CS487 - Symbolic Computation

University of Waterloo
Nicholas Pun
Winter 2020

Contents

Lecture 1: Introduction

1.1 Course Preview

Example 1.1 (Simplyfying Rational Expressions)**.** Suppose we have the two following expressions:

$$f := \frac{x+1}{x-1} - \frac{x^3-2x+x^2+2}{x^3+2x-x^2-2} + \frac{x^2+3}{x-1} \tag{1.1}$$

$$g := \frac{(x-1)^2 - x^2 - x + 2x}{(x+y+2)^{100}} \tag{1.2}$$

Question: How do we simplify these expressions to a single $\frac{poly}{poly}$ or return that it is 0?

One idea: Define a "normal" function:

1. If expression is 0, the normal function will be 0
2. If not, the normal function will be the simplest form

(More) Questions: What else do we need to consider?

- How do we represent polynomials (i.e. What data structure do we use?)
- How do we perform polynomial operations computationally?
- Do we need to consider the size of the integers in our computations?

Example 1.2 (Solving Recurrences)**.** Suppose we have the recurrence:

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + \frac{n}{2} & n > 1 \\ 1 & n = 1 \end{cases} \tag{1.3}$$

We can solve this by hand (using Master theorem or other techniques) to obtain the answer:

$$T(n) = n(1 + \log_2(n)) \tag{1.4}$$

Question: How do we do this computationally?

Example 1.3. Consider the following identities:

$$\sum_{k=0}^{n} k = \frac{n(n-1)}{2} \tag{1.5}$$

$$\sum_{k=0}^{n} k^4 = \frac{n(n-1)(2n-1)(3n^3-3n-1)}{30} \tag{1.6}$$

Question: Can we return a closed form (without involving the index k) for any general expression or report that one doesn't exist?

1.2 Representation of Integers

Current computers are based on architecture with 64 bits (We will call this number of bits the word size)

Example 1.4. The unsigned long in C represents integers in exactly the range $[0, 2^{64} - 1]$

Question: How do we represent larger numbers?

Idea: Use an array of word size numbers.

Any integer a can be expressed as the following summation:

$$a = (-1)^s \sum_{i=0}^{n} a_i 2^{64i} \tag{1.7}$$

where $s \in \{0, 1\}$ represents the sign of a and $0 \leq a_i \leq 2^{64} - 1$ are the individual elements in the array.

If we assume $0 \leq n + 1 \leq 2^{63}$, then we can encode a as an array:

$$[s \cdot 2^{63} + n + 1, a_0, a_1, \ldots, a_n] \tag{1.8}$$

This is sufficient for all practical purposes.

Note. The length of a is given by: $\lfloor \log_{2^{64}} |a| \rfloor + 1 \in \mathcal{O}(\log |a|)$ words

Addition of Integers:

Suppose our input is $a : a_0 + a_1\beta + a_2\beta + \ldots a_n\beta_n$ and $b : b_0 + b_1\beta + b_2\beta + \ldots b_m\beta_m$ (where $m \leq n$).

Question: How large can each individual nu

Lecture 2: