You are standing next to a water source. You have two empty jugs: A 3 litre jug and a 5 litre jug with no marks on them. You must fill the larger jug with precisely 4 litres of water. Can you do it?

This scenario is straight out of the 1995 classic "Die Hard 3" featuring Bruce Willis and Samuel L. Jackson. Should they fail to complete this task within 3 minutes, a bomb goes off and New York City is obliterated. In the nick of time, Bruce and Sam come up with a solution:

large jug	small jug	action
—	3ℓ	fill up small jug from source
3ℓ	—	transfer water into large jug
3ℓ	3ℓ	fill up small jug
5ℓ	1ℓ	top up large jug from small jug
—	1ℓ	spill large jug
1ℓ	—	transfer water into large jug
1ℓ	3ℓ	fill up small jug
4ℓ	—	done!

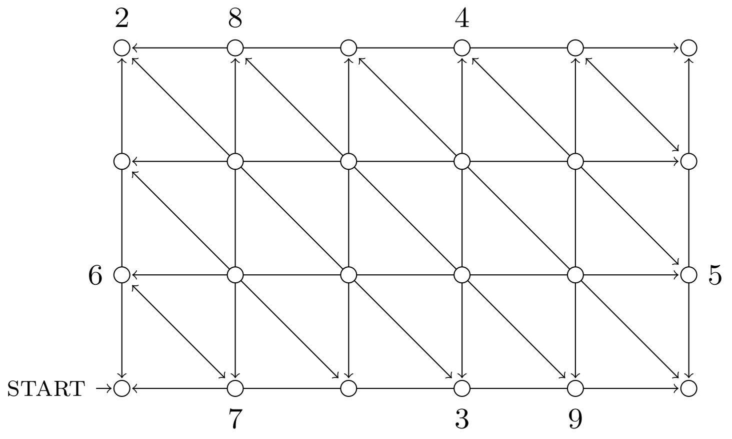
In the sequels that were never filmed, Bruce and Sam were to be given 21 and 26 litre jugs in Die Hard 4, 899 litre and 1,147 litre jugs in Die Hard 5; and 6 and 9 litre jugs in Die Hard 6. What should they have done? Why, state machines, of course.

1 The "Die Hard" state machine

The water jug game is modeled by a state machine. The states are pairs of numbers (x, y) describing the amount of water in the large and smaller jug, respectively. The transitions represent the possible pouring moves. For example, the transition $(1, 2) \to (3, 0)$ represents the move in which starting with 1 litre of water in the 5 litre jug and 2 litres of water in the 3 litre jug, the water is poured from the smaller into the larger jug, resulting with 3 litres in the larger jug and no water in the smaller one.

The initial state is $(0, 0)$, and we are interested in reaching a state of the form $(4, y)$.

We can draw a diagram for the Die Hard 3 state machine. The sequence of transitions START $\to 2 \to 3 \to \cdots \to 9$ achieves the desired objective.



In general, the jugs can be large and we have to describe the state machine mathematically. If a and b are the jug capacities the possible states are the pairs

$$(x, y) \text{ such that } 0 \le x \le a \text{ and } 0 \le y \le b.$$

There are at most six transitions out of each state (represented by the horizontal, vertical, and diagonal arrows in the picture) given by the pouring rules:

1. **Spill a jug:** The spilled jug now contains 0 litres, while the contents of the other jug stay the same.
$$(x, y) \to (0, y) \tag{T1}$$
$$(x, y) \to (x, 0) \tag{T2}$$

2. **Fill up a jug from the source:** The filled up jug now contains as much water as its capacity. The contents of the other jug stay the same.
$$(x, y) \to (a, y) \tag{T3}$$
$$(x, y) \to (x, b) \tag{T4}$$

3. **Transfer water between jugs:** Water can be poured from one jug into the other until one of them is full or empty. These transitions are trickier to write down, but the effort will pay off later on.

