Перевірка на неорієнтовність

За заданою квадратною матрицею **n** × **n** з нулів та одиниць визначити, чи може вона бути матрицею суміжності простого неорієнтовного графа.

Вхідні дані

У першому рядку задано число **n** (**1** ≤ **n** ≤ **100**). Потім йдуть **n** рядків по **n** елементів у кожному - опис матриці суміжності.

Вихідні дані

Вивести **YES**, якщо граф неорієтований, та **NO** у протилежному випадку.

Вхідні дані

3

0 1 1

1 0 1

1 1 0

Вихідні дані

YES

Ідея розв'язку:

Квадратна матриця не може бути матрицею суміжності неорієнтованого графа, якщо елементи ***A[i,j]<>A[j,i]*** або при ***i=j A[i,j]<>0***.

Лістінг програми:

```
var A:array[1..100,1..100] of integer;
     i,j,n:integer;
begin
readln(n);
for i:=1 to n do
  for j:=1 to n do
   read(A[i,j]);
for i:=1 to n do
  for j:=1 to n do
    begin
```

```
    if (A[i,j]<>A[j,i]) then
                        begin
                        write('NO');
                        exit;
                        end;
    if (i=j) and (A[i,j]<>0) then
                            begin
                            write('NO');
                            exit;
                            end;
  end;
write('YES');
end.
```

Від матриці суміжності до списків суміжності

Простий орієнтовний граф задано матрицей суміжності. Виведіть його подання у вигляді списків суміжності.

Вхідні дані

У першому рядку знаходиться кількість вершин графа **n** (**1** ≤ **n**≤ **100**). У другому рядку і далі - матриця суміжності. Гарантується, що граф не містить петель.

Вихідні дані

Виведіть **n** рядків - списки суміжності графа. В **i**-му рядку спочатку виведіть кількість ребер, які виходять з **i**-ої вершини, а потім - номери вершин, у які ці ребра входять, впорядковані за зростанням.

Вхідні дані

```
5
0 0 1 0 0
1 0 1 0 0
0 0 0 0 1
1 1 0 0 0
```

1 1 0 0 0

Вихідні дані

1 3

2 1 3

1 5

2 1 2

2 1 2

Ідея розв’язку:

Наявність ребра між вершинами ***i*** і ***j*** в орієнтованому графі визначається ***A[i,j]=1*** в матриці суміжності. Для кожного рядка матриці суміжності знаходимо кількість елементів рівних 1 та виводимо номери стовпців цих елементів.

Лістінг програми:

```
var A:array[1..100,1..100] of integer;
   n,i,j,s:integer;
begin
readln(n);
for i:=1 to n do
 for j:=1 to n do
 read(A[i,j]);
for i:=1 to n do
 begin
 s:=0;
 for j:=1 to n do
  if a[i,j]=1 then s:=s+1;
 write(s,' ');
 for j:=1 to n do
  if A[i,j]=1 then write(j,' ');
 writeln;
 end;
```

end.

Півстепені вершин

Орієнтовний граф задано матрицею суміжності. Знайдіть півстепені заходу та півстепені виходу усіх вершин графа (тобто кількості ребер, які входять у неї, та виходять з неї відповідно для кожної вершини).

Вхідні дані

N - число вершин у графі ($1 \leq N \leq 100$), потім матриця суміжності: N рядків по N чисел, кожне з яких дорівнює 0 або 1.

Вихідні дані

Виведіть N пар чисел: для кожної вершини спочатку півстепінь заходу і потім півстепінь виходу.

Вхідні дані

4

0 1 0 1

1 0 1 1

0 1 0 0

1 1 1 1

Вихідні дані

2 2

3 3

2 1

3 4

Ідея розв'язку:

В матриці суміжності знаходимо в кожному рядку кількість елементів ***s1*** рівних 1 – півстепінь виходу та в кожному стовпцю кількість елементів ***s2*** рівних 1 – півстепінь заходу.

