Задача С2. Школа

Започна новата учебна година и в Школата по информатика в град Ш. отново се събраха много нови състезатели. За да подсигури редовното уведомяване на учениците за сбирките на групата, ръководителката на Школата направила малко проучване и установила, че новите ученици са N ($5 \le N \le 500$), номерирани с числата от 1 до N, по реда, в който са записани в дневника на Школата. M двойки от тях се познават и когато единият от двойката научи за поредната сбирка, може да уведоми другия за сбирката. За да си улесни работата, ръководителката би искала да избере колкото може по-малко ученици, които да уведоми лично, а те от своя страна да разпространят информацията до всички, като използват познанствата помежду си. Разбира се, че всеки уведомен за сбирката може да се свърже с всички свои познати и да ги уведоми, ако някой друг не го е направил до момента.

Напишете програма **SCHOOL**, която по зададени познанствата на учениците намира и извежда на стандартния изход минималния брой ученици, които трябва да бъдат уведомени от ръководителката така, че информацията да може да стигне до всички останали.

На първия ред на стандартния вход ще бъдат зададени числата N и M**,** разделени с един интервал. На всеки един от следващите M реда, разделени с интервал, са зададени два номера на ученици, които се познават.

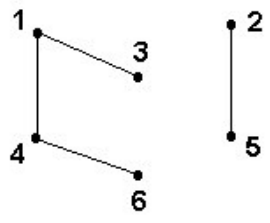
Пример.
Вход:
```
6 4
1 3
1 4
2 5
4 6
```

Изход:
```
2
```

Решение:

За решението на тази задача ще използваме техника от теорията на графите. Неориентираният ***граф*** $G(V,E)$ е съставен от множество от ***върхове*** $V=\{1,2,\ldots,N\}$ и множество от ***ребра*** E такова, че всяко ребро от E свързва два върха u и w от E – означаваме го с (u,w). Редицата от върхове $u_1,u_2,\ldots,u_k$ наричеме ***път*** от върха u_1 до върха u_k, ако всеки два съседни в редицата върха u_i и u_{i+1} са свързани с ребро. Когато $u_1= u_k$ тогава пътя наричаме ***цикъл***. Графът е ***свързан***, ако в него има път от всеки връх до всички останали. Графът е ***дърво***, ако е свързан и няма цикли. Дървото $D(V,E')$ такова, че $E' \subseteq E$ наричаме ***покриващо дърво*** на G. В сила е следното важно свойство: графът G е свързан тогава и само тогава, когато има покриващо дърво.

Ако графът не е свързан, тогава той се разпада на отделни свързани ***подграфи***, всеки от които наричаме ***свързана компонента***. На Фиг. 1 е показан граф с 6 върха и 4 ребра, който не е свързан и има 2 свързани компоненти (върховете 1, 3, 4 е 6 са в едната, а върховете 2 и 5 в другата).



Фиг 1.

	0	1	2	3	…	
1	2	3	4			
2	1	5				
3	1	1				
4	2	1	6			
5	1	2				
6	1	4				

Фиг. 2.

Да представим даденото в задачата като неориентиран граф. На всеки от учениците да съпоставим връх на графа и да свържем с ребро два върха, ако съответните им ученици се познават и, ако единият от тях бъде уведомен за сбирка на Школата, може да уведоми другия. Нека ученикът, съответен на върха с номер 1 е уведомен за сбирката, той може да уведоми всички ученици с които се познава. Всеки от тях може да уведоми всеки от познатите си, които още не са уведомени. В резултат ще бъдат уведомени всички ученици, за върховете на които в графа има път до (от) върха с номер 1, т.е. тези които са в една и съща свързана компонента на графа. Очевидно, за всяка свързана компонента ще е необходимо и достатъчно да бъде уведомен точно един от учениците, т.е търсеният в задача минимален брой ученици, които трябва да бъдат уведомени е равен на броя на свързаните компоненти на графа.

За намиране на броя на свързаните компоненти ще приложим сравнително универсалната техника "обхождане в ширина". С малка модификация, обаче, алгоритъмът може да се приложи и за решаване на задачата за определяне на самите компоненти (опитайте се да го направите сами).

Много важно за успешната реализация е начинът по който ще представим графа. Ще използваме представяне, което наричаме "списъци на съседите". Нека `g` е двумерен масив с толкова реда, колкото са върховете на графа и стълбове с 1 повече от върховете на графа (в реализацията използваме редовете с номера 1,2,..., *N* и стълбовете с номера 0,1,2,..., *N*). Списъкът на съседите на върха *u* съхраняваме в реда с номер *u*. Първият, вторият и т.н до *k*-тия съсед на *u* записваме в първия, втория и т.н. до *k*-тия елемент на реда, а в нулевия елемент поставяме броя *k* на съседите на *u*. На Фиг.2 е показано представянето на графа от Фиг.1 със списъци на съседите.

При "обхождане в ширина" използваме опашка `q`, като в променливите `qs` и `qe` се намират началото и краят на опашката. Започваме с произволен начален връх (например 1). Поставяме го в опашката, обявяваме го за обходен (като поставим 1 в съответния елемент на масива `used`), преброявяме първата свързана компонента (променливата `cnt`) и обхождаме тази компонента в ширина. За целта, докато в опашката има елементи, изваждаме елемент от началото на опашката и добавяме в края на опашката всички негови необходени съседи, обявявайки всеки един такъв съсед за обходен. Ако в края на тази стъпка няма необходени върхове – алгоритъмът завършва. Ако имаме необходен връх – избираме този връх за начален, преброяваме още една компонента и повтаряме описаните вече стъпки. На Фиг. 3 е показан програмен фрагмент с основните стъпки на алгоритъма.

```
cnt=0;
for(i=1;i<=N;i++)  {used[i]=0;g[i][0]=0;}
for(k=1;k<=N;k++)
{
   if(used[k]==1) continue;
   qs=qe=0;q[qs]=k;used[k]=1;cnt++;
   while(qs<=qe)
   {
      x=q[qs++];
      for(i=1;i<=g[x][0];i++)
      {
         y=g[x][i];
       if(used[y]==0)
       {used[y]=1;q[++qe]=y;}
    }
   }
 }
cout<<cnt<<endl;
```

Фиг. 3