   If water is poured from the first into the second jug then two things can happen. If the total amount of water does not exceed the capacity b of the second jug, the second one will contain $x + y$ litres. Otherwise, the second jug fills up to its capacity b, and $x + y - b$ litres of water remain in the first jug:
$$(x, y) \to \begin{cases} (0, x+y), & \text{if } x + y \leq b, \\ (x+y-b, b), & \text{if } x + y > b. \end{cases} \tag{T5}$$

   We have analogous transitions for pouring water from the second jug into the first one:
$$(x, y) \to \begin{cases} (x+y, 0), & \text{if } x + y \leq a, \\ (a, x+y-a), & \text{if } x + y > a. \end{cases} \tag{T6}$$

We can now formulate the water jug game with a v litre target question mathematically: Is there a sequence of transitions that reaches a state of the type (x, v) or (v, y)? If you play more with the Die Hard 3 state machine, you will notice that this is possible for any target v (as long as it is an integer between 0 and 5). But what happens in the Die Hard 4, 5, and 6 scenarios?

In Die Hard 6 something different happens: No matter what you do, the amount of water in either jug is always 3, 6, or 9 litres. To describe this behaviour we introduce the concept of integer linear combination.

Integer linear combinations

A number c is an *integer linear combination* (in short, *combination*) of a and b if there exist integers s and t such that $c = s \cdot a + t \cdot b$. For example,

- 2 is a combination of 4 and 6 because $2 = (-1) \cdot 4 + 1 \cdot 6$.
- 1 is a combination of 3 and 5 because $1 = 2 \cdot 3 + (-1) \cdot 5$.
- 2 is a combination of 2 and 6 because $2 = 1 \cdot 2 + 0 \cdot 6$.

We can now state an invariant of the Die Hard state machines.

Invariant 1. *The amount of water in each jug is always a combination of a and b.*

To prove this invariant, the following property of integer combinations will be useful.

Lemma 1. *If x and y are combinations of a and b, then any combination of x and y is also a combination of a and b.*

For example, 6 and 9 are both combinations of 12 and 15:

$$6 = 3 \cdot 12 - 2 \cdot 15$$
$$9 = 2 \cdot 12 - 15$$

and 3 is a combination of 6 and 9:

$$3 = -6 + 9.$$

By combining the equations we can write 3 as a combination of 12 and 15:

$$3 = -6 + 9 = -(3 \cdot 12 - 2 \cdot 15) + (2 \cdot 12 - 15) = -12 + 15.$$

We can generalize this reasoning into a proof of Lemma 1.

Proof of Lemma 1. If x and y are combinations of a and b, we can write $x = s \cdot a + t \cdot b$ and $y = s' \cdot a + t' \cdot b$ for some integers s, t, s', t'. Then any combination of x and y has the form

$$p \cdot x + q \cdot y = p \cdot (sa + tb) + q \cdot (s'a + t'b) = (ps + qs') \cdot a + (pt + qt') \cdot b$$

so it is a combination of a and b. □

We will now apply Lemma 1 to prove the invariant.

Proof of Invariant 1. We will prove that the invariant holds in the initial state and that it is preserved by the transitions. It then follows by induction that the invariant holds after an arbitrary number of steps.

Initial state: Initially, each jug has 0 liters of water and $0 = 0 \cdot a + 0 \cdot b$.

Transitions: Assume the invariant holds in state (x, y). namely both x and y are combinations of a and b. We show that this remains true after any possible transition $(x, y) \to (x', y')$. In all the transitions T1-T6, x' and y' are combinations of x, y, a, b. By Lemma 1, x' and y' are therefore combinations of a and b only. □

Greatest common divisors

We say d *divides* a if $a = kd$ for some integer k. For example, 2 divides 4 and -6 but 2 does not divide 5; 4 does not divide 2; and every number divides zero.

Notice that d divides a number a *if and only if* a is a combination of d and zero.

