1 Die hard, once and for all

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?

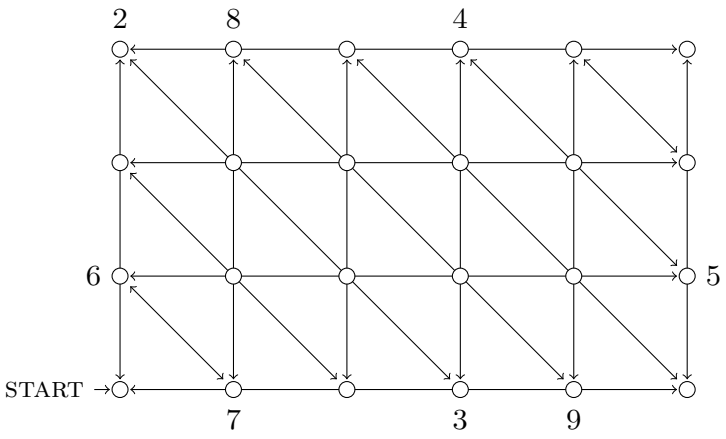
This scenario is straight out of the 1995 classic "Die Hard 3: With a Vengeance" featuring Bruce Willis and Samuel L. Jackson. Should they fail to complete this task within 5 minutes, a bomb goes off and New York City is obliterated. In the nick of time, Bruce and Samuel come up with a solution:

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!

The rumour is that in Die Hard 4, Bruce and Samuel will be given a 21 litre and a 26 litre jug; in Die Hard 5, a 899 litre and 1,147 litre jug; and in Die Hard 6: "Die hard, once and for all", a 6 litre and 9 litre jug. What should they do?

The Die Hard state machine The water jug game can be described 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 allowed pourings. The initial state is $(0, 0)$, and we are interested in reaching a state of the form $(4, y)$.

The Die Hard 3 state machine is small enough that we can represent it with a diagram. 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 need to describe the state machine mathematically. If a and b are the jug capacities the possible states are the pairs

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

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 out jug now contains 0 litres, while the contents of the other jug stay the same. Symbolically,

$$(x, y) \to (0, y) \tag{T1}$$
$$(x, y) \to (x, 0) \tag{T2}$$

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. Symbolically,

$$(x, y) \to (a, y) \tag{T3}$$
$$(x, y) \to (x, b) \tag{T4}$$

3. **Transfer water between jugs:** Water can be poured from one jug into the other until one of them becomes full or empty. These transitions are a bit 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, in which case 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 a bit 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 other movies?

If you repeat the experiment with the Die Hard 6 state machine you notice 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 combination.

Integer combinations

A number c is an *integer 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 an integer combination of 4 and 6 because $2 = (-1) \cdot 4 + 1 \cdot 6$.

- 1 is an integer combination of 3 and 5 because $1 = 2 \cdot 3 + (-1) \cdot 5$.
- 2 is an integer 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 an integer 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 integer combinations of a and b, then any integer combination of x and y is also an integer combination of a and b.*

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

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

and 3 is an integer combination of 6 and 9:

$$3 = -6 + 9.$$

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

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

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

Proof of Lemma 1. If x and y are integer 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 integer 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 an integer 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 integer 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, both of x' and y' are some integer combination of x, y, a, b. By Lemma 1, x' and y' are therefore integer combinations of a and b. □

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.

Lemma 2. *Every common divisor of a and b also divides all integer combinations of a and b.*

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

If d is a common divisor of a and b then a and b are integer combinations of d and zero. By Lemma 1 so is any integer combination of a and b, so d divides this integer 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, $\gcd(2, 6) = 2$, $\gcd(4, 6) = 2$, $\gcd(3, 5) = 1$.

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 dies.*

2 Euclid's algorithm

Euclid's algorithm is a procedure for calculating the GCD of two positive integers. It was invented by the Euclideans around 3,000 years ago and it still the fastest procedure for calculating GCDs. 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*

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

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 initial state is the input to the algorithm. 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* $\gcd(n, d)$.

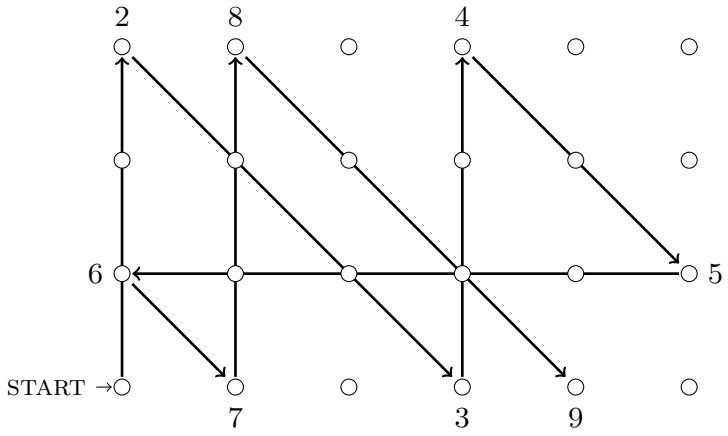
Proof. For termination, we observe that the value of d decreases in each transition but remains non-negative because d is replaced by the remainder r which is always smaller than d. Therefore the algorithm must terminate after a finite number of steps.

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 be a star in any Die Hard movie

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. Notice the curious zigzagging pattern in the transitions 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 out 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
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	$7 \to 8 \to 9$	$4 = 3 \cdot 3 - 1 \cdot 5$

Notice that 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 smaller 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 at most $v - a$ litres which is negative, while if $t' < t$ then jug A would contain at least $v + a$ litres, so it would overfill. Both cases give a contradiction. It follows that 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 make sure that the target v can be written as $s \cdot b - t \cdot a$ and the algorithm will do the rest of the work. We know that this is impossible unless v happens to a multiple of $\gcd(a, b)$. It happens that 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 now have a complete understanding of the Die Hard state machine:

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

Proof. Assume Bruce lives. 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 automatically, 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 lives 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:

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