**Problema A: Amigos**

*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Quake trabalha em uma empresa na área de redes sociais. Quake deu uma tarefa a um de seus estagiários bem simples: dadas as relações de amigos das pessoas, contar quantos amigos cada pessoa tem.



Ao invés de trabalhar, seu estagiário ficou se distraindo com os video games que tinham no corredor da empresa, e agora perdeu o prazo de submissão da tarefa. Você é o único quem pode salvá-lo! Se não ajudá-lo no problema pode ser que a responsabilidade recaia sobre Quake, que é responsável por ele. Como amigo de Quake você resolveu quebrar esse galho para ele. Resolva esta tarefa e ajude Quake!

**Entrada**

Há varios testes. A primeira linha de cada caso de teste contém um número **N** (1 <= **N** <= 10^5), indicando quantas relações de amigos seguirão. Nas próximas **N** linhas haverá dois números inteiros **a** e **b** (1 <= **a,b** <= 2^31-1), indicando que a pessoa identificada pelo número **a** é amiga da pessoa **b**. Não haverá casos onde a pessoa é amiga dela mesma ou relações duplicadas/redundantes. Leia até o **EOF**.

**Saída**

Para cada caso de teste, imprima cada pessoa e o número de amigos dela separados por um espaço. Imprima de forma ordenada, ou seja, a pessoa com identificação menor vem primeiro! Após cada caso de teste imprima uma linha com o caracter **‘\*’**!!

|  |  |
| --- | --- |
| **Entrada de Teste**  2  0 3  5 3  3  1 2  1 3  1 4  1  1 10 | **Saída de Teste**  0 1  3 2  5 1  \*  1 3  2 1  3 1  4 1  \*  1 1  10 1  \* |



**Problema B: Carlota no Carnaval do Rio**

*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Carlota foi visitar o Carnaval do Rio. Ela não é muito boa com distâncias, por isto pediu a sua ajuda. Você tem o mapa da cidade com pontos turísticos e calçadas os conectando entre si em ambas direções. (Se uma calçada conecta o ponto 1 ao ponto 2, Carlota pode caminhar do ponto 1 para o ponto 2 e também do ponto 2 para o ponto 1).

Cada ponto é identificado por um número único de 1 a **N**. Carlota está atualmente no ponto 1 e deseja chegar ao ponto **N**. É possível que não se possa chegar em alguns pontos, isto significa que não se pode ir a pé para lá, ou seja, utilizando apenas calçadas. A pergunta é: Carlota pode chegar no destino apenas andando?



**Entrada**

A primeira linha de cada caso de teste contém os números **N** (2 <= **N** <= 1000) e **M** (1 <= **M** <= 10^5), indicando o número de pontos turísticos e número de calçadas respectivamente. Cada uma das **M** linhas seguintes representa uma calçada. Há dois números em cada linha, que são os identificadores de dois pontos conectados pela calçada. Leia até o **EOF**.

**Saída**

Imprima **S** caso for possível chegar ao destino ou **N** caso contrário.

|  |  |
| --- | --- |
| **Entrada de Teste**  3 2  1 2  2 3 4 4  1 2  2 1  3 1  4 1  3 3  1 1  1 1  1 1 | **Saída de Teste**  S  S  N |

**Problema C: Barril de Rum**

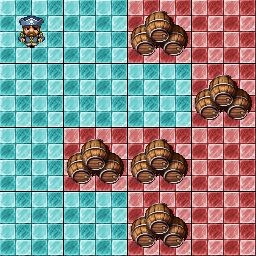
*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Os piratas do Caribe comandados por Diego pretendem invadir o armazém contendo barris de rum. O armazém pode ser representado por uma matriz de **N** linhas e **M** colunas. Cada posição da matriz pode conter ou não um barril de rum.