Лістінг програми:

```
var A:array[1..100,1..100] of integer;
   i,j,n,s1,s2:integer;
begin
```

```
readln(n);
for i:=1 to n do
  for j:=1 to n do
  read(A[i,j]);
for i:=1 to n do
  begin
  s1:=0;
  s2:=0;
  for j:=1 to n do
    begin
    if A[i,j]=1 then s1:=s1+1;
    if A[j,i]=1 then s2:=s2+1;
    end;
  writeln(s2,'  ',s1);
  end;
end.
```

Витоки та стоки

Вершина орієттовного графа називається *витоком*, якщо в неї не входить жодне ребро, і *стоком*, якщо з неї не виходить жодного ребра.

Орієнтовний граф задано матрицею суміжності. Знайдіть усі його вершины-витоки та усі вершини-стоки.

Вхідні дані

Перший рядок містить кількість вершин у графі **n** (**1** ≤ **n** ≤ **100**), далі йде матриця суміжності - **n** рядків по **n** чисел, кожне з яких дорівнює **0**або **1**.

Вихідні дані

У першому рядку виведіть кількість витоків у графі, потім номери вершин, які є витоками у порядку зростання. У другому рядку виведіть інформацію про стоки у такому ж форматі.

Вхідні дані #1

```
5
```

0 0 0 0 0

0 0 0 0 1

1 1 0 0 0

0 0 0 0 0

0 0 0 0 0

Вихідні дані #1

2 3 4

3 1 4 5

Ідея розв’язку:

В матриці суміжності знаходимо кількість стовпців, всі значення в яких дорівнюють 0 та номери цих стовпців. Цим ми знайдемо кількість витоків у графі та номери вершин, які є витоками.

Аналогічно знаходимо кількість рядків, всі значення в яких дорівнюють 0 та номери цих рядків – інформація про стоки.

Лістінг програми:

```
var A:array[1..100,1..100] of integer;
   V,S:array[1..100] of integer;
   i,j,n,s1,s2,k1,k2:integer;
begin
readln(n);
for i:=1 to n do
  for j:=1 to n do
  read(A[i,j]);
k1:=0;
k2:=0;
for i:=1 to n do
  begin
  s1:=0;
  s2:=0;
  for j:=1 to n do
```

```
    begin
    if A[i,j]=1 then s1:=s1+1;
    if A[j,i]=1 then s2:=s2+1;
    end;
  if s1=0 then begin
              k1:=k1+1;
              S[k1]:=i;
              end;
  if s2=0 then begin
              k2:=k2+1;
              V[k2]:=i;
              end;
  end;
write(k2,'  ');
for i:=1 to k2 do
  write(V[i],'  ');
writeln;
write(k1,'  ');
for i:=1 to k1 do
  write(S[i],'  ');
end.
```

Повний граф

Неорієнтовний граф називається *повним*, якщо довільна пара його різних вершин з'єднана хоча б одним ребром. Для заданого списком ребер графа перевірте, чи є він повним.

Вхідні дані

Програмі на вхід подаються числа **N** та **M**, де **N** - кількість вершин (**1** ≤ **N**≤ **100**) і **M** - кількість ребер (**1**≤ **M**≤ **10000**), а потім **M** пар чисел - ребра графа.

Вихідні дані

Виведіть **YES**, якщо граф є повним, і **NO** у протилежному випадку.

Вхідні дані

3 3

1 2

1 3

2 3

Вихідні дані

YES

Ідея розв'язку:

Перетворюємо список ребер на матрицю суміжності.

В матриці суміжності визначаємо для кожного рядка кількість елементів ***s*** рівних 1. Якщо ***s<n-1***, то граф не повний.

Лістінг програми:

```
var A:array[1..100,1..100] of integer;
   i,j,n,s,m,k:longint;
begin
readln(n,m);
for k:=1 to m do
 begin
 read(i,j);
 A[i,j]:=1;
 A[j,i]:=1;
 end;
for i:=1 to n do
 begin
 s:=0;
 for j:=1 to n do
  if (A[i,j]=1) and (i<>j) then s:=s+1;
 if s<n-1 then begin
            write('NO');
```

```
            exit;
            end;
  end;
write('YES');
end.
```

Кількість ребер у неорієнтовному графі

Простий неорієнтовний граф задано матрицею суміжності.

Знайти кількість ребер у графі.

Вхідні дані

У першому рядку вхідного файлу задано число **N** (**1** ≤ **N**≤ **100**). Потім йде **N** рядків по **N** елементів у кожному - опис матриці суміжності.

