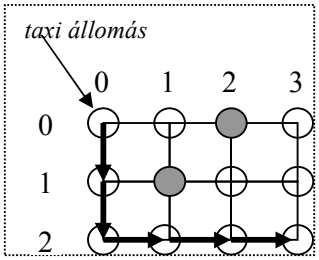
1. Feladat - Útvonalak

Csajádaröcsöge városát számítástechnikusok alapították, ezért úthálózata *m*x*n*-es méretű téglalap alakú rácsot alkot. A rácspontokat észak-dél, illetve nyugat-kelet irányokban nullától kezdődően sorszámozzuk, egy rácspontot a sorindex és az oszlopindex határoz meg. Egy taxitársaság állomása az észak-nyugati sarokban, a (0,0) pontban van. Az egyik taxi rendelést kapott a délkeleti város-sarokból, az (*m*-1,*n*-1) koordinátájú pontból.

A taxisofőr tudja, hogy bizonyos útkereszteződések építés alatt vannak, amelyeken nem lehet áthaladni, és a legrövidebb utat szeretné választani. Ezért érdekelné, hány különböző legrövidebb út létezik a taxiállomás és a délkeleti város-sarok között?

A mellékelt példa egy 3x4-es téglalap alakú úthálózatot szemléltet (*m*=3, *n*=4), ahol az (1,1) és a (0,2) koordinátájú útkereszteződések le vannak zárva. A délkeleti város-sarok koordinátája (2,3), és egyetlen minimális hosszúságú út létezik (a vastagított nyilak jelzik az útvonalat).



Követelmény

Határozzuk meg a taxiállomás és a délkeleti város-sarok közötti minimális hosszúságú utak számát.

Bemeneti állomány

Az **ut.in** bemeneti állomány egy úthálózat-sorozatot tartalmaz. Minden hálózat megadása a vízszintes (Nyugat-Kelet), illetve függőleges (Észak-Dél) utak számának megadásával kezdődik, m és n két természetes szám, ami szóközzel van elválasztva. Az ezt követő sorok egész számpárokat tartalmaznak szóközzel elválasztva, amelyek az építés alatt levő útkereszteződéseket jelzik. Az első szám a sorindex, a második az oszlopindex. Sem a taxiállomás, sem pedig az ezzel szemközti sarok nem szerepel ebben a listában. Egy úthálózat adatainak végét a 0 0 számpár jelzi. A feladat bemenetét szintén a 0 0 számpár zárja.

Kimeneti állomány

Az **ut.out** kimeneti állomány minden sora a bemenetben szereplő minden egyes úthálózatra egy-egy természetes számot fog tartalmazni, az éppen soron következő úthálózat minimális útjainak számát.

Megkötések és pontosítások

- $0<m,n<30$
- A minimális útvonalak száma nem fogja meghaladni a 2^{31}-1 értéket egyetlen esetben sem

Példa

ut.in	ut.out	Magyarázat
3 4 1 1 0 2 0 0 3 3 0 0 4 4 1 0 0 1 0 0 0 0	1 6 0	A bemeneti állomány három úthálózatot tartalmaz. Az első úthálózat 3 vízszintes és 4 függőleges útból áll és két lezárt útkereszteződése van (1,1) és (0,2) pontokban (lásd a fenti példát), a minimális hosszúságú utak száma 1. A második úthálózat 3x3-as méretű, nem tartalmaz lezárt utakat és a minimális hosszúságú utak száma 6. A harmadik úthálózat 4x4-es, két lezárt úttal, és egyetlen megoldása sincs.

Maximális futási idő/teszt: 0.1 másodperc

Rendelkezésre álló memória: 4 MB (amiből 2 MB a verem)

A forráskód maximális mérete: 50 KB

2. Feladat -Tényező

Tudjuk, hogy egy természetes szám faktoriálisa a következőképpen számítható ki: `n!= 1*2*...*n`. Például `5!=1*2*3*4*5=120`. A természetes számok faktoriális értékei nagyon hamar nőnek, például `7!=5040`, `10!=3628800`.

Ezért sokkal célszerűbb a faktoriálisokat törzstényezőkre bontott változatban tárolni.

Tekintsük a prímszámok halmazát (`2, 3, 5, 7, 11, 13, 17, ...`). Minden `p` természetes számnak egyértelműen megfeleltethetünk egy $a_1, a_2, \ldots a_k$ sorozatot, amelyben az a_i elem jelentése: az `i`-ik prímszám hatványa a `p` szám felbontásában.

Például $1071=3^2*7*17$, tehát a megfelelő sorozat: `(0, 2, 0, 1, 0, 0, 1)`, ami azt jelenti, hogy az `1071` felbontásában a `2` nem szerepel egyszer sem, a `3` pontosan kétszer szerepel, az `5` egyszer sem, a `7` egyszer, a `11` nem szerepel, a `13` sem, a `17` pedig egyszer szerepel.

Követelmény

Ismerve az `n` szám különböző értékeit, írjunk programot, amely az `n!` értékét felbontja a fenti módszerrel.

Bemenő adatok

A `tenyezo.in` bemeneti állomány több természetes számot tartalmaz, minden sorban egyet-egyet. Az állomány utolsó sorában egy 0 érték található, amit nem kell feldolgozni.

Kimenő adatok

A `tenyezo.out` kimeneti állomány a bemeneti állományban szereplő minden nem nulla elemnek megfeleltet egy-egy sort, amelyben az illető szám felbontása lesz. A felbontások elemeit egy-egy szóközzel választjuk el egymástól.

