Outlab 2: LaTeX and gdb

P. Balasubramaniam and Shubh Kumar

Autumn 2021

Contents

1 Basic Theory of Linear Recurrences **2**
1.1 Computing terms of an LRS 2
1.2 Problems associated with LRS 4

2 Programming **5**

3 Hints and Tips **6**

1 Basic Theory of Linear Recurrences

We all know about the Fibonacci Sequence, given by the recurrence

$$f_{n+2} = f_{n+1} + f_n$$

and initialised as $f_0 = 0$, $f_1 = 1$. This is an instance of what is called a Linear Recurrence Sequence, or LRS.

Definition 1. *LRS $(u_n)_{n=0}^{\infty}$ of order k over field $\mathbb{F}$ is the sequence determined by the recurrence*

$$u_{n+k} = \sum_{j=0}^{k-1} a_j u_{n+j} \tag{1}$$

for integers $n \geq 0$ with $a_0, \ldots, a_{k-1} \in \mathbb{F}$, $a_0 \neq 0$, and an initialization vector

$$\begin{bmatrix} u_0 & \ldots & u_{k-1} \end{bmatrix}^{\mathsf{T}} \in \mathbb{F}^{k \times 1}$$

Thanks to Definition 1, when we say that we are given $(u_n)_{n=0}^{\infty}$ of order k over $\mathbb{F}$, we actually mean that we are given the $2k$ constants $a_0, \ldots a_{k-1}, u_0, \ldots, u_{k-1} \in \mathbb{F}$.

The Fibonacci sequence shows up a lot in pop culture and DSA assignments, however, the study of LRS is a vast subject in its own right, with applications in software verification, quantum computing, formal languages, statistical physics, combinatorics, and theoretical biology, to name a few.

1.1 Computing terms of an LRS

For a fixed k, assuming we can do arithmetic operations in $\mathbb{F}$ in constant time, how long does it take to compute the n^{th} term of a given LRS?

Algorithm 1: Naive first attempt naive

Data: LRS $(u_n)_{n=0}^{\infty}$, n
Result: u_n

```
1 if n < k then
2 |   return u_n
3 end
4 return Σ_{j=0}^{k-1} a_j · naive((u_n)_{n=0}^∞, n − k + j)
```

This algorithm, unfortunately, is painfully inefficient: as you can see, the number of recursive calls will be exponentially many in n. This immediately prompts a more "bottom-up" [1] approach: we start from the first k terms, use them to compute the next term, and iterate this way upto n. This takes $\mathcal{O}(n)$ iterations.

But we can do better, in fact, we only need $\mathcal{O}(\log n)$ iterations [2].

Technically, even Algorithm 2 is *inefficient*: when we analyse complexity, we do it with respect to the size of the *bit representation* of the input, i.e. how many bits it takes to specify the input. An integer n takes $\log n$ bits to represent, so the parameter for describing complexity is not n, but $\eta = \log n$.

[1] Not to be confused with the drinking phrase
[2] In complexity analysis, the base of the logarithm is implicitly taken as 2

Algorithm 2: Bottom up dynamic programming approach bottomup

Data: LRS $(u_n)_{n=0}^{\infty}$, n
Result: u_n
if $n < k$ **then**
  return u_n
end
circularArray $\leftarrow \{u_0, \dots u_{k-1}\}$
arrayStartIndex $\leftarrow 0$
$n_{last} \leftarrow k-1$
while $n_{last} < n$ **do**
  nextTerm $\leftarrow \sum_{j=0}^{k-1} a_j \cdot$ circularArray[(arrayStartIndex $+ j)\%k$]
  circularArray[arrayStartIndex] $\leftarrow$ nextTerm
  arrayStartIndex $\leftarrow$ (arrayStartIndex $+ 1)\%k$
  $n_{last} \leftarrow n_{last} + 1$
end
return circularArray[(arrayStartIndex $+ k - 1)\%k$]

Yes, we can compute the n^{th} term with $\mathcal{O}(\eta)$ operations, and the trick here is a method called **iterated squaring.** Given an LRS $(u_n)_{n=0}^{\infty}$, define its companion matrix $\mathbf{M} \in \mathbb{F}^{k \times k}$ as

$$\mathbf{M} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ a_0 & a_1 & a_2 & \dots & a_{k-1} \end{bmatrix} \tag{2}$$

Compare equations 1 and 2 and observe that

$$\mathbf{M}^n \begin{bmatrix} u_0 \\ \vdots \\ u_{k-1} \end{bmatrix} = \begin{bmatrix} u_n \\ \vdots \\ u_{n+k-1} \end{bmatrix} \tag{3}$$

Iterated squaring hinges on this ridiculously trivial observation: consider the (unique!) binary representation of n: if you append a 0 to it, it becomes $2n$, if you append a 1, it becomes $2n+1$.

Given $\mathbf{M}$, to compute $\mathbf{M}^n$, all we have to do is start with $\mathbf{I}$, read the binary representation of n from most to least significant: if at some point we have $\mathbf{M}^j, \mathbf{M}^j \cdot \mathbf{M}^j = \mathbf{M}^{2j}, \mathbf{M}^{2j} \cdot \mathbf{M} = \mathbf{M}^{2j+1}$

In Algorithm 3, we use a data structure called a **stack**. The constant time operations it supports are

- pushing an element onto the top of the stack
- popping an element from the top of the stack (i.e. deleting the topmost element)
- reading the topmost element
- checking whether the stack is empty

As you can see, it's Last In, First Out.

The division while loop gets the bits of n from least to most significant, the iterated square while loop uses the bits from most to least significant. A stack fits the bill.