Вихідні дані

У вихідний файл виведіть єдине число - кількість ребер у графі.

Вхідні дані

3

0 1 0

1 0 1

0 1 0

Вихідні дані

2

Ідея розв'язку:

Матриця суміжності неорієнтованого графа є симетричною відносно головної діагоналі, тобто кожне ребро задається два рази. Підраховуємо в матриці суміжності кількість елементів ***k***, які дорівнюють 1 та виводимо половину значення змінної ***k***.

Лістінг програми:

```
var a,i,j,k,n:integer;
begin
readln(n);
k:=0;
```

```
for i:=1 to n do
  for j:=1 to n do
  begin
  read(a);
  if a=1 then k:=k+1;
  end;
write(k div 2);
end.
```

Петлі

За заданою матрицею суміжності неорієнтовного графа визначте, чи містить він петлі.

Вхідні дані

У першому рядку вхідного файлу задано число **N** (**1** ≤ **N**≤ **100**). Потім йде **N** рядків по **N** елементів у кожному - опис матриці суміжності.

Вихідні дані

У вихідний файл вивести "**YES**", якщо граф містить петлі, і "**NO**" у протилежному випадку.

Вхідні дані #1

```
3
0 1 1
1 0 1
1 1 0
```

Вихідні дані #1

```
NO
```

Вхідні дані #2

```
3
0 1 0
1 1 1
0 1 0
```

Вихідні дані #2

YES

Ідея розв’язку:

Якщо на головній діагоналі матриці суміжності є елемент, який дорівнює 1, то граф має петлю.

Лістінг програми:

```
var a, i,j,n:integer;
begin
readln(n);
for i:=1 to n do
  for j:=1 to n do
  begin
  read(a);
  if (i=j) and (a=1) then
                    begin
                    write('YES');
                    exit;
                    end;
  end;
write('NO');
end.
```

Від матриці суміжності до списку ребер

Простий неорієнтовний граф задано матрицею суміжності.

Виведіть його подання у вигляді списку ребер.

Вхідні дані

У першому рядку вхідного файлу задано число **N** (**1** ≤ **N**≤ **100**). Потім йде **N** рядків по **N** елементів у кожному - опис матриці суміжності.

Вихідні дані

У вихідний файл виведіть список ребер, упорядкований спочатку по першій вершині і парі вершин, яка описує ребро, а потім по другій.

Вхідні дані

3

0 1 1

1 0 1

1 1 0

Вихідні дані

1 2

1 3

2 3

Ідея розв’язку:

В матриці суміжності простого неорієнтованого графа знаходимо елементи над головною діагоналлю значення яких дорівнює 1 і виводимо індекси цих елементів що відповідають парам вершин, які описують ребра цього графа.

Лістінг програми:

```
var A:array[1..100,1..100] of  integer;
      i,j,n:integer;
begin
readln(n);
for i:=1 to n do
   for j:=1 to n do
   read(A[i,j]);
for i:=1 to n do
  for j:=1 to n do
    if (j>i) and (A[i,j]=1) then  writeln(i,'  ',j);
end.
```

Від списку ребер до матриці суміжності

Простий неорієнтовний граф задано списком ребер. Виведіть його подання у вигляді матриці суміжності.

Вхідні дані

У першому рядку задано два цілих числа **n** (**1** ≤ **n** ≤ **100**) - число вершин та **m** (**1** ≤ **m** ≤ **n** · (**n - 1**) / **2**) - число ребер. Далі у **m** рядках міститься **m** пар чисел, кожна з яких описує одне ребро графа.

Вихідні дані

Виведіть матрицю суміжності графа.

Вхідні дані #1

3 3

1 2

2 3

1 3

Вихідні дані #1

0 1 1

1 0 1

1 1 0

Ідея розв'язку:

Зчитуємо пару чисел ***i*** та ***j***. В матриці суміжності задаємо значення елементам ***A[i,j]***= ***A[j,i]***=1.

Лістінг програми:

```
var A:array [1..100,1..100] of integer;
   i,j,k,n,m:longint;
begin
readln(n,m);
for k:=1 to m do
 begin
 readln(i,j);
 A[i,j]:=1;
 A[j,i]:=1;
 end;
for i:=1 to n do
 begin
```

```
  for j:=1 to n do
  write(A[i,j],'  ');
  writeln;
  end;
end.
```

Степені вершин

Простий неорієнтовний граф задано матрицею суміжності. Знайдіть степені усіх вершин графа.

