**A. "Все, Степан! Ти мене дістав!"**

[**http://www.e-olymp.com/en/problems/7239**](http://www.e-olimp.com.ua/ua/problems/7239)

Степан нещодавно відпочивав у Японії і привіз звідти нову жувальну гумку. На першій парі в університеті він поділився гумкою зі своїм товаришем. Дочекавшись моменту, коли лектор повернувся до дошки, на рахунок "три - чотири" хлопці дружньо почали надувати бульбашки. Відомо, що Степан надуває бульбашку до максимально можливого розміру за час ***t1***, після чого бульбашка миттєво лопається, і Степан починає надувати бульбашку заново з тією ж швидкістю. Товариш Степана робить те ж саме за час ***t2***.

Весь цей час викладач настільки захоплений доведенням теореми, що взагалі нічого не чує. І тільки коли обидві бульбашки лопнуть одночасно, викладач почує шум і обернеться. І тоді вже точно студентам попаде на горіхи, а більше усього тому, хто приніс на пару жувальні гумки.

Визначте, скільки часу хлопці можуть насолоджуватись надуванням бульбашок, не замічені викладачем.

Наприклад, якщо ***t1 = 2, t2 = 3***, то буде відбуватись наступне:

Степан надуває бульбашку з моменту часу ***t = 0*** до моменту часу ***t = 2***, потім бульбашка лопається, і він надуває бульбашку заново - з моменту часу ***t = 2*** до моменту часу ***t = 4***, а потім ще раз - з моменту часу ***t = 4*** до ***t = 6***.

Товариш Степана надуває бульбашку з ***t = 0*** до ***t = 3*** і ще раз з ***t = 3*** до ***t = 6***.

В момент часу ***t = 6*** бульбашки лопаються одночасно в обох студентів, викладач повертається і каже: "Все, Степан! Ти мене дістав!".

**Формат вхідних даних:** Перший рядок вхідного файлу містить два цілих числа ***t1, t2 (1 ≤ t1, t2 ≤ 109)***.

**Формат вихідних даних:** Вихідний файл повинен містити одне ціле число - час, протягом якого Степан з товаришем можуть насолоджуватись надуванням бульбашок.

**Приклад вхідних та вихідних даних:**

|  |  |
| --- | --- |
| **bubble.in** | **bubble.out** |
| 2 3 | 6 |
| 1 16 | 16 |

|  |  |
| --- | --- |
| #include "fstream"  using namespace std;  ifstream cin("input.txt");  ofstream cout("output.txt");  int main()  {long long a, b, a1, b1, t;  cin>>a>>b;  a1=a; b1=b;  while(b!=0)  {t=a%b;  a=b;  b=t;  }  long long nsd=a;  long long nsk=a1\*b1/nsd;  cout<<nsk<<endl;  return 0;  } | #include<fstream> using namespace std; ifstream in("bubble.in"); ofstream out("bubble.out"); int gcd(int a,int b) {         while (b)         {                 a%=b;                 swap(a,b);         }         return a; } long long a,b; int main() {         in>>a>>b;         out<<a/gcd(a,b)\*b<<endl; |

Степан вирішив сьогодні поекспериментувати з послідовністю натуральних чисел від 1 до N. Він спочатку викреслив усі непарні числа. Потім з тих, що залишились викреслив числа, які стоять на не парних місцях. Цю процедуру він повторював до тих пір, поки не залишилось тільки одно число.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8  2 4 6 8  4 8  8 | 1 2 3 4 5 6 7 8 9 10  2 4 6 8 10  4 8  8 | 1 2 3 4 5 6 7 8 9 10 11 12  2 4 6 8 10 12  4 8 12  8 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  2 4 6 8 10 12 14 16  4 8 12 16  8 16  16 |

Дано число n. Знайдіть кількість одиниць у бінарному записі числа 2"+n.

Формат вхідних даних

Перший рядок містить одне ціле число t (1<= 100) — кількість тестів.

Кожний з наступних t рядків містить одне ціле число n (0<n<1018).

Формат вихідних даних

Для кожного тесту вам потрібно вивести одне число в окремому рядку — кількість одиничних бітів числа 2n + n.

23 + 3 = 1110= 10112 — три одиничних біта.

21 + 1 = 310 = 112 -В два одиничних біта.

22 + 2 = 610 = 1102 — два одиничних біта.

**4 тур - з 03.12 по 10.12.2018**

точка входу для відправлення розв'язків  
<http://134.249.159.199//cgi-bin/new-client?contest_id=65>

**Задача A. Новорічні забави (100 балів)**

Обмеження по часу: 1 секунда

Обмеження по пам’яті: 512 мегабайт

В країні Фестляндії у лабораторії теоретичної піротехніки вивчають нові технології організації фейерверків. Фейерверк – це дерево, а оскільки кожен елемент фейерверку вибухає, створюючи нові фейерверки, то вчені виводять операцію піднесення дерева в степінь.

Дерево фейерверків містить одну або кілька вершин. Одна з вершин виділена та називається коренем, для кожної з решти вершин тільки одна інша вершина є батьківською. При цьому від будь-якої вершини можна дістатись до кореня, послідовно рухаючись від вершини до її батьківської вершини. Вершина, яка не є батьківською для жодної іншої вершини, - називається листом. Якщо вершина *х* є батьківською для вершини *y*, тоді вершина *у* називається нащадком вершини *х*. Кажуть, що вершина та її батьківська вершина з’єднані ребром.

На рис.1 показано приклад дерева з коренем в вершині 1. Батьківською вершиною для вершин 2 та 3 є вершина 1, батьківською вершиною для вершини 4 є вершин 2. Вершини 2 та 3 – нащадки вершини 1, а вершина 4 – нащадок вершини 2. Листами є вершини 3 та 4.

