Evaluating the Cost of an Integer Using Primitive Recursive Functions

MAA EPaDel, Fall 2023 Section Meeting - Villanova University

John Seibert

Bloomsburg University of Pennsylvania

November 11, 2023

Today's Itinerary

1. Constructing $C_s(n)$ and its Computability
   - Definition of $C_s(n)$
   - Computability of the Cost Function
2. Python Program Demonstration
3. Fun Results
   - Upper Bound of $C_s(n)$
   - Given s only consists of multiplication and binary, why does binary always get output given n $>$ 16?

We assume $C_s(n)$ is constructed to have s consist of multiplication, the Fibonacci sequence, successor, addition, exponentiation, and the binary representation of a given integer n unless stated otherwise. This will be further explained later.

Definition of $C_s(n)$

We start off with a definition of $C_s(n)$.

Definition

Let n be a non-negative integer. Let s consist of a finite set of primitive recursive functions fed into $C_s(n)$. Then we establish $C_s(0) = 1$ and $C_s(1) = 1$. For n > 1, $C_s(n)$ is calculated as the minimum output of any n-ary function $C_s(n)$ is fed.

Each of function fed into $C_s(n)$ is calculated through recursion. Ties are broken through which function fed obtains the minimum first.

Formulae for Functions fed into $C_s(n)$

Let a, b, ... be integers used to calculate the n-ary function $C_s(n)$ for a given n > 1.

- Multiplication: $C_s(a) + C_s(b) + 3$
- Fibonacci: $C_s(a) + C_s(b) + 4$
- Successor: $C_s(a) + 1$
- Addition: $C_s(a) + C_s(b) + 2$
- Exponentiation: $C_s(a) + C_s(b) + 4$
- Binary: $\lfloor \log_2(n) \rfloor + 8$ (non-powers of 2), $\log_2(n) + 8$ (else)

For each binary operation, we are looking for two numbers that perform its respective operation such that the two numbers equal *n*. (The successor function can be thought of as adding 1, n times.)

Computability of $C_s(n)$

Lemma

For any set of primitive recursive functions S and any positive integer m, $C_S(m) \leq m$.

Since the operators of primitive recursion preserve computability, the set of all primitive recursive functions is a subclass of the class of all computable functions. Hence, the cost function is computable.

The value of $C_s(m)$ proceeds straight from the cost definition.

Program Demonstration

We will now show, for practical purposes, this function using a Python program. We show a given n, $C_s(n)$, and the method of obtaining $C_s(n)$ using this website.

The program can be viewed here. Here are a couple highlights for those curious:

- The cost of a given n $\geq$ 12 will not equal itself.
- Binary can produce the cost as early as when n $=$ 14, but it must come as late as n $=$ 19.
- The output of $C_s(n)$ can go up only by 1.

Obtaining the Upper Bound

Theorem (Upper Bound)

Let s consist of multiplication, Fibonacci, successor, exponentiation, addition, and binary. Then $C_s(n) \leq \lfloor \log_2(\mathrm{n}) \rfloor + 8$.

By way of contradiction, suppose $C_s(n) > \lfloor \log_2(\mathrm{n}) \rfloor + 8$. Then consider s to just consist of binary. The value of $C_s(n)$ will be exactly equal to $\log_2(n) + 8$ when n is a perfect square and is equal to $\lfloor \log_2(n) \rfloor + 8$ otherwise. Hence, we reach a contradiction. So $C_s(n) \leq \lfloor \log_2(\mathrm{n}) \rfloor + 8$.

A Result on a Smaller Set of s

Theorem

Let s consist of binary and multiplication in that order. Then binary will always produce $C_s(n)$ for a given n > 16.

Base Case: Given n $= 17$, if we treat s to only consist of binary $C_s(n) = 12$. Should s consist of just multiplication, $C_s(n) = 17$ since $C_s(1) + 17$ exceeds 17. This proves the base case.

Inductive Hypothesis: We will assume the pattern of $C_s(n)$ when s solely consists of binary $\leq C_s(n)$ when s consists of solely multiplication holds up to a given n - 1. In other words, binary will produce the cost since it is either the sole minimum or wins the tie due to it being first in order.

Proof Continued

WWTS this pattern continues to hold for any given n.

Case 1: n is a power of 2. Let m $\in \mathrm{Z}^+$, such that n $= 2^{\mathrm{m}}$. Since m $= \log_2(n)$, then $C_s(n)$ when s is solely binary is m $+ 8$. Contrast this to $C_s(n)$ when s consists of just multiplication, and that is at least 2m $+ 3$, which is when n would be a perfect square. Since m $+ 8 \leq$ 2m $+ 3$. Then $5 \leq$ m. So all powers of 2 are covered.

Proof Continued

Case 2: n is not a power of 2. Let p, q $\in Z^+$ such that it factorizes n to produce the minimum cost for factorizing a given n. In other words, n = p * q. Then we also know that $\lfloor \log_2(n) \rfloor + 1$ is equal to the length of the binary representation of n. Laws of logarithms tell us that $\lfloor \log_2(n) \rfloor = \lfloor \log_2(p) + \log_2(q) \rfloor$. By definition of $C_*(n) = C(p) + C(q) + 3$. By definition of $C_B(n) = \lfloor \log_2(n) \rfloor + 8$. So, $\lfloor \log_2(p) + \log_2(q) \rfloor \leq C(p) + C(q) - 5$. We can simplify the LHS as $\lfloor \log_2(n) \rfloor + 1 \leq C(p) + C(q) - 4$. Eventually, the cost of multiplication will grow to a point where the cost of one factor will exceed the cost of binary. This is because the cost of multiplication has a slower runtime complexity in worst-case scenario, $O(\sqrt{n})$, than binary, O(log(n)), as n increases. Since the hypothesis assumes the pattern holds for a given n - 1, the pattern must hold for n as well. $\square$

Future Directions

- Develop (and prove) a lower bound for $C_s(n)$
- Show that without accounting for the cost of the operations, if n has a prime factorization of $2^a * 3^b * 5^c$, then addition would never lower the cost.
- Reconstruct the program accounting for induction on the values of $C_s(n)$.

Thank you!

Any questions?

Special acknowledgements:

- Bill Calhoun: creation of the original 2005 programming contest problem, *Cheap Integers* where this is derived from; mentorship during Summer 2023, when the research took place.

- Max Norfolk: contributions to the previous literature through the 2021 Rose-Hulman Undergraduate Mathematics Journal article *The Cost of a Positive Integer* where much of this work is derived from.