1. (a) \(\Rightarrow\) Let \(m = [a_1, \ldots, a_n]\). By definition \(m\) is the least number that satisfies (i) and (ii). We just have to show \(m\) satisfies (iii).

Suppose \(a_1, \ldots, a_n|m'\) and \(m \not| m'\). Then \(m' = Mm + r\) where \(1 \leq r < m\). Since \(a_1, \ldots, a_n|m, a_1, \ldots, a_n|m' - Mm = r\). We have shown \(r\) satisfies (i) and (ii) and thus \(r \geq m\), a contradiction. Therefore we must have \(m| m'\).

\((\Leftarrow)\) Suppose \(m\) satisfies (i)-(iii). Suppose \(m'\) satisfies (i) and (ii) but \(m' < m\). By (iii) we have \(m|m'\) which implies \(m \leq m'\), a contradiction. Thus \(m\) is the least number satisfying (i) and (ii) and so \(m = [a_1, \ldots, a_n]\), by definition.

(b) \(\Rightarrow\) Let \(d = (a_1, \ldots, a_n)\). Thus \(d\) satisfies (i) and (ii) by definition.

Now we show \(d\) satisfies (iii). Suppose \(d'\) is a divisor of \(a_1, \ldots, a_n\). Since there exist \(x_1, \ldots, x_n\) so that \(d = a_1x_1 + \cdots + a_nx_n\), we have \(d'|d\), as desired.

\((\Leftarrow)\) Suppose \(d\) satisfies (i)-(iii). Suppose \(d'\) is a divisor of \(a_1, \ldots, a_n\) but \(d' \geq d\). By (iii) we have \(d'|d\) which implies \(d' \leq d\), a contradiction. Thus \(d\) is the greatest number satisfying (i) and (ii) so \(d = (a_1, \ldots, a_n)\) by definition.

2. First we describe the algorithm:

Initialize \(k \leftarrow 1\), \(x_0 \leftarrow a\), \(y_0 \leftarrow b\).

Repeat:

If \(y_{k-1} = 0\) stop and return \(x_{k-1}\).

Calculate \(m, r\) such that \(x_{k-1} = my_{k-1} + r\) and \(|r| \leq (1/2)|y_{k-1}|\).

Let \(x_k \leftarrow y_{k-1}\), \(y_k \leftarrow r\).

Let \(k \leftarrow k + 1\).

Suppose the algorithm receives \(a \geq b > 0\) as input. Let \(m = \text{size}(a)\), \(n = \text{size}(b)\). Since \(|y_0| = |b|\) and \(|y_k| \leq (1/2)|y_{k-1}|\) for all \(k \geq 1\) we have \(|y_k| \leq (1/2^k)|b|\) for \(k \geq 0\). Also \(|x_k| = |y_{k-1}| \leq (1/2^{k-1})|b|\) and \(|x_0| = \lfloor a \rfloor\). Since \(n\) is the number of bits used to represent \(b\) in binary, \(|b| < 2^n\) and so \(|y_n| < 1\). Thus for some \(k' \leq n\) we have \(y_{k'} = 0\) and the algorithm outputs \(x_{k'}\).

We have \((a, b) = (x_0, y_0)\) and for all \(k \geq 1\) we have \((x_k, y_k) = (y_{k-1}, x_{k-1} - my_{k-1}) = (x_{k-1}, y_{k-1})\). Thus the algorithm correctly outputs \(x_{k'} = \ldots\)
\((x_{k'}, 0) = (x_{k'}, y_{k'}) = (x_0, y_0) = (a, b)\) after \(k' \leq n\) cycles through the main loop.

During the \(k\)th cycle through the loop, the algorithm calculates a division algorithm on \(x_{k-1}, y_{k-1}\) which we assumed will take \(\leq \text{size}(x_{k-1})\text{size}(y_{k-1}) \leq Cmn\) steps. There are \(k' \leq n\) cycles so the algorithm stops after \(\leq Cmn^2\) steps as claimed.

You can be more careful. In fact, the algorithm takes \(\leq \sum_{k=1}^{k'} C\text{size}(x_{k-1})\text{size}(y_{k-1}) = C(mn + \sum_{k=2}^{k'} (1/2^{k-1})m(1/2^k)m) \leq C(mn + (1/6)m^2)\) steps.

3. These equations are solvable if and only if \(2 \mid a + b\) and \(6 \mid 2a - 3b + c\).

If \(a = b = c = 1\) then the solutions are \(x_1 = 1 - 2x_2' + 3x_3' - 4x_4', x_2 = x_2' - 3x_3' + 6x_4', x_3 = x_3' - 4x_4', x_4 = x_4'\) where \(x_2', x_3', x_4'\) arbitrary.

4. If \(a \equiv b \pmod{m}\) and \(c \equiv d \pmod{m}\) then \(m|(a-b)\) and \(m|(c-d)\).

So \(m|(A-b)c + (c-d)b = (ac - bc) + (bc - bd) = ac - bd\) or \(ac \equiv bd \pmod{m}\).
5. Suppose $x \equiv y \pmod{m}$. Then $m|(x - y)$ and $m_i|m|(x - y)$ or $x \equiv y \pmod{m_i}$ for all $i$. Let $d = x - y$. We now show that if $(m_i, m_j) = 1$ for all $i \neq j$ and $m_i|d$ (i.e. $x \equiv y \pmod{m_i}$) for all $i$ then $m|d$ (i.e. $x \equiv y \pmod{m}$). We prove this by induction on $r$. If $r = 1$, there is nothing to prove. Suppose now that $r > 1$. Since $m_r|d$ we have $d = k m_r$. Since $(m_i, m_r) = 1$ and $m_i|d = k m_r$ we have $m_i|k$ for $1 \leq i \leq r - 1$. Thus $m_1 m_2 \ldots m_{r-1}|k$ by induction and so $m_1 m_2 \ldots m_r|k m_r = d$. 