Lemma 2. *If d divides a and b then d divides all combinations of a and b.*

Proof. If d divides a and b then a and b are combinations of d and zero. By Lemma 1 so is any combination of a and b, so d divides this combination. □

The *greatest common divisor (GCD)* of two integers a, b is the largest integer d so that d divides a and d divides b. We write $\gcd(a, b)$ for the GCD of a and b. For example, $\gcd(2, 6) = 2$, $\gcd(4, 6) = 2$, $\gcd(3, 5) = 1$, $\gcd(0, 6) = 6$.

Combining Invariant 1 and Lemma 2, we obtain a useful corollary:

Corollary 3. $\gcd(a, b)$ *always divides the amount of water in each jug.*

In particular, a 6 litre jug and a 9 litre jug can never measure 4 litres of water:

Corollary 4. *In Die Hard 6, Bruce and Sam don't make it out alive.*

Proof. The GCD of 6 and 9 is 3. By Corollary 3, 3 must divide the amount of water in both in all reachable states. So there can never be 4 litres in either jug. □

2 Euclid's algorithm

Euclid's algorithm is a procedure for calculating the GCD of two numbers. It was invented by the Euclideans around 3,000 years ago and it still the fastest procedure for this task. To explain Euclid's algorithm, we need to recall division with remainder.

Theorem 5. *Let n and d be integers with $d > 0$. There exists unique integers q and r such that*

$$n = q \cdot d + r \quad \textit{and} \quad 0 \leq r < d.$$

For example, if $n = 13$ and $d = 3$, we can write $13 = 4 \cdot 3 + 1$. Moreover, $q = 4$ and $r = 1$ are unique assuming r is in the range $0 \leq r < d$.

The numbers q and r can be calculated by the usual "division with remainder rule" that you learned in school.

Proof. First, we show existence: Let q be the largest integer such that $qd \leq n$. That means $(q+1)d > n$. Then $0 \leq n - qd < d$. Set $r = n - qd$.

Now we show uniqueness: Suppose we can write n in the desired form in two different ways:

$$n = qd + r, 0 \leq r < d$$
$$n = q'd + r', 0 \leq r' < d.$$

Then $qd + r = q'd + r'$, so $(q - q')d = r' - r$. Therefore $r' - r$ divides d. Since $0 \leq r, r' < d$ we must have $-d < r' - r < d$. The only number in this range that divides d is zero, so $r' - r = 0$ and $q' - q = 0$. It follows that $q' = q$ and $r' = r$, so the two representations are the same. □

Euclid's algorithm is a *recursive* algorithm for calculating the GCD of positive integers.

Euclid's algorithm $E(n, d)$, where n, d are integers such that $n \geq d \geq 0$.

```
  If $d = 0$, return $n$.
  Otherwise, write $n = qd + r$ where $0 \leq r < d$. Return $E(d, r)$.
```

Let's apply Euclid's algorithm to calculate the GCD of 1147 and 899:

$$\begin{aligned}
E(1147, 899) &= E(899, 248) && \text{because } 1147 = 1 \cdot 899 + 248 \\
&= E(248, 155) && \text{because } 899 = 3 \cdot 248 + 155 \\
&= E(155, 93) && \text{because } 248 = 1 \cdot 155 + 93 \\
&= E(93, 62) && \text{because } 155 = 1 \cdot 93 + 62 \\
&= E(62, 31) && \text{because } 93 = 1 \cdot 62 + 31 \\
&= E(31, 0) && \text{because } 62 = 2 \cdot 31 + 0 \\
&= 31.
\end{aligned}$$

How can we be sure that Euclid's algorithm indeed outputs the GCD of n and d? This is a theorem that we will have to prove.

To analyze Euclid's algorithm we can model it as a state machine. The states are pairs of integers. The transitions have the form $(n, d) \to (d, r)$ where r is the remainder of dividing n by d assuming $d \geq 0$.

Invariant 2. *The* gcd *of the state stays the same throughout the execution of Euclid's algorithm.*

Proof. Consider any transition $(n, d) \to (d, r)$. Recall that n is of the form $q \cdot d + r$. By Corollary 2, every common divisor of d and r also divides their integer combination $n = qd + r$, so it is a common divisor of n and d.