Um pirata pode andar apenas horizontalmente ou verticalmente no armazém, ou seja, dar um passo para cima, baixo, direita ou esquerda. O pirata só pode andar se e somente se não houver um barril de rum para a posição onde ele for. Ele não pode sair da matriz que representa o armazém

A entrada do armazém fica no canto superior esquerdo. Dado a configuração do armazém, qual a área máxima que um pirata pode cobrir? Ou seja, conte o número de posições que um pirata pode atingir!



A figura acima ilustra o último caso de exemplo. A matriz é de 4 linhas por 4 colunas, o pirata só consegue chegar nas áreas azuis, que totalizam 8 de área.

**Entrada**

A primeira linha de cada caso de teste contém o número **N** (1 <= **N** <= 1000) e **M** (1 <= **M** <= 1000), o número de linhas e de colunas. Nas próximas **N** linhas há **M** strings representando a configuração do armazém, se houver um ‘**.**’ então não há nenhum barril, se houver um ‘**o**’ então há um barril. É garantido que a entrada do armazém não contém um barril. Leia até o **EOF**.

**Saída**

Para cada caso imprima a área coberta pelos piratas.

|  |  |
| --- | --- |
| **Entrada de Teste**  1 1  .  3 1  .  o  .  3 3  .o.  …  o.o  4 4  ..o.  ...o  .oo.  ..o. | **Saída de Teste**  1  1  6  8 |



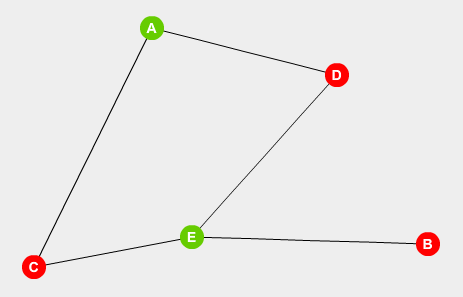
**Problema D: Dromedários do Deserto**

*TL: 2 segundos (~50 casos)*

*ML: 64 MB*

Fábio é um Sheik do deserto. Ele monopoliza o transporte de dromedários por lá. Há algumas cidades no deserto conectadas por estradas de areia. Há sempre um caminho entre as cidades e entre cada par de cidades há no máximo uma estrada de duas mãos.

Fábio é muito excêntrico e em cada cidade ele tem um castelo. Existe dois tipos de castelo, e ele quer que entre cidades adjacentes (em cidades conectadas por uma estrada) os castelos sejam de tipos diferentes. Dado a configuração das cidades e dos castelos isto é possível?



O último caso de teste está representado acima. Neste caso é possível obter a configuração desejada.

**Entrada**

A primeira linha de cada caso de teste contém os números **N** (1 <= **N** <= 10^5) e **M** (1 <= **M** <= 10^6), indicando o número de cidades e o número de estradas de areia. Cada uma das **M** linhas seguintes representa uma estrada. Há dois números em cada linha, que são os identificadores de duas cidades conectadas pela estrada, o identificador das cidades irá de 1 até **N**. Leia até o **EOF**.

**Saída**

Imprima **S** se existe uma configuração ou **N** caso contrário.

|  |  |
| --- | --- |
| **Entrada de Teste**  3 2  1 2  2 3  3 3  1 2  3 2  3 1  5 5  1 2  1 3  2 4  3 4  4 5 | **Saída de Teste**  S  N  S |



**Problema E: Labirinto do Minotauro**

*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Shi está preso no labirinto do Minotauro. O labirinto pode ser representado por uma matriz de **N** linhas e **M** colunas. Em cada posição do labirinto há um identificador que indica qual ação que ele pode tomar:

* **>** : Shi pode dar um passo para a direita.
* **<** : Shi pode dar um passo para a esquerda.
* **^** : Shi pode dar um passo para cima.
* **v** : Shi pode dar um passo para baixo.
* **#** : Shi pode dar um passo em qualquer direção.

Um passo é válido se e somente se após andar Shi ainda está dentro dos limites do labirinto! Shi jamais pode sair dos limites do labirito! Ele começa no canto superior esquerdo.

