# 第六次实验报告

## Q1

### Description

#### C. 埃及分数

**单点时限:** 1.0 sec
**内存限制:** 512 MB

在古埃及，人们使用不同单位分数的和（形如 $\frac{1}{a}$ ， $a$ 是自然数）表示一切有理数。

如： $\frac{2}{3}=\frac{1}{2}+\frac{1}{6}$ ，但不允许 $\frac{2}{3}=\frac{1}{3}+\frac{1}{3}$ ，因为加数中有相同的。

对于一个分数 $\frac{a}{b}$ ，表示方法有很多种，但是哪种最好呢?

首先，加数少的比加数多的好，其次，加数个数相同的，最小的分数越大越好。

如:

$\frac{19}{45}=\frac{1}{3}+\frac{1}{12}+\frac{1}{180}$

$\frac{19}{45}=\frac{1}{3}+\frac{1}{15}+\frac{1}{45}$

$\frac{19}{45}=\frac{1}{3}+\frac{1}{18}+\frac{1}{30}$

$\frac{19}{45}=\frac{1}{4}+\frac{1}{6}+\frac{1}{180}$

$\frac{19}{45}=\frac{1}{5}+\frac{1}{6}+\frac{1}{18}$

最后一种最好，因为 $\frac{1}{18}$ 比 $\frac{1}{180}$ 、 $\frac{1}{45}$ 、 $\frac{1}{30}$ 、 $\frac{1}{180}$ 都大。

给出 $a,b(1\leq a<b<1000)$ ，输出最佳表示方式。

### Input

#### 输入格式

输入仅包含一行两个用空格隔开的整数 $a,b(1\leq a<b\leq 1000)$ ，表示分数 $\frac{a}{b}$ ，我们并不保证 $\frac{a}{b}$ 是一个最简分数。

### Output

#### 输出格式

若干个数，自小到大排列，依次是单位分数的分母。

如果在满足加数个数最小且最小的分数最大的情况下仍有多组解，可以输出任意解。

### Example

#### 样例

| input |
| --- |
| 19 45 |
| output |
| 5 6 18 |

# Solution

这一题采用dfs的方式，主要有两个限制

1、加数的个数要最少（故而我们从两个加数开始搜索，如果两个加数没有解，那么搜索三个加数的情况，直到出现第一个可行解，第一个可行解的加数即是最少的加数）

2、最后一个加数的分母要尽可能小（这一条为我们搜索最优解提供约束）

开始搜索：

假设现在在**k**个加数这一个领域里搜索，并且已经搜索到了第**i**个加数（还剩**k-i+1**个加数未确定）

剩余的真分数（即原分数减去前面**i-1**个加数后剩余的部分）分子为**x**，分母为**y**

第**i**个加数的搜索限制具体为：

1、$y_i$（第i个加数的分母）> $y_{i-1}$

2、$(k-i+1) / y_i > x / y$ (后面的加数只会比第**i**个加数更小)

3、$1 / y_i < x / y$

更进一步优化有：

1、$1 / y_i > 1 / ans_k$ (已经搜索到可行解的情况下)

2、将正向搜索改为逆向搜索（采用$y_i$递增的方式搜索改为$y_i$递减的方式搜索）

递归终止条件为：**k == i**

若最终$x_k$ == 1, 即搜索到一个可行解

与暂存的可行解进行比较，最终得出最优解

# Code

## main.cpp

```cpp
#include <iostream>
#include <ctime>
using namespace std;
const int MAXN = 110;

long long a, b, anss, t;
long long ans[MAXN];
long long s[MAXN];

long long gcd(long long a, long long b)
{
    if (a > b)
        return gcd(a % b, b);
    if (a == 0)
        return b;
    return gcd(b % a, a);
}
```

```
bool dfs(long long k, long long x, long long y)
{
    if (x > y)
    {
        return dfs(k, y, x);
    }
    if (k == 1)
    {
        // if (x == 1)
        // {
        //     for (long long i = 1; i <= t; i++)
        //     {
        //         cout << s[i] << " ";
        //     }
        //     cout << y << endl;
        // }

        // cerr << "wrong" << endl;
        if (x == 1 && y > s[t] && (anss == 0 || y < ans[anss]))
        {
            anss = ++t;
            s[t] = y;
            for (long long i = 1; i <= t; i++)
                ans[i] = s[i];
            t--;
            return true;
        }
        return false;
    }

    bool getans = false;

    long long s1 = s[t] + 1;
    long long s2 = y / x + 1;
    //cerr << "wrong" << x << " " << y << endl;
    long long imax = k * y / x;
    if (ans[anss] != 0 && ans[anss] < imax)
        imax = ans[anss];
    long long im = s1 > s2 ? s1 : s2;
    for (long long i = imax; i >= im; i--)
    {
        // if (ans[anss] != 0 && i >= ans[anss])
        //     break;
        long long yy = y * i;
        long long xx = x * i - y;
        long long gg = gcd(xx, yy);
        xx /= gg;
        yy /= gg;
        s[++t] = i;
        if (dfs(k-1, xx, yy))
            getans = true;
        t--;
        // cerr << "wrong" << endl;
    }
    return getans;
}
```

```
int main()
{
    clock_t start, end;


    cin >> b >> a;
    start = clock();
    long long gd = gcd(a, b);
    if (gd != 1)
    {
        b /= gd;
        a /= gd;
    }
    if (b == 1)
    {
        cout << a << endl;
        return 0;
    }


    for (long long i = 2; i <=MAXN ;i++)
    {
        t = 0;
        anss = 0;
        // cout << "wrong" << endl;
        if (dfs(i, b, a))
        {
            for (long long i = 1; i <= anss; i++)
                cout << ans[i] << " ";
            cout << endl;

            end = clock();
             cout << "The run time is: " <<(double)(end - start) /
CLOCKS_PER_SEC << "s" << endl;

            return 0;
        }
        //cout << i << endl;
    }

}
```

