Algorithms

Kevin Li

April 2022

1 Revisiting the Euclidean Algorithm

Recall the Euclidean algorithm from last quarter:
Let a, b be natural numbers with $a > b$. Using the division algorithm, we have the following steps:

$$a = q_0 b + r_0$$

$$b = q_1 r_0 + r_1$$

$$r_0 = q_2 r_1 + r_2$$

$$r_1 = q_3 r_2 + r_3$$

$$\vdots$$

$$r_{n-2} = q_n r_{n-1} + r_n$$

This process continues until $r_n = 0$ in which case r_{n-1} is $gcd(a, b)$. (Remember, at each step $0 \leq r_{i+1} < r_i$)

Problem 1 *Why is this algorithm guaranteed to stop? Can you give an upper bound on the number of steps it is guaranteed to stop in? (Hint: It is in terms of a, b)*

Problem 2 *Do you think this is a good upper bound? Why or why not?*

Problem 3 *Let's try to do a little better. Consider the terms $r_0, r_1, \ldots, r_n$. Show that after every 2 steps, have that*

$$r_{i+2} < \frac{r_i}{2} \quad \forall i \in \{1, 2, \ldots, n\}$$

1. Look at r_{i+1}. What happens if $r_{i+1} \leq \frac{1}{2} r_i$?

2. Now, suppose $r_{i+1} > \frac{1}{2} r_i$. How can we use the division algorithm on r_i and r_{i+1} here? (Hint: How many times can r_{i+1} go into r_i?)

Problem 4 *Using Problem 3, show that after* $2n$ *steps the remainder has been reduced by at least a factor of* $\frac{1}{2^n}$. *What happens when some* $r_i < 1$*? Conclude that the Euclidean Algorithm takes at most* $2log_2(b)$ *steps.*

Problem 5 *What is an approximate way to get the number of digits in a number using logarithm? Show that the Euclidean Algorithm takes at most (*$7 *$ number of digits of b) steps. You may use a calculator for this problem.*

2 Algorithms and Runtime

Definition 1 *Suppose* $f, g : \mathbb{N} \to \mathbb{R}$

1. *A function* f *is said to be in Big-Oh of* g*, denoted* $f \in \mathcal{O}(g)$ *or* $f = \mathcal{O}(g)$*, if there exists a constant* $c \in \mathbb{R}_{>0}$ *and* $N \in \mathbb{N}$ *such that* $f(x) \leq cg(x)$ *for all* $x \geq N$

2. *A function* f *is said to be in Big-Oh of* g*, denoted* $f \in \Omega(g)$ *or* $f = \Omega(g)$*, if there exists a constant* $c \in \mathbb{R}_{>0}$ *and* $N \in \mathbb{N}$ *such that* $cg(x) \leq f(x)$ *for all* $x \geq N$

3. *A function* f *is said to be in Big-Theta of* g*, denoted* $f \in \Theta(g)$ *or* $f = \Theta(g)$*, if* $f \in \mathcal{O}(g)$ *and* $f \in \Omega(g)$

Problem 6 *Show that* $x = \Theta(3x)$

Problem 7 *Show that for any fixed* $a, b \in \mathbb{N}$*,* $a, b \geq 2$ *that* $log_a(n) = \Theta(log_b(n))$*. After we show this, we do not have to write a base for the logarithm. Instead, we will just write* $log(n)$*.*

Problem 8 *Let $f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ be non-negative functions. Let $a \in \mathbb{N}$. Show that if $f(x) = \mathcal{O}(g(x))$ then $f^a(x) = \mathcal{O}(g^a(x))$*

Problem 9 *Let $f, g, h : \mathbb{N} \to \mathbb{R}$. Suppose $f(x) = \mathcal{O}(g(x))$ and $g(x) = \mathcal{O}(h(x))$. Then, show that $f(x) = \mathcal{O}(h(x))$*

Problem 10 *Show that* $log(n!) = \Theta(nlog(n))$. *(For one direction, use the log product rule. For the other, consider the function* $\frac{n}{2} - \sqrt{n}$ *for* $n \geq 4$ *and show it is increasing. Then, conclude* $n! \geq n^{\frac{n}{4}}$*).*

Problem 11 *Let* $p(x)$ *be any polynomial. Can we have* $n^{log(n)} = \mathcal{O}(p(x))$*? Why or why not? (CHALLENGE: Give a formal proof)*

It is hard to formalize what an algorithm is mathematically. If you have any computer science or programming background, you can think of an algorithm as a piece of code to be executed.

Definition 2 *The **time complexity/running time** of an algorithm is a function $f(x)$ such that the **worst case** number of steps of an input of size n is in $\mathcal{O}(f(n))$. Assume basic operations (arithmetic, deletion, comparing, retrieving, etc.).*

Example 1 *In the Euclidean Algorithm, our inputs are considered a, b. Remember that the crude estimate for the maximum number of steps the algorithm finishes in was b. Therefore, we can consider being "size n" in this context to mean "n is the value of the input b" i.e. $n = b$.*

Example 2 *Let's consider a list of integers $[x_1, x_2, \ldots, x_n]$. This is considered an input if size n since there are n numbers in the list. Notice this does not depend on the value of the integers, just how many there are. Now, let's think of an algorithm to sort this list in **ascending** order (from least to greatest). One way you may try to do this is: "Look at the list. Take the least element, put it in the front. Then find the second least element, it in the second spot,..." until the list is sorted. However, we have to think like a computer, not a human. Think about what you have to do to find the minimum element in a list. You have to look at the first element, then compare it to the second element, then the third, etc. A computer considers each of these **one step**.*

Problem 12 *Give a function $f : \mathbb{N} \to \mathbb{R}$ that describes the 3 running times of the Euclidean algorithm at the beginning of the worksheet. Which of the running times would you consider to be "better" than the rest? What does it mean for an algorithm to be better than another?*

Problem 13 *Consider the following algorithm: Take input $n \in \mathbb{N}$. At each iteration, subtract 1 from n and and make another copy of the existing value. Stop when we reach all 0's. (For example: start with 4. The first iteration gives 3, 3. The second iteration gives 2, 2, 2, 2. etc.) What is the time complexity of this algorithm? What if we subtracted 2 instead of 1? What if we subtract 1 but make 2 copies instead of 1?*

Problem 14 *Design an algorithm which sorts a list of length n, $[x_1, x_2, \ldots, x_n]$. Try to give a function which describes the running-time of your list. We can assume retrieving an element in a spot in the list, comparing 2 elements, deleting elements, adding elements, switching elements, etc. all take 1 step (or equivalently these all have running time of $\mathcal{O}(1)$. Work together in your group and build off of each other's ideas. This should be collaborative.*

Problem 15 ***CHALLENGE:*** *Design an algorithm to sort a list of length n,* $[x_1, x_2, \ldots, x_n]$