Chino, Sanches, Rafael, André, Lucas, Denis e Bianca estão a caminho para resgatá-lo do labirinto. O único meio de sobreviver até que seus amigos cheguem é se Shi conseguir sempre dar um passo. Isto é, o minotauro está sempre atrás dele, e caso ele fique encurralado irá alcançá-lo!

Por simplicidade o Minotauro não ocupa nenhuma posição na matriz. A cada segundo Shi deve dar um passo válido, ou seja, deve obedecer o tipo da ação da posição em que se encontra. Se o próximo passo levará Shi para ‘fora’ do labirinto então o passo é inválido.

Existe alguma sequência infinita de passos que Shi pode seguir? Ou seja, existe algum meio de ele sempre dar um passo a cada segundo se ele seguir alguma estratégia válida?

**Entrada**

A primeira linha de cada caso de teste contém o número **N** (1 <= **N** <= 1000) e **M** (1 <= **M** <= 1000), o número de linhas e de colunas. Nas próximas **N** linhas há **M** strings representando a configuração do labirinto. Leia até o **EOF**.

**Saída**

Imprima **S** caso Shi consiga sobreviver ou **N** caso contrário.

|  |  |
| --- | --- |
| **Entrada de Teste**  1 1  #  2 2  >v  ^<  2 2  ##  ##  4 3  #>>  v<#  v^#  >>^  4 3  #>>  v^<  v^^  >>^ | **Saída de Teste**  N  S  S  S  N |

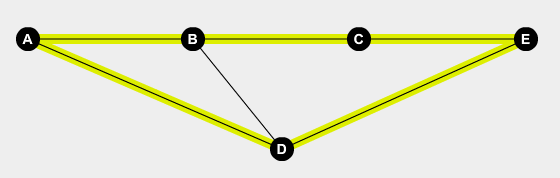
**Problema F: Mario Kart**

*TL: 1 segundo (~50 casos)*

*ML: 64 MB*

Filipe está jogando Mario Kart com Anderson. Eles querem jogar de modo cooperativo. Para isso eles decidem nunca usar os mesmos caminhos de um ponto até o outro da corrida, ou seja, eles querem usar caminhos distintos para evitar possíveis conflitos na corrida. Todos os pontos podem ser alcançados do ponto inicial.

Na corrida temos alguns pontos, e para mover de um ponto até o outro podemos ter vários caminhos. Seu objetivo é descobrir se eles podem chegar do ponto 1 até o ponto **N** sem usar os mesmos caminhos (arestas).



Os pontos estão representados pelos círculos pretos com uma letra dentro. Os caminhos são as linhas que ligam estes pontos. Este é o último caso de exemplo. Um dos dois pode ir por *A->B->C->E* e o outro por *A->D->E*, portanto eles não usaram os mesmos caminhos e a resposta é S. Eles podem passar pelos mesmos pontos (vértices), mas não devem usar os mesmos caminhos (arestas)!

**Entrada**

A primeira linha de cada caso de teste contém os números **N** (1 <= **N** <= 10^3) e **M** (1 <= **M** <= 3\*10^3), indicando o número de pontos e o número de caminhos. Cada uma das **M** linhas seguintes representa um caminho. Há dois números em cada linha, que são os identificadores de dois pontos conectados pelo caminho, o identificador dos pontos irá de 1 até **N**. A entrada poderá conter mais de um caminho conectando dois pontos, assim como um caminho conectando um ponto a ele mesmo. Leia até o **EOF**.

**Saída**

Imprima **S** caso possível ou **N** caso contrário.

|  |  |
| --- | --- |
| **Entrada de Teste**  3 3  1 2  2 3  1 3  3 3  1 2  2 3  2 3  5 6  1 2  2 3  3 5  1 4  2 4  4 5  2 2  2 1  1 2 | **Saída de Teste**  S  N  S  S |