# Q2

## Description

## D. Prime Path

**单点时限:** 2.0 sec
**内存限制:** 256 MB

The ministers of the cabinet were quite upset by the message from the Chief of Security stating that they would all have to change the four-digit room numbers on their offices.

— It is a matter of security to change such things every now and then, to keep the enemy in the dark.

— But look, I have chosen my number 1033 for good reasons. I am the Prime minister, you know!

— I know, so therefore your new number 8179 is also a prime. You will just have to paste four new digits over the four old ones on your office door.

— No, it's not that simple. Suppose that I change the first digit to an 8, then the number will read 8033 which is not a prime!

— I see, being the prime minister you cannot stand having a non-prime number on your door even for a few seconds.

— Correct! So I must invent a scheme for going from 1033 to 8179 by a path of prime numbers where only one digit is changed from one prime to the next prime.

Now, the minister of finance, who had been eavesdropping, intervened.

— No unnecessary expenditure, please! I happen to know that the price of a digit is one pound.

— Hmm, in that case I need a computer program to minimize the cost. You don't know some very cheap software gurus, do you?

— In fact, I do. You see, there is this programming contest going on... Help the prime minister to find the cheapest prime path between any two given four-digit primes! The first digit must be nonzero, of course. Here is a solution in the case above.

1033

1733

3733

3739

3779

8779

8179

The cost of this solution is 6 pounds. Note that the digit 1 which got pasted over in step 2 can not be reused in the last step – a new 1 must be purchased.

# Input

### 输入格式

One line with a positive number: the number of test cases (at most 100). Then for each test case, one line with two numbers separated by a blank. Both numbers are four-digit primes (without leading zeros).

# Output

### 输出格式

One line for each case, either with a number stating the minimal cost or containing the word Impossible.

# Example

### 样例

| input |
| --- |
| 3<br>1033 8179<br>1373 8017<br>1033 1033 |
| output |
| 6<br>7<br>0 |

## Solution

题目大致意思就是：从某个四位数质数号牌的房间到另一个四位数质数号牌的房间，搜索一条最短的质数路径，每一次转移，只能到同为质数号牌的房间，并且每次号牌只改变一位数字，

简单的bfs即可，用haveIn数组表示是否已经将经由这个房间的路径加入进队列中，

## Code

### main.cpp

```
#include <iostream>
#include <queue>
using namespace std;
const int MAXN = 1e5 + 10;
bool not_prime[MAXN];
int haveIn[MAXN];

void prime()
{
    not_prime[1] = 1;
    for (int i = 2; i * i <= MAXN;i++)
    {
        if(!not_prime[i])
        {
            for (int j = i * i; j <= MAXN; j+=i)
                not_prime[j] = 1;
        }
    }
}

int bfs(int src, int des)
{
    queue<int> st;
    st.push(src);
    haveIn[src] = 1;

    int s, s1;
    int rest1, rest2;
    int weight;
    while (!st.empty())
    {
        s = st.front();
        st.pop();
        // cout << s << endl;
        if (s == des)
```

```
            return haveIn[s];

        weight = 1000;

        for (; weight > 0; weight /= 10)
        {
            rest1 = s / (weight * 10);
            rest2 = s % weight;
            for (int i = 0; i < 10; i++)
            {
                s1 = rest1 * weight * 10 + weight * i + rest2;
                if (s1 >= 1000 && !not_prime[s1] && !haveIn[s1])
                {
                    st.push(s1);
                    haveIn[s1] = haveIn[s] + 1;
                }
            }
        }
    }
    return 0;
}

int main()
{
    prime();
    // cout << not_prime[8017] << endl;
    int N, a, b, ans;
    cin >> N;
    while (N--)
    {
        for (int i = 0; i < MAXN; i++)
            haveIn[i] = 0;
        cin >> a >> b;
        ans = bfs(a, b);
        cout << ans - 1 << endl;
    }
}
```