H. ddd 和渡渡鸟

请找到一个长度最大的整数数组 r（每个数绝对值不超过 10^9），使得这个数组中任意连续 a 个数的和是正的，任意连续 b 个数的和是负的。

数据范围：$1 \le a, b \le 3 \cdot 10^5$

出题人：杨栋

概览

共 6 人尝试了这道题， 1 人通过。这道题目成了本次比赛的“冠军题”。通过的选手是 $O(n)$ 的最优算法，其他选手们有的尝试了不同种类的贪心，有的输出了 `gg`。

算法一

显然数组的长度是可以二分的。那么假如我们固定了数组长度是 x，如何检验呢？考虑 r 的前缀和数组 s，（即 $s_i = \sum_{j=1}^{i} r_j,\ i = 0, 1, \ldots, x$），则上面的条件就等价于对于所有合法的 i，$s_i > s_{i-a}$ 且 $s_i < s_{i-b}$。要用图论的语言表示上面条件，我们建立一个 $x+1$ 个点的图（最左侧新增一个零号节点），连两种类型的边：$i \to i-b$ 与 $i-a \to i$ 表示小于关系。然后原问题就转化为了典型的拓扑排序问题，有环对应无解，无环一定可以构造出方案。

不妨设 $n = \max\{a, b\}$ 因此我们得到了时间复杂度是 $O(n \log R)$ 的算法，其中 R 是二分上界。经过简单证明或实验可以发现 R 不超过 $a+b$，所以此算法时间复杂度是 $O(n \log n)$。

算法二

事实上，数组的长度不需要二分就可以直接计算出来！答案是 $a+b-\gcd(a,b)-1$。如果观察出或证明出此结论，就只需要进行一次拓扑排序，从而就得到了时间复杂度是 $O(n)$ 的优秀算法。

下面给出答案是 $a+b-\gcd(a,b)-1$ 的证明，分两种情况：

1. 当 $\gcd(a,b) = 1$ 时

   i. 一方面，若 r 的长度是 $a+b-1$，则上面建出的图共 $a+b$ 个点，可以发现每个点都恰有一条出边，随意选定一个起点，不断沿着当前点的出边移动，最终一定会通过某个之前经过的点，也就是找到了环。故答案小于 $a+b-1$。

   ii. 另一方面，若 r 的长度是 $a+b-2$，则上面建出的图共 $a+b-1$ 个点。由于 $\gcd(a,b)=1$，所以一个简单环至少要往右走 b 次，往左走 a 次。但是现在总共点数只有 $a+b-1$，不够 $a+b$，所以这时的图无环。

2. 当 $\gcd(a,b) = d, d \ne 1$ 时

   此时我们建出的图如果把节点按照编号 $\pmod d$ 的余数分组，那么不同组的点之间不可能有边。而每一组都和互质的情况一样，最多有 $\frac{a}{d} + \frac{d}{b} - 1$ 个节点。所以合起来图中最多可以有 $a+b-d$ 个节点。去掉零号节点，数组的长度最大就是 $a+b-d-1$ 了。

彩蛋

验题时，这道题被发现是 1977 年第十九届 IMO 第二道题的加强版。IMO 题是固定了 $a = 11, b = 7$ 的情况，官方的解答方法是利用算两次原理证明上界，以及利用手动构造证明了下界。但是这道题用图论的视角来证明与解答更加自然。

代码

算法一代码（357 ms, 16 MB）：

```
// Author: Yang Dong
#include<bits/stdc++.h>
using namespace std;

int a, b; // a pos, b neg
vector<int> check(int x) {
    vector<int> ret, val(x + 1);
    vector<array<int,2>> in(x + 1, {-1, -1}), out(x + 1, {-1, -1});
    int cloc = 0;
    queue<int> q;
    for(int i = 0; i <= x; ++i) {
        if(i + b <= x) out[i + b][0] = i, in[i][0] = i + b;
        if(i - a >= 0)   out[i - a][1] = i, in[i][1] = i - a;
        if(in[i][0] == -1 && in[i][1] == -1) q.push(i);
    }
    while(!q.empty()) {
        int u = q.front(), v; q.pop();
        val[u] = ++cloc;
        if(out[u][0] != -1) {
            v = out[u][0];
            in[u][0] = out[u][0] = -1;
            if(in[v][1] == -1) q.push(v);
        }
        if(out[u][1] != -1) {
            v = out[u][1];
            in[v][1] = out[u][1] = -1;
            if(in[v][0] == -1) q.push(v);
        }
    }
    if(cloc != x + 1) return {};
    for(int i = 1; i <= x; ++i)
        ret.push_back(val[i] - val[i - 1]);
    return ret;
}
int main() {
    cin >> a >> b;
    int l = 0, r = 1000000;
    while(l < r) {
        int mid = (l + r) >> 1;
        if(l == r - 1) mid = r;
        if(check(mid).size()) l = mid;
        else r = mid - 1;
    }
    auto z = check(l);
```

```
    printf("%d\n", l);
    for(auto g : z) printf("%d ", g);
    return 0;
}
```

算法二代码（80 ms, 12 MB）：

```
// Author: Yang Dong
#include<bits/stdc++.h>
using namespace std;

int gcd(int x, int y) { return y == 0 ? x : gcd(y, x % y);}
const int MAXN = 606060;
int in[MAXN][2], out[MAXN][2], val[MAXN];
int main() {
    memset(in, -1, sizeof in);
    memset(out, -1, sizeof out);
    int a, b, ans, cloc = 0;
    cin >> a >> b;
    ans = a + b - gcd(a, b) - 1;
    queue<int> q;
    for(int i = 0; i <= ans; ++i) {
        if(i + b <= ans) out[i + b][0] = i, in[i][0] = i + b;
        if(i - a >= 0)   out[i - a][1] = i, in[i][1] = i - a;
        if(in[i][0] == -1 && in[i][1] == -1) q.push(i);
    }
    while(!q.empty()) {
        int u = q.front(), v; q.pop();
        val[u] = ++cloc;
        if(out[u][0] != -1) {
            v = out[u][0];
            in[u][0] = out[u][0] = -1;
            if(in[v][1] == -1) q.push(v);
        }
        if(out[u][1] != -1) {
            v = out[u][1];
            in[v][1] = out[u][1] = -1;
            if(in[v][0] == -1) q.push(v);
        }
    }
    printf("%d\n", ans);
    for(int i = 1; i <= ans; ++i)
        printf("%d ", val[i] - val[i - 1]);
    return 0;
}
```