By Lemma 2 again, every common divisor of n and d also divides their integer combination $r = n - q \cdot d$, so it is a common divisor of d and r.

It follows that the common divisors of n and d are the same as those of d and r, so the gcd of both pairs must be the same. □

Invariant 2 shows that Euclid's algorithm preserves the gcd of its inputs throughout its execution. Is this a good enough reason to conclude that the algorithm is correct? Not really, because for all we know the algorithm may end up running forever! We must also argue that the algorithm eventually terminates.

Theorem 6. *For all integers $n \geq d \geq 0$, $E(n, d)$ terminates and outputs* $\gcd(n, d)$.

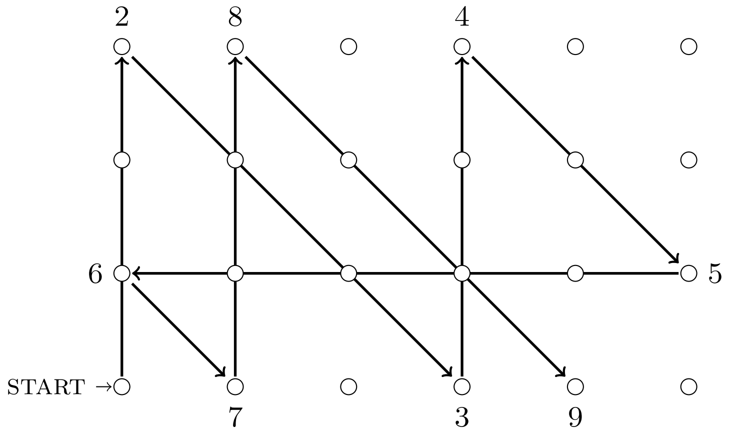
Proof. For termination, we observe that the value of d decreases in each transition but remains non-negative because d is replaced by the remainder r which is always smaller than d but non-negative. Therefore the algorithm must terminate after a finite number of steps.

The algorithm terminates when it reaches a state of the form $(n, 0)$ in which case it outputs $n = \gcd(n, 0)$. By Invariant 2, its output must equal the gcd of its inputs. □

3 How to Die less Hard

Corollary 3 says that if the amount v is *not* a multiple of the gcd of the jug capacities a and b, then Bruce dies for sure. If it is not, can he actually survive?

To answer this question, let's look at what happened in Die Hard 3 again. Observe the curious zigzagging pattern taken by our state machine:



In words, we keep zigzagging up-diagonal-up-diagonal until we hit the right boundary. Then we move all the way left, continue along the diagonal, and keep going until we get lucky. We can rephrase this strategy as a candidate algorithm:

```
ALGORITHM DieHard
INPUTS: jug capacities $a, b$ and target $v$.
1    Repeat until jug $A$ contains $v$ litres:
2        Fill up jug $B$.
3        Pour as much as possible from jug $B$ into jug $A$.
4        If jug $A$ is full,
5            Spill jug $A$.
6            Pour out the remaining contents of jug $B$ into jug $A$.
```

How do we know that this algorithm works? Notice how in a single iteration of the loop 2-6, jug B is filled up once and spilled once. Since all the water from jug B ends up being poured into jug A, after the s-th iteration of the loop jug A has received $s \cdot b$ litres from jug B. Owing to step 5, some amount has been poured out of jug A. We don't know how much this is, but it is always an integer multiple of its capacity a. In summary,

After s iterations, jug A contains $s \cdot b - t \cdot a$ litres for some nonnegative integer t.

Let's see how this checks out with our Die Hard 3 scenario:

iteration	transitions	water in large jug A
1	START $\to 2 \to 3$	$3 = 1 \cdot 3 - 0 \cdot 5$
2	$3 \to 4 \to 5 \to 6 \to 7$	$1 = 2 \cdot 3 - 1 \cdot 5$
3	$7 \to 8 \to 9$	$4 = 3 \cdot 3 - 1 \cdot 5$

The number t is uniquely determined by s, a, and b in every iteration. If it was any smaller the large jug would overfill, and it was any larger the amount of water in it would be negative! We can summarize our reasoning in the following lemma.

