Cutting Sticks: Solution

We present two algorithms for cutting the sticks.

Greedy Algorithm. Greedy algorithms try to get as close to an objective as possible at each step. A sufficiently greedy chess AI, for example, would fail to sacrifice a piece even if sacrificing would have meant winning the next turn, since it measures closeness to winning by material on the board and will thus choose to avoid losing pieces whenever possible.

We can try to maximize how much we cut from one of the sticks. Starting with the stick of length a, cut out the largest value from $1, 2 \cdots, n$ possible that hasn't already been cut from it, and continue doing so as long as possible.

Suppose this process terminated with some stick left over. If at any stage we didn't cut the maximum unused length from the set $\{1, \cdots, n\}$, then (because our original process had leftover stick) we could have increased the cut at that stage without problem, contradicting the assumption we chose lengths greedily. Thus, we must have cut lengths n, $n-1$, $n-2$, $\cdots$, and if our last cut wasn't 1 then the process didn't terminate because at the end we can now cut 1 out. But this means we cut the total net length $n + \cdots + 2 + 1$ out of the first stick, so there can't be any leftover length, a contradiction.

The sum total of all the lengths from $1, 2, \cdots, n$ that weren't cut from the first stick must equal the length of the second stick. Therefore, these lengths can be cut out of the second stick in any order (e.g. greedily) to finish.

Notice how this algorithm does not require the hypothesis $a, b \geq n$. However, the condition is necessary for a generalization: if we have sticks of lengths $\ell_1, \cdots, \ell_k \geq n$ with sum total length $\ell_1 + \cdots + \ell_k = 1 + 2 + \cdots + n$ then we can cut them into lengths $1, 2, \cdots, n$. The greedy algorithm doesn't work for this generalization. Indeed, whether or not this assertion is true is an unsolved question - the **cutting sticks conjecture**!

The following recursive algorithm, on the other hand, does work for the generalization for all n but for only finitely many exceptions for any given number of sticks k (though the number of exceptions grows with k).

Recursive Algorithm. A recursive algorithm solves a problem by using itself on a subproblem at each step. For example, the recursive algorithm computing $1 + 2 + \cdots + n$ would compute $1 + 2 + \cdots + (n - 1)$ then add n.

In the sticks problem, both lengths are $a, b \geq n$. If we cut out a length of n from the longer stick (suppose $a \geq b$), then either the remaining stick length $a-n$ is $\geq n-1$ or else $< n-1$. In the former case, we've reduced the problem for minimal length of n to the same problem for a minimal length $n - 1$, so our recursive algorithm can use itself on this subproblem.

The cases where we can't use this algorithm are when the remaining stick length is $a - n < n - 1$. In these cases, $b \leq a < 2n - 1$ implies an upper bound on the total length $1 + 2 + \cdots + n = a + b \leq 2a < 2(2n - 1)$. It is well-known the so-called triangular number $1 + 2 + \cdots + n$ is given by the formula $\frac{n(n+1)}{2}$, so our bound is $\frac{n(n+1)}{2} < 2(2n - 1)$. Clearing denominators and distributing makes this $n^2 + n < 8n - 4$, or equivalently $n(n - 7) < -4$, which is only possible for $n < 7$. Thus for $n = 3, 4, 5, 6$ we manually find and list a solution for the cases where $a < 2n - 1$:

n	$\frac{n(n+1)}{2}$	$2n-1$	$a \rightarrow$	$\cdots$	$b \rightarrow$	$\cdots$
6	21	11				
5	15	9	$8\rightarrow$	5+3	$7\rightarrow$	4+2+1
4	10	7	$6\rightarrow$	3+2+1	$4\rightarrow$	4
			$5\rightarrow$	4+1	$5\rightarrow$	3+2
3	6	5	$4\rightarrow$	3+1	$2\rightarrow$	2
			$3\rightarrow$	2+1	$3\rightarrow$	3