Pontosítások

`1< n <= 60000`

Példa

tenyezo.in	tenyezo.out	Magyarázat
2	1	2!=2
8	7 2 1 1	8!=2*2*2*2*2*2*2*3*3*5*7
15	11 6 3 2 1 1	15!= 2*2*2*2*2*2*2*2*2*2*2*3*3*3*3*3*3*5*5*7*7*11*13
10	8 4 2 1	10!=2*2*2*2*2*2*2*2*3*3*3*3*5*5*7
0		

Maximális futási idő/teszt: 0.5 másodperc

Rendelkezésre álló memória: 4 MB (amiből 2 MB a verem)

A forráskód maximális mérete: 50 KB

3. Feladat - pfh (Pascal/Fibonomiális háromszög)

1. A *Pascal háromszög* *i*-edik ($0 \le i < n$) sorának, *j*-edik ($0 \le j \le i$) eleme:

$P[i,j] = C_i^j$, ahol $C_i^j = \frac{i \cdot (i-1) \cdot \ldots \cdot (i-j+1)}{1 \cdot 2 \cdot \ldots \cdot j}$. A C_i^j kifejezés jelentése: „i elem j-ed osztályú kombinációja”.

A *Pascal háromszög* és a *Pascal mod 3 háromszög* első 8 sora a következőképpen néz ki:

	0	1	2	3	4	5	6	7
0	1							
1	1	1						
2	1	2	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
6	1	6	15	20	15	6	1	
7	1	7	21	35	35	21	7	1

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

Észrevesszük, hogy a *Pascal háromszög* nyolcadik sorának negyedik eleme: $P[7,3] = C_7^3 = \frac{7 \cdot 6 \cdot 5}{1 \cdot 2 \cdot 3} = 35$.

2. A *Fibonomiális háromszög* *i*-edik ($-1 \le i < n-1$) sorának *j*-edik ($0 \le j \le i+1$) elemét, az $F[i,j]$-t a következőképpen számoljuk ki

$F[i,i+1] = 1$ bármely $-1 \le i < n-1$.

$F[i,j] = \frac{f_j \cdot f_{j+1} \cdot \ldots \cdot f_i}{f_0 \cdot f_1 \cdot \ldots \cdot f_{i-j}}$, ha $0 \le i < n-1$ és $0 \le j \le i$,

ahol f_i az *i*-edik Fibonacci szám, azaz $f_0=1$, $f_1=1$ és $f_k=f_{k-1}+f_{k-2}$, ha $k>1$.

A *Fibonomiális háromszög* és *a Fibonomiális mod 3 háromszög* első 8 sora a következőképpen néz ki:

	0	1	2	3	4	5	6	7
-1	1							
0	1	1						
1	1	1	1					
2	1	2	2	1				
3	1	3	6	3	1			
4	1	5	15	15	5	1		
5	1	8	40	60	40	8	1	
6	1	13	104	260	260	104	13	1

	0	1	2	3	4	5	6	7
-1	1							
0	1	1						
1	1	1	1					
2	1	2	2	1				
3	1	0	0	0	1			
4	1	2	0	0	2	1		
5	1	2	1	0	1	2	1	
6	1	1	2	2	2	2	1	1

Észrevesszük, hogy a *Fibonomiális háromszög* nyolcadik sorának negyedik eleme: $F[6,3]=\frac{f_3 \cdot f_4 \cdot f_5 \cdot f_6}{f_0 \cdot f_1 \cdot f_2 \cdot f_3} = \frac{3 \cdot 5 \cdot 8 \cdot 13}{1 \cdot 1 \cdot 2 \cdot 3} = 260$.

Követelmény

Írj programot, amely az n ($0<n<50$) és p (p prímszám, $0<p<10$) természetes számok ismeretében, előállítja az n soros „*Pascal mod p*” és „*Fibonomiális mod p*” háromszögeket.

Bemeneti állomány

A **pfh.in** bemeneti állomány az első sorában három számot tartalmaz, egy-egy szóközzel elválasztva. Az első szám az n értéke, a második a p értéke, és a harmadik szám 0 vagy 1 aszerint, hogy a *Pascal mod p háromszög* vagy a *Fibonomiális mod p háromszög* első n sorát kell generálni.

Kimeneti állomány

A **pfh.out** kimeneti állomány a megfelelő háromszög első n sorát tartalmazza, az elemeket egy-egy szóközzel választjuk el.

Megkötések és pontosítások

- $0<n<49$;
- $0<p<10$, p prímszám.

Példa

pfh.in	pfh.out	Magyarázat
8 3 0	1 1 1 1 2 1 1 0 0 1 1 1 0 1 1 1 2 1 1 2 1 1 0 0 2 0 0 1 1 1 0 2 2 0 1 1	n=8, p=3 és az utolsó beolvasott szám 0, tehát a *Pascal mod* 3 *háromszög* első nyolc sorát kell kiírni
8 3 1	1 1 1 1 1 1 1 2 2 1 1 0 0 0 1 1 2 0 0 2 1 1 2 1 0 1 2 1 1 1 2 2 2 2 1 1	n=8, p=3 és az utolsó beolvasott szám 1, tehát a *Fibonomiális mod* 3 *háromszög* első nyolc sorát kell kiírni

Maximális futási idő/teszt: 0.5 másodperc

Rendelkezésre álló memória: 4 MB (amiből 2 MB a verem)

A forráskód maximális mérete: 50 KB