Lemma 7. *If $0 < v < a$ and $v = s \cdot b - t \cdot a$ for some non-negative integers s, t, then algorithm* DieHard *has terminated within s iterations.*

Proof. If algorithm DieHard goes on for s iterations, $s \cdot b$ litres have been poured into jug A, and $t' \cdot a$ litres have been poured out of it for some integer t'. The amount of water in jug A after s iterations is then $s \cdot b - t' \cdot a = v + (t - t') \cdot a$.

We claim that t' must equal t: If $t' > t$ then jug A would contain $v - a$ litres which is negative, while if $t' < t$ then jug A would contain $v + a$ litres, which exceeds in capacity. Both cases give a contradiction. It follows that $t' = t$ and jug A contains exactly v litres after s iterations. □

Lemma 7 tells us that if you want to be a Die Hard star, all you need to do is write the target v as $s \cdot b - t \cdot a$. The algorithm will do the rest of the work. We know that this is impossible unless v happens to a multiple of $\gcd(a, b)$. The converse is also true:

Lemma 8. *For all a and b there exist non-negative integers s and t such that $\gcd(a, b) = s \cdot b - t \cdot a$.*

We can now forego the tedium of watching Die Hard 4, 5, 6, 7 up to infinity because we know what will happen:

Theorem 9. *Assume $a \geq b$. Bruce and Sam survive if and only if $0 \leq v \leq a$ and $\gcd(a, b)$ divides v.*

Proof. Assume they survive. By Corollary 3, $\gcd(a, b)$ must divide v. Clearly v must also be within the size of the larger bin.

Now assume $\gcd(a, b)$ divides v and $0 \leq v \leq a$. If $v = 0$ Bruce wins right away and if $v = a$ Bruce wins after filling in the large jug. Otherwise, $0 < v < a$. By Lemma 8, $\gcd(a, b) = s \cdot b - t \cdot a$, so $v = k \cdot \gcd(a, b) = (ks) \cdot b - (kt) \cdot a$ for some integer k. Bruce and Sam survive by Lemma 7. □

We are almost done. All that remains is to prove Lemma 8. This can be done by analyzing the following algorithm, whose purpose it is to compute the numbers s and t.

Extended Euclid's algorithm $X(n, d)$, where n, d are integers such that $n \geq d \geq 0$.

```
    If $d = 0$, return $(1, 0)$.
    Otherwise,
        Write $n = qd + r$ where $0 \leq r < d$.
        Calculate $(s, t) = X(d, r)$.
        Return $(t, s - q \cdot t)$.
```

Theorem 10. *For every pair of integers n, d such that $n > d \geq 0$, $X(n, d)$ terminates and outputs a pair of integers (s, t) such that $\gcd(n, d) = s \cdot n + t \cdot d$.*

Instead of proving this theorem, let us show on an example how s and t can be calculated by extending the reasoning we used for Euclid's algorithm. Let $n = 73$ and $d = 50$. Euclid's algorithm goes through the following steps:

$$\begin{aligned} E(73,50) &= E(50,23) && \text{because } 73 = 1 \cdot 50 + 23 && \text{(E1)} \\ &= E(23,4) && \text{because } 50 = 2 \cdot 23 + 4 && \text{(E2)} \\ &= E(4,3) && \text{because } 23 = 5 \cdot 4 + 3 && \text{(E3)} \\ &= E(3,1) && \text{because } 4 = 1 \cdot 3 + 1 && \text{(E4)} \\ &= 1. \end{aligned}$$

