Problema D – Oni está o Oni?

Este é um problema de **output-only**.
Ao contrário dos outros problemas em que deves fazer leitura de dados e escrita do output, neste problema deves apenas submeter um ficheiro de texto com extensão `.txt`

Um *Oni* é uma criatura muito poderosa, associada ao domínio das forças da natureza, mas não passa de uma lenda da mitologia japonesa... ou pelo menos era assim que se pensava até à semana passada! Começaram a surgir rumores de ter sido avistada a cria de um *Oni*, conhecida como Tobias, numa das cavernas de um bosque denso, e desde então os habitantes da vila mais próxima não descansam enquanto não a encontrarem! Segundo a lenda, se o *Oni* for encontrado e alimentado por um humano, as populações locais ficarão protegidas contra desastres durante 73 mil anos. Será que a população da vila é capaz de desenvolver uma estratégia para o encontrar?

O bosque pode ser representado por um grafo não dirigido com $\boldsymbol{N}$ vértices e $\boldsymbol{M}$ arestas. Inicialmente, o *Oni* encontra-se num destes vértices. Todos os dias, a população seleciona exatamente $\boldsymbol{K}$ caçadores para procurar o *Oni*, e cada um ficará encarregue de investigar um dos nós do bosque para descobrir se ele está ali ou não. Para facilitar a coordenação das operações, a população da vila decidiu que o número de caçadores selecionado diariamente, $\boldsymbol{K}$, é fixo, mas é permitido que haja vários caçadores a investigar o mesmo nó (redundantemente) no caso de não serem necessários tantos caçadores nesse dia. Assim, todos os dias a população seleciona um multiconjunto de $\boldsymbol{K}$ nós que irá visitar nesse dia. Sabe-se que, se o *Oni* não for encontrado num determinado dia, irá mover-se para um dos nós adjacentes no dia seguinte, e a busca continua.

Determina uma estratégia para visitar os nós que permita à população da vila encontrar garantidamente o *Oni*, independentemente dos movimentos deste. Definimos uma estratégia como uma sequência de $\boldsymbol{D}$ multiconjuntos de $\boldsymbol{K}$ nós cada, onde $\boldsymbol{D}$ é o número de dias de que os caçadores precisarão para encontrar o *Oni*. Enquanto que o número $\boldsymbol{K}$ de caçadores selecionados diariamente é escolhido pela população, podes escolher o número $\boldsymbol{D}$, desde que este não ultrapasse 5000.

Restrições

São garantidos os seguintes limites em todos os casos de teste que irão ser colocados ao programa:

$2 \leq \boldsymbol{N} \leq 100$	Número de nós do grafo
$1 \leq \boldsymbol{M} \leq \frac{\boldsymbol{N} \times (\boldsymbol{N}-1)}{2}$	Número de arestas do grafo
$1 \leq \boldsymbol{K} \leq \boldsymbol{N}$	Número de caçadores selecionados diariamente
$1 \leq \boldsymbol{D} \leq 5000$	Número máximo de dias

Formato de Input

Cada caso de teste começa com três inteiros separados por espaços: $\boldsymbol{N}$ (o número de nós), $\boldsymbol{M}$ (o número de arestas) e $\boldsymbol{K}$ (o número de caçadores a selecionar por dia).

Seguem-se $\boldsymbol{M}$ linhas, cada uma com dois inteiros u_i, v_i entre 1 e $\boldsymbol{N}$ separados por espaços, representando uma aresta. É garantido que o grafo é conexo, que **não há múltiplas arestas** entre o mesmo par de vértices, e que **todas as arestas ligam vértices diferentes**.

Casos de teste

Há 9 casos de teste, organizados em 9 grupos com restrições adicionais diferentes e cada um com uma pontuação fixa (sem pontuações parciais para cada um). Os casos de teste são os seguintes (arquivo zip contendo todos os casos):