Вхідні дані

У першому рядку задано число **n** (**1** ≤ **n** ≤ **100**). Потім йдуть **n** рядків по **n** елементів у кожному - опис матриці суміжності.

Вихідні дані

Виведіть **n** чисел - степені усіх вершин.

Вхідні дані #1

3

0 1 0

1 0 1

0 1 0

Вихідні дані #1

1

2

1

Ідея розв’язку:

В матриці суміжності знаходимо в кожному рядку кількість елементів, значення яких дорівнює 1.

Лістінг програми:

```
var a,i,j,k,n:integer;
begin
readln(n);
for i:=1 to n do
```

```
  begin
  k:=0;
  for j:=1 to n do
    begin
    read(a);
    if a=1 then k:=k+1;
    end;
  writeln(k);
  end;
end.
```

Кількість ребер у орієнтовному графі

Орієнтовний граф задано матрицею суміжності.

Знайдіть кількість ребер у графі.

Вхідні дані

У першому рядку вхідного файлу задано число **N (1 ≤ N≤ 100)**. Потім йдуть **N** рядків по **N** елементів у кожному - опис матриці суміжності.

Вихідні дані

У вихідній файл виведіть єдине число - кількість ребер у графі.

Вхідні дані

3

0 1 1

1 0 1

0 1 1

Вихідні дані

6

Ідея розв’язку:

В матриці суміжності знаходимо кількість елементів, значення яких дорівнює 1.

Лістінг програми:

```
var a,i,j,k,n:longint;
```

```
begin
readln(n);
k:=0;
for i:=1 to n do
  for j:=1 to n do
  begin
  read(a);
  if a=1 then k:=k+1;
  end;
write(k);
end.
```

Від матриці суміжності до списку ребер - 2

Орієнтовний граф задано матрицею суміжності.

Виведіть його подання у вигляді списку ребер.

Вхідні дані

У першому рядку вхідного файлу задано число **N** (**1** ≤ **N**≤ **100**). Потім йдуть **N** рядків по **N** елементів у кожному - опис матриці суміжності.

Вихідні дані

У вихідний файл виведіть список ребер, впорядкований спочатку за першою вершиною у парі вершин, яка описує ребро, а потім по другій.

Вхідні дані

```
3
0 1 0
0 0 1
1 1 0
```

Вихідні дані

```
1 2
2 3
3 1
3 2
```

Ідея розв’язку:

Знаходимо елемент матриці суміжності, значення якого дорівнює 1. Виводимо номер рядка та номер стовпчика, в якому він знаходиться.

Лістінг програми:

```
var a,i,j,n:integer;
begin
readln(n);
for i:=1 to n do
  for j:=1 to n do
  begin
  read(a);
  if  a=1 then
        begin
        writeln(i,'  ',j);
        end;
  end;
end.
```

Від списку ребер до матриці суміжності - 2

Простий орієнтовний граф задано списком ребер.

Виведіть його подання у вигляді матриці суміжності.

Вхідні дані

У першому рядку вхідного файлу задано два цілих числа **N** (**1** ≤ **N**≤ **100**) - число вершин та **M** (**1**≤ **M**≤ **N**·(**N-1**)/**2**) - число ребер. Далі у **M**рядках містяться **M** пар чисел, кожна з яких описує одне ребро графа.

Вихідні дані

У вихідний файл виведіть матрицю суміжності графа.

Вхідні дані

3 4

1 2

2 3

3 1

3 2

Вихідні дані

0 1 0

0 0 1

1 1 0

Ідея розв’язку:

Зчитуємо пару чисел ***i*** та ***j***. В матриці суміжності задаємо значення елементу ***A[i,j]***=1.

Лістінг програми:

```
var A:array [1..100,1..100] of integer;
   i,j,k,n,m:longint;
begin
readln(n,m);
for k:=1 to m do
  begin
  readln(i,j);
  A[i,j]:=1;
  end;
for i:=1 to n do
  begin
  for j:=1 to n do
  write(A[i,j],'  ');
  writeln;
  end;
end.
```