - N is the number of vertices in the graph (with IDs 0 through $N - 1$)
  - M is the number of edges in the graph
  - K is the number of vertices being tested for
- The next M lines each describe an edge with two values indicating the IDs of the vertices being connected.

Constraints

- $1 \leq N \leq 200$
- $1 \leq M \leq 10{,}000$
- $0 \leq K \leq N$

Output Format

You must output the converted problem instance for the K-Clique problem that would have the same YES/NO answer as the original vertex cover problem. Otherwise, it should have the same format as the input.

NOTE: Each edge pair must have the lower ID first and be sorted in numerical order.

Example 0

Sample Input

```
5 6 3
0 1
0 2
0 4
1 4
2 3
2 4
```

Sample Output

```
5 4 2
0 3
1 2
1 3
3 4
```

Explanation

There are 5 vertices and 6 edges in the input graph; the question asks if 3 vertices can cover the graph.

To change this to independent set, we simply need to ask if the same graph has an independent set of size (5 - 3) = 2.

THEN to change this problem to *K*-Clique, we have to turn off the existing edges and turn on the ones that were missing in the original graph. There are (5 x 4 / 2) = 10 possible edges; 6 of them were in the original graph, but 4 were missing. We list only those four missing edges in our output.

Example 1

Sample Input

```
7 15 3
0 3
0 4
0 5
0 6
1 2
1 3
1 5
2 3
2 4
2 6
3 4
3 5
3 6
4 6
5 6
```

Sample Output

```
7 6 4
0 1
0 2
1 4
1 6
2 5
4 5
```

Example 2

Sample Input

```
10 20 4
0 3
0 5
0 7
0 8
0 9
1 4
1 6
1 7
2 3
3 4
3 5
3 6
3 9
4 6
4 8
4 9
5 7
6 7
7 9
8 9
```

Sample Output

```
10 25 6
0 1
0 2
0 4
0 6
1 2
1 3
1 5
1 8
1 9
2 4
2 5
2 6
2 7
2 8
2 9
3 7
3 8
4 5
4 7
5 6
5 8
5 9
6 8
6 9
7 8
```

2. Rapid Hamiltonian Cycle?

(200 pts)

Description

The second super-fast computer found by Oganesson takes a graph and outputs a series of vertices that describes a Hamiltonian Cycle. Or at least it often does; sometimes it makes a mistake. You must write a program that takes the input graph and a set of output cycles, and then verifies if each one is correct.

Input Format

- The first line contains N and M, the number of vertices and the number of edges in the input graph, respectively.
- The next M lines each describe an edge, providing the IDs of the two vertices it connects.
- The next line provides T, the number of test cycles.
- The final T lines each provides a test cycle with N vertex IDs in the proposed order of traversal.

Constraints

- $5 \leq N \leq 1000$
- $5 \leq M \leq 50{,}000$
- $1 \leq T \leq 10$
- $0 \leq v_i < N$ (for all v_i, as vertex IDs)

Output Format

T lines, each with the word "YES" or "NO", indicating whether the associated test cycle is valid.

Example 0

Sample Input

```
5 6
0 2
0 3
1 3
1 4
2 4
3 4
3
1 3 0 2 4
0 1 2 3 4
2 0 3 1 4
```

Sample Output

```
YES
NO
YES
```

Example 1

Sample Input

```
5 8
0 1
0 3
0 4
1 2
1 3
1 4
2 3
2 4
5
1 4 2 3 0
3 0 1 2 4
2 3 0 1 4
1 3 2 1 4
0 3 2 4 1
```

Sample Output

```
YES
NO
YES
NO
YES
```

Example 2

Sample Input

```
10 30
0 1
0 2
0 4
0 6
1 2
1 3
1 4
1 5
1 7
1 8
2 3
2 4
2 5
2 6
2 7
3 4
3 5
3 6
3 7
3 9
4 6
4 7
5 6
5 7
5 8
5 9
6 8
6 9
7 9
8 9
4
4 3 1 5 2 7 9 8 6 0
0 1 2 3 4 5 6 7 8 9
0 2 4 6 8 1 3 5 7 9
7 9 8 6 0 4 3 1 5 2
```

Sample Output

```
YES
NO
NO
YES
```

3. To Boldly Go…

(600 pts)

Description

Oganesson is now building huge star ships capable of multi-year long missions. These ships need to have a large crew where there are many skills represented. Can you help them determine who should go?

But wait! Haven't you already solved this problem? Technically, you have, but now the crews are much larger and your old algorithm is too slow. In fact, any classical algorithm is going to be too slow to find a *perfect* answer. As such, you just need to find the best answer that you can -- and print out the full answer.

For this problem, you will be graded on how *close* your answer comes to the correct answer. For every test case, you MUST return AN answer (i.e., timing out will result in no points for the given test case); the quality of your answer will determine how much credit you receive on the test case.

IMPORTANT NOTE: Full credit for the problem will be 400 points. Thus, 200 bonus points are available.

Input Format

- The first line has two values, N and K.
- N represents the number of potential crew members available (numbered 0 through N - 1), and K is the number of distinct skills that need to be included (numbered 0 through K - 1).
- The next N pairs of lines each provide information about a single person.
- The first line for person i indicates the number of skills that person has (P_i), and the second line has P_i values indicating the specific skill IDs.

Constraints

- $N == 1000$
- $500 \leq K \leq 1200$
- $8 \leq P_i \leq 200$

Output Format

You should output two lines. The first line indicates S, the size of the best solution you could find (the minimum number of people). The next line has S space-separated values indicating the specific IDs of the people to recruit onto the team.

Example 0

Sample Input

```
3 5
2
1 3
3
0 1 2
3
0 2 4
```

Sample Output

```
2
0 2
```

Explanation

There are three people to choose from in the example input, and five total skills. These people are then presented in order. Person 0 has 2 skills: 1 and 3. Person 1 has 3 skills: 0, 1, and 2. Person 2 also has 3 skills: 0, 2, and 4.

The output provided indicates that two people are chosen for the team: person 0 and person 2. Since these two people comprise the full set of skills, this answer would be deemed "correct" and would receive points. Given that it is also the minimal answer, it would receive the maximum number of points. If, instead, all three people were included in the answer, it would still be correct (since all of the skills would be covered), but it would not be minimal so it would receive fewer points.