Grupo	Número de Pontos	Restrições Adicionais
1 (`inp1.txt`)	5	$\boldsymbol{K} = 3$ e o grafo é uma árvore com 6 nós
2 (`inp2.txt`)	10	$\boldsymbol{K} = 1$ e o grafo é uma árvore com 6 nós
3 (`inp3.txt`)	5	$\boldsymbol{K} = 3$ e o grafo é uma linha com 100 nós
4 (`inp4.txt`)	15	$\boldsymbol{K} = 1$ e o grafo é uma linha com 100 nós
5 (`inp5.txt`)	10	$\boldsymbol{K} = 2$ e o grafo é uma ameba (ver em baixo) com 100 nós
6 (`inp6.txt`)	15	$\boldsymbol{K} = 1$ e o grafo é uma ameba (ver em baixo) com 100 nós
7 (`inp7.txt`)	10	$\boldsymbol{K} = 6$ e o grafo é uma grelha 5×5
8 (`inp8.txt`)	15	$\boldsymbol{K} = 6$ e o grafo é uma grelha 10×10
9 (`inp9.txt`)	15	$\boldsymbol{K} = 7$ e o grafo é uma árvore com 100 nós

Um **grafo ameba** é uma árvore que respeita a seguinte propriedade: todos os vértices estão a distância menor ou igual a 2 de um vértice do diâmetro da árvore. O diâmetro de uma árvore é um conjunto de vértices que forma o caminho mais longo entre quaisquer dois vértices da árvore.

Formato de Output

Deve ser submetido um **único** ficheiro de `txt` (o formato é importante, o teu ficheiro deve terminar em `.txt`), contendo a resposta para todos os casos de teste. Por cada caso de teste, pela ordem correta, o ficheiro deve conter $\boldsymbol{D} + 1$ linhas, onde $\boldsymbol{D}$ é o número de dias que a tua estratégia usa para esse caso de teste (podes utilizar diferentes valores de $\boldsymbol{D}$ para casos de teste diferentes).

O formato por caso de teste é o seguinte:

- uma linha com o inteiro $\boldsymbol{D}$,
- seguida de $\boldsymbol{D}$ linhas, cada uma com $\boldsymbol{K}$ inteiros $u_1, u_2, \ldots, u_{\boldsymbol{K}}$ entre 1 e $\boldsymbol{N}$: os nós que serão visitados nesse dia. Podem vir por qualquer ordem e não têm de ser distintos, mas têm de ser exatamente $\boldsymbol{K}$.

Observações importantes:

- Caso não tenhas solução para um dos casos de teste, imprime uma linha com $\boldsymbol{D} = 0$. Terás 0 pontos nesse caso de teste, mas não afetará a tua pontuação nos outros casos de teste.
- Se não respeitares o formato mencionado (por exemplo, imprimindo mais do que $\boldsymbol{D}$ linhas para um determinado caso de teste), é possível que a tua pontuação seja de 0 pontos para todos os casos de teste, porque o avaliador pode não ler as tuas soluções corretamente.
- Se algum dos outputs for inválido (por exemplo, seja porque não contém inteiros suficientes, ou porque os inteiros não estão entre 1 e $\boldsymbol{N}$), o resultado da submissão será **Wrong Answer** e terás 0 pontos nesse caso de teste.

Um possível exemplo de um ficheiro solução seria o seguinte (a sua pontuação é de 0 pontos):

```
0
3
1
2
3
0
0
0
0
0
0
1
1 1 50 50 100 100 100
```

Dicas para gerar o output

Visto que é necessário criar um ficheiro com extensão `.txt` para a tua submissão, pode ser útil usar o terminal para o gerar (isto não é obrigatório, podes criar o ficheiro como quiseres).

Se para correres o teu programa usares o comando `X` (por exemplo, para C++ isto seria algo como `./a.out`, depois de compilar com `g++ codigo.cpp`), podes usar o seguinte comando para ler o input de um ficheiro `inp.txt` e escrever para um ficheiro `out.txt`: `X < inp.txt > out.txt`.

Para facilitar a organização do ficheiro de submissão, é aconselhado gerar um `txt` por cada caso

de teste (por exemplo, para o caso de teste 2 ter um `out2.txt`).

Para posteriormente gerar um ficheiro `out.txt` para submeter podes, por exemplo, usar o seguinte comando (assumindo que tens um ficheiro para cada um dos 9 casos de teste):

```
cat out1.txt > out.txt; cat out2.txt >> out.txt; cat out3.txt >> out.txt; cat out4.txt
>> out.txt; cat out5.txt >> out.txt; cat out6.txt >> out.txt; cat out7.txt >> out.txt;
cat out8.txt >> out.txt; cat out9.txt >> out.txt
```