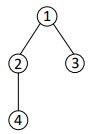


Рис.1 Приклад дерева з коренем в вершині 1, листами 3 та 4.

Фейерверк задається своїм базовим деревом Т та потужністю m. Фейерверк є деревом, яке отримується в результаті піднесення дерева Т до степеня m. Операція піднесення дерева до степеня побудована наступним чином. Якщо m=1, тоді результат Т1 – саме дерево. Т. Для m>1 розглянемо дерево Тm-1. Виконаємо наступну операцію: для кожного листа *х* дерева Тm-1 створимо копію дерева Т та призначимо лист *х* батьківською вершиною для кореня відповідної копії. Отримане дерево буде деревом Тm.

На рис. 2 показано дерево (рис.1), в степенях 1, 2 та 3.

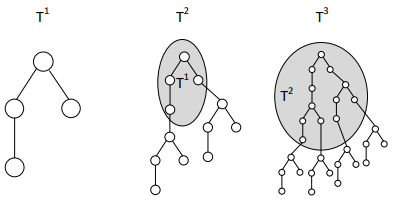


Рис.2. Приклад піднесення дерева до степенів 1,2 та 3.

Шляхом в дереві називається послідовність вершин, в якій дві сусідні вершини з’єднані ребром. Всі вершини на шляху – різні.

Щоб оцінити красу фейерверку, необхідно обчислити, яку максимальну кількість вершин може містити шлях в дереві, яким подається фейерверк. На рис.3 наведено шлях в дереві Т2, який містить максимальну кількість вершин. Отже, краса фейерверку з базовим дерево Т та потужністю 2 дорівнює 10.

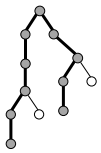


Рис. 3. Шлях в дереві Т2, який містить максимальну кількість вершин.

Необхідно написати програму, яка за описом дерева Т та натуральним числом m визначає красу фейерверку з базовим деревом Т та потужністю m.

**Формат вхідних даних**

Перший рядок вхідних даних містить два натуральних числа n та m – кількість вершин в базовому дереві фейерверку Т та його потужність (3≤*n*≤200 000, 1≤*m*≤200 000).

Другий рядок описує дерево Т та містить (*n*-1) цілих чисел: *p1*, *p2*, …, *pn* – номера батьківських вершин 2, 3, …, *n*, відповідно (1≤*pi*≤*i*-1).

**Формат вихідних даних**

Необхідно вивести одне ціле число – красу фейерверку, який представлений деревом Тm.

Приклад вхідних та вихідних даних

|  |  |
| --- | --- |
| input.txt | output.txt |
| 4 2  1 1 2 | 10 |

#include <fstream>

#include <algorithm>

#include <stdbool.h>

typedef bool uint1\_t;

using namespace std;

ifstream cin("input.txt");

ofstream cout("output.txt");

bool g[10000][10000];

long int n,m,c[1000000],mm=0,x,u;

bool f(int v,int d)

{

bool t=true;

for(int i=1;i<=d;i++)

if(v==c[i])t=false;

return t;

}

void p(int k,int v)

{c[k]=v;

if(k>mm){mm=k;u=v;}

for(int i=1;i<=n;i++)

if(g[v][i]==1 && f(i,k))p(k+1,i);

}

int main()

{cin>>n>>m;

for(long i=2;i<=n;i++)

{cin>>x;g[x][i]=1;g[i][x]=1;}

p(1,1);

int m1=mm;

//cout <<mm <<" "<<u<< endl;

p(1,u);

//cout <<mm <<" "<<u<< endl;

/\*for(int i=1;i<=n;i++){

for(int j=1;j<=n;j++)

cout<<g[i][j]<<" ";

cout<<endl;

\*/

int r=mm+2\*(m1)\*(m-1);

cout<<r<<endl;

return 0;

}

Остове дерево

// Floid.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <math.h>

using namespace std;

int main()

{const int n=25;

const int p=0;

int a[n][n];

int x[1000],y[1000],kol\_ver[1000],v[1000];

int k,i,j,vi,vj,min,s;

int ver[1000][3];

int f;

for(i=0;i<n;i++){

for (j=0;j<n;j++)

cout<<a[i][j]<<" ";

cout<<endl;

}

k=0; v[k]=p;s=0;

while (k<n-1) {

min=100000;

for (i=0;i<=k;i++)

for(j=0;j<n;j++)

if (a[v[i]][j]<min) {min=a[v[i]][j];vi=v[i];vj=j;}

f=1;

for (i=0;i<=k;i++)if (vj==v[i])f=0;

if (f==1) {k=k+1;

ver[k][1]=vi+1;

ver[k][2]=vj+1;

v[k]=vj;

kol\_ver[vj]=kol\_ver[vj]+1;

kol\_ver[vi]=kol\_ver[vi]+1;

s=s+a[vi][vj];

}

a[vi][vj]=1e30;a[vj][vi]=1e30;

}

cout<<s<<endl;

for(i=1;i<n;i++) cout<<ver[i][1]<<' '<<ver[i][2]<<endl;

system ("Pause");

return 0;

}

<https://www.e-olymp.com/uk/problems/1690>

<https://www.e-olymp.com/uk/problems/2298>

<https://www.e-olymp.com/uk/problems/3605>

<https://www.e-olymp.com/uk/problems/83>

<https://www.e-olymp.com/uk/problems/981>

<https://www.e-olymp.com/uk/problems/1445>

<https://www.e-olymp.com/uk/problems/1048>