J. Xingqiu's Joke

题意

有 $T(1 \leq T \leq 300)$ 次询问，每次询问给定两个数字$a, b\ (1 \leq a, b \leq 10^9, a \neq b)$, 你可以对这两个数字进行如下三种操作任意次:

1. 让 a 和 b 同时减1

2. 让 a 和 b 同时加1

3. 如果 a 和 b 有一个公共质因数 p , 你可以让 a 和 b 同时除以这个质数

求最小的操作次数，使得 a 或者 b 等于 1

Example

input Copy

```
5
4 7
9 8
32 84
11 35
2 1
```

output Copy

```
2
7
5
4
0
```

Note

For the first sample test case, the optimal way is $(4, 7) \rightarrow (3, 6) \rightarrow (1, 2)$.

For the second sample test case, the optimal way is to apply the first type of operation 7 times.

For the third sample test case, the optimal way is $(32, 84) \rightarrow (16, 42) \rightarrow (15, 41) \rightarrow (14, 40) \rightarrow (13, 39) \rightarrow (1, 3)$.

For the fourth sample test case, the optimal way is $(11, 35) \rightarrow (12, 36) \rightarrow (6, 18) \rightarrow (2, 6) \rightarrow (1, 3)$.

分析

注意到，如果 a 和 b 要想同时除一个质数 p，必须满足:

$$a \equiv b \pmod{p}$$

这样 a 和 b 可以通过操作1或操作2达到 p 的某个倍数, 从而使用操作3除以 p

比如 a = 7, b = 16, p = 3

由于$7 \% 3 = 16 \% 3 = 1$

我们可以通过让 a b 一起减1，或者一起加2， 从而消掉余数，使得 a b 均可以被3整除

因此我们可以得到，如果 p 是一个可以在操作3中合法使用的质数， 那么 p 满足 :

$$b - a \equiv 0 \pmod{p}$$

也即所有操作3中可以合法使用的质数一定是 $b - a$ 的一个质因数。

令 $c = b - a$ ，注意到操作1和操作2并不会改变 c 的值， 而操作3会使 $c \rightarrow \frac{c}{p}$

也就是说给定了 a 和 b ， 通过质因数分解 $b - a$ ，所有可以进行操作3的质数我们都可以预见到， 并且每进行一次操作3， $b - a$ 的某一个质因数的次数就会减一。

比如 $b-a=120=2^3*3^1*5^1$ 意味着我们最多能使用操作3 让 a 和 b 除以3次2，1次3和1次5。

当我们想使用操作3除以某个质数的时候，我们可以使用前两个操作使得a走到这个质数的倍数。

这样问题就转换成了（假定$a<b$）：

Ⅰ. 先分解 $b-a$ 的质因数

Ⅱ. 枚举 $b-a$ 的质因数的排列，对于每种排列 $p_1,p_2,p_3...p_n$，从左往右对于每个质因数 p_i 让 a 加/减上某个数，使得 a 等于最近的 p_i 的 倍数，再让 a 除以 p_i (比如7想要除以3，要么向上走到9，要么向下走到6)，直到 $a=1$。

容易想到一种dfs做法

```
void dfs(int x, int y, int op)
{
    // ans 表示最优答案
    // op 表示当前状态的操作次数
    // x 即问题中的 a， y 即问题中的 b - a
    if (op >= ans) return;
    if (x == 1)
    {
        ans = op;
        return;
    }

    dfs(1, y, op + x - 1); //一直做减法减到1也是一种合法的操作

    //f 存初始 b - a 的每种质因数
    for (int i = 0; i < f.size(); i ++)
    {
        if (y % f[i]) continue; //f[i]不能整除y，表示这个质因数的次数已经被用完了

        int d = x % f[i];//往下方逼近当前质数的倍数的距离
        int u = f[i] - x % f[i]; //往上方逼近当前质数的倍数的距离

        if (x - d > 0) dfs((x - d) / f[i], y / f[i], op + d + 1);
        dfs((x + u) / f[i], y / f[i], op + u + 1);
    }
}
```

通过这种做法，我们得到了TLE的好成绩

Problem	Lang	Verdict	Time	Memory	Sent	Judged		
1034701 - 22	GNU C++20 (64)	Time limit exceeded on test 2	2000 ms	204 KB	2023-03-03 05:56:13	2023-03-03 05:56:13	☆	Compare

考虑记忆化搜索优化

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
#include <queue>
#include <unordered_map>
#include <unordered_set>
```

```cpp
#include <map>
#include <set>

using namespace std;
typedef long long ll;
const int N = 4e4;

map<pair<int, int>, ll> dp; //由于值域较大我们无法用二维数组存储状态，所以使用map来存
ll ans;
int primes[N], cnt;
bool st[N];
vector<int> f;

//线性筛预处理出1~sqrt(1e9)的所有质数
void get_primes(int n)
{
    for (int i = 2; i <= n; i ++ )
    {
        if (!st[i]) primes[cnt ++ ] = i;
        for (int j = 0; primes[j] <= n / i; j ++ )
        {
            st[primes[j] * i] = true;
            if (i % primes[j] == 0) break;
        }
    }
}

//分解质因数
vector<int> factors(int x)
{
    int i = 0;
    vector<int> ans;
    while (x > 1)
    {
    //如果一个数试除完了1~sqrt(1e9)的所有质数仍然不等于1，说明剩下的那个数是一个大于
sqrt(1e9)的大质数
        if (x <= primes[i] || i >= cnt)
        {
            ans.push_back(x);
            break;
        }
        if (x % primes[i] == 0) x /= primes[i], ans.push_back(primes[i]);
        else i ++;
    }
    return ans;
}

//记忆化搜索
ll dfs(int x, int y)
{
    if (dp.count({x, y})) return dp[{x, y}]; //这个状态已经搜索过了
    if (x == 1) return 0; //边界条件

    dp[{x, y}] = x - 1; //一直减到1是一种合法的方案
    for (int i = 0; i < f.size(); i ++)
```

```
    {
        //同dfs做法
        if (y % f[i]) continue;

        int d = x % f[i];
        int u = f[i] - x % f[i];
        dp[{x, y}] = min(min(dfs((x - d) / f[i], y / f[i]) + d, dfs((x + u) /
f[i], y / f[i]) + u) + 1, dp[{x, y}]);
    }
    return dp[{x, y}];
}
void solve()
{
    int a, b;
    cin >> a >> b;
    dp.clear();

    if (a > b) swap(a, b);
    f = factors(b - a); //分解质因数
    f.erase(unique(f.begin(), f.end()), f.end());
    //去重，比如将数组{2, 2, 2, 3, 5} 变成 {2, 3, 5}
    cout << dfs(a, b - a) << '\n';
}
int main()
{
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    int tt;
    cin >> tt;
    get_primes(N - 1); //筛子
    while (tt --)
    {
        solve();
    }
    return 0;
}
```

比较极限地通过了这一题

Problem	Lang	Verdict	Time	Memory	Sent	Judged		
103470J - 22	GNU C++20 (64)	Accepted	1793 ms	480 KB	2023-03-03 07:05:38	2023-03-03 07:05:38		Compare

一种优化方法是将map替换成unordered_map，但是unordered_map不支持哈希pair类型，需要自己写一个哈希函数