We want to express 1 as an integer combination of 73 and 50. We start with the equation (E4), which writes 1 as an integer combination of 4 and 3 and work backwards. We take (E4) and (E3) and cancel out the 3 which occurs in both. To do this, we scale the (E3) by -1 on both sides and add (E4) to it to get

$$-1 \cdot 23 + 4 = -5 \cdot 4 + 1.$$

We can rewrite this as

$$-1 \cdot 23 = -6 \cdot 4 + 1, \tag{1}$$

which is a representation of 1 as an integer combination of 23 and 4. We can now cancel out the 4 by combining (E2) and (1): After multiplying (E2) by 6 on both sides and subtracting (1) we get

$$6 \cdot 50 - 1 \cdot 23 = 12 \cdot 23 + 1,$$

from where

$$6 \cdot 50 = 13 \cdot 23 + 1, \tag{2}$$

which is a representation of 1 as an integer combination of 50 and 23. Finally, the 23 disappears if we multiply (E1) by -13 and subtract (2) to obtain

$$-13 \cdot 73 + 6 \cdot 50 = -13 \cdot 50 + 1$$

which gives us the desired integer combination

$$1 = -13 \cdot 73 + 19 \cdot 50.$$

Lemma 8 is almost the same as Theorem 10. The only difference is that Lemma 8 says that s and t are non-negative, while Theorem 10 says nothing about signs. In this respect, Lemma 8 is *more general* than Theorem 10: It provides a stronger conclusion under the same assumptions. So how can we use Theorem 10 to prove Lemma 8?

As usual, it helps to work through an example first. Suppose we want to write $4 = s \cdot 3 - t \cdot 5$ for non-negative s and t, but all we have is a representation with the wrong sign pattern, say

$$4 = -2 \cdot 3 + 2 \cdot 5.$$

What can we do to correct the signs? We can add $5 \cdot 3$ to the first term and subtract $3 \cdot 5$ from the second term. Then

$$4 = (-2 + 5) \cdot 3 + (2 - 3) \cdot 5 = 3 \cdot 3 - 1 \cdot 5$$

which is exactly what we wanted. In general, this may not happen after one step but we can keep repeating until we get what we want.

Proof of Lemma 8. By Theorem 10, there exist integers s and t such that $\gcd(a,b) = s \cdot a + t \cdot b$. Then for every integer k,

$$(s + kb) \cdot a + (t - ka) \cdot b = sa + tb = \gcd(a,b).$$

No matter what the values of s and t are, when k is sufficiently large, $s + kb$ is positive and $t - ka$ is negative. This gives us a representation of $\gcd(a,b)$ of the desired form. □

4 Prime decomposition

A number $p > 1$ is a *prime* if it has no divisors between 2 and $p - 1$. We all know that

Theorem 11. *Every number greater than* 1 *has a unique prime factorization (in increasing order).*

For example, $18 = 2 \cdot 3 \cdot 3$ (existence) and there is no other way to write 18 as a product of primes. Why is this always true?

The existence of prime factorizations is a consequence of strong induction.

Proof of existence of factorizations. The base case is that 2 has a prime factorization. Now assume all numbers between 1 and $n - 1$ have a prime factorization. There are two cases for n. Either it is prime and its prime factorization is itself, or it has a factor a between 2 and $n - 1$. The other factor n/a is also between 2 and $n - 1$ so both a and b have a prime factorization. Their product gives a prime factorization of n. □

Uniqueness of prime factorization is trickier. In fact, there are alternative number systems in algebra where prime factorizations are not unique. The integers are special in this way.

Let's try to understand why the same number cannot be represented by different prime factorizations. The idea is to argue by contradiction. Suppose for example that some number factors both as $2 \cdot 3 \cdot 3 \cdot 7$ and as $3 \cdot 5 \cdot 7$. Then the two factorizations must be equal:

$$2 \cdot 3 \cdot 7 = 3 \cdot 3 \cdot 5.$$

Now we can cancel a 3 and a 7 from both sides to obtain

$$2 \cdot 7 = 3 \cdot 5.$$

This is clearly absurd, but why? It has to do with divisibility: 2 divides the number on the left but not on the right. How do we know that 2 does not divide $3 \cdot 5$? This turns out to be a key property of prime numbers: If 2 does not divide 3 and 2 does not divide 5, 2 cannot divide their product $3 \cdot 5$. In contrapositive form, this says

Lemma 12. *If p is a prime that divides ab, then p divides a* or *p divides b.*

Proof. Assume that p divides ab but p does not divide a. We will show that p must divide b. Since p does not divide a their only common divisor is 1, so $\gcd(p, a) = 1$. By Lemma 8, 1 is a combination of p and a:

$$1 = s \cdot p + t \cdot a \qquad \text{for some } s, t.$$

Multiplying both sides of this combination by b we get that b is a combination of p and ab:

$$b = sb \cdot p + t \cdot ab$$

Since p divides ab, b must be a multiple of p:

$$ab = kp \qquad \text{for some } k\text{, so} \qquad b = sb \cdot p + tk \cdot p = (sb + tk) \cdot p.$$ □

We now have all the ingredients to prove uniqueness of factorizations.

Proof of uniqueness of factorizations. Suppose for contradiction that some number has two factorizations A and B. Then one of the prime factors p must appear more times in one factorization than in the other. After possibly removing some copies of p from A and B we obtain two factorizations A' and B' of the same number one of which contains a copy of p but the other one doesn't. Suppose A' is the one that contains the copy of p. Then p divides A' so p must divide B'. By Lemma 12, p must divide some prime that appear in B'. As p is a prime that does not appear in B' it cannot divide any of the primes that appear in B'. This is a contradiction. □

Prime factorizations provide an alternative method for calculating GCDs. For example, if we want to calculate the GCD of 12 and 18, we can write out the factorizations $12 = 2 \cdot 2 \cdot 3$ and $18 = 2 \cdot 3 \cdot 3$ and calculate the GCD as the product $2 \cdot 3 = 6$ of the common factors 2 and 3 that appear in both. In general, if the prime factorization of a contains a_2 copies of 2, a_3 copies of 3, a_5 copies of 5, a_7 copies of 7, and so on, and the factorization of b contains b_2 copies of 2, b_3 copies of 3, b_5 copies of 5, and so on (zero copies are allowed), then the factorization of $\gcd(a, b)$ will have $\min\{a_2, b_2\}$ copies of 2, $\min\{a_3, b_3\}$ copies of 3, $\min\{a_5, b_5\}$ copies of 5, and so on. To obtain the GCD we just need to multiply those factors together:

Schoolbook GCD algorithm: Given integers $a, b > 0$,

```
    Factor $a$ as $2^{a_2} \cdot 3^{a_3} \cdot 5^{a_5} \cdot 7^{a_7} \cdots$
    Factor $b$ as $2^{b_2} \cdot 3^{b_3} \cdot 5^{b_5} \cdot 7^{b_7} \cdots$
    Calculate $c_i = \min\{a_i, b_i\}$ for all nonzero $a_i$ and $b_i$
    Output $2^{c_2} \cdot 3^{c_3} \cdot 5^{c_5} \cdot 7^{c_7} \cdots$
```

This is indeed the way we learned to calculate GCDs in school. So why bother with Euclid's algorithm? The reason is that although both the Schoolbook GCD and Euclid's algorithm are correct, Euclid's is much faster when the numbers involved are large. The bottleneck is the factorization of a and b: Nobody knows a good way to factor numbers with hundreds of digits even on state-of-the-art computers. (One reason there is much interest in *quantum computers* is that they are believed to possess this capability, but noone has managed to build one yet.) In contrast, Euclid's algorithm has no trouble at all even if a and b has millions of digits. This inability to factor large numbers even on supercomputers turns out to be crucial for the security of the RSA data encryption scheme which we will talk about in the next lecture.

References

This lecture is based on Chapter 8 of the text *Mathematics for Computer Science* by E. Lehman, T. Leighton, and A. Meyer.