Algorithm 3: Iterated squaring approach efficient

Data: LRS $(u_n)_{n=0}^{\infty}$, n
Result: u_n

```
quotient ← n
operationStack ← {}
M ← companion((u_n)_{n=0}^∞)
x ← [u_0 ... u_{k-1}]^T
A ← I_{k×k}
while quotient ≠ 0 do
    push(operationStack, quotient%2)
    quotient ← quotient/2
end
while operationStack ≠ {} do
    A ← A · A
    if top(operationStack) = 1 then
        A ← A · M
    end
    pop(operationStack)
end
y ← Ax
return y[0]
```

1.2 Problems associated with LRS

Surprisingly, the following rather simple *decision problem* [3], referred to as the Skolem problem, has been open for the last eight decades or so.

Definition 2. *(Skolem problem). For the arbitrary LRS $(u_n)_{n=0}^{\infty}$ whose description is given as input, does there exist an integer $n \geq 0$ such that $u_n = 0$?*

This has a bunch of equivalent formulations, and by being open, we mean that nobody really knows of an algorithm that can decide it (the algorithm should terminate and give the correct answer for *all* possible inputs), or whether there is such an algorithm at all.

We can also consider another related problem for LRS over fields where the $>$ operator is defined:

Definition 3. *(Positivity problem). For the arbitrary LRS $(u_n)_{n=0}^{\infty}$ whose description is given as input, is it the case that for all integers $n \geq 0$, $u_n \geq 0$?*

Deciding even special cases of these problems (i.e. restricting what kind of LRS can be fed as input) requires sophisticated maths, like Kronecker's theorem on Diophantine approximations, [2], [Chap. 7, Sec. 1.3, Prop. 7], and profound properties of "algebraic" numbers [3] and then some more mathematical arsenal [1, 4].

If you found this short write up interesting, here. is a survey talk about the problem, given not so long ago.

[3] Given an arbitrary input, answer a particular question about it with Yes or No

2 Programming

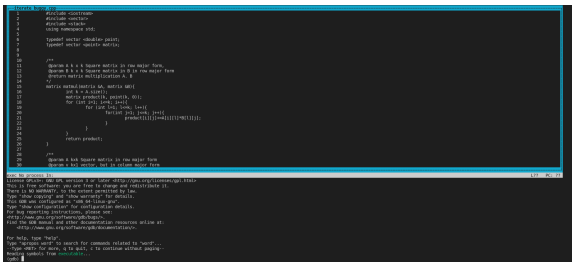

Figure 1: Screenshot 1:gdb interface

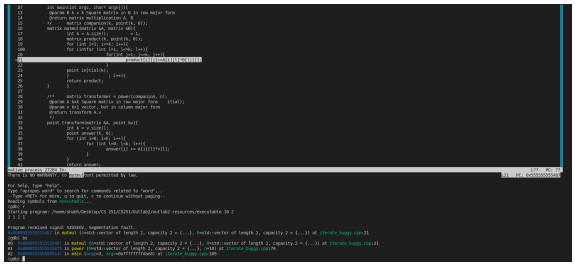

Figure 2: Screenshot 2:gdb backtrace

Here, are the snippets we had to complete :

```
(gdb) print working
```

```
$1 = std::vector of length 2, capacity 2 = {std::vector of
   length 2, capacity 2 = {1, 0}, std::vector of length 2,
   capacity 2 = {0, 1}}
```

```
(gdb) print operations
```

```
$2 = std::stack wrapping: std::deque with 9 elements = {0, 1,
    0, 1, 1, 0, 1, 1, 1}
```

operations	working
$\{0,1,0,1,1,0,1,1,1\}$	$\{1,0,0,1\}$
$\{0,1,0,1,1,0,1,1\}$	$\{0,1,1,1\}$
$\{0,1,0,1,1,0,1\}$	$\{1,2,2,3\}$
$\{0,1,0,1,1,0\}$	$\{8,13,13,21\}$
$\{0,1,0,1,1\}$	$\{233,377,377,610\}$
$\{0,1,0,1\}$	$\{317811,514229,514229,832040\}$
$\{0,1,0\}$	$\{591286729879,956722026041,956722026041,1548008755920\}$
$\{0,1\}$	$\{1.26e+24,2.04e+24,2.04e+24,3.31e+24\}$
$\{0\}$	$\{9.36e+48,1.51e+49,1.51e+49,2.45e+49\}$
$\{\}$	$\{3.17e+98,5.13e+98,5.13e+98,8.31e+98\}$

Table 1: operations vs. working

3 Hints and Tips

Listing 2: [LaTeX]TeX
Packages imported in the original

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[a4paper,hmargin=1.2in,bottom=1.5in]{geometry}
\usepackage[parfill]{parskip}
\usepackage{hyperref}
\usepackage{fancyhdr}
\usepackage{enumitem}
\usepackage{amsmath}
\usepackage{amsthm}
\usepackage{amssymb}
\usepackage[linesnumbered,ruled,vlined]{algorithm2e}
\usepackage{listings}
\usepackage{xcolor}
\usepackage{floatrow}
\usepackage{graphicx}
\bibliographystyle{plainurl}
\bibliography{biblio}
```

References

[1] J. P. Bell and S. Gerhold. On the positivity set of a linear recurrence. *Israel Jour. Math*, 57, 2007.

[2] N. Bourbaki. *Elements of Mathematics: General Topology (Part 2)*. Addison-Wesley, 1966.

[3] M. Mignotte. Some useful bounds. In *Computer Algebra*, 1982.

[4] James Renegar. On the computational complexity and geometry of the first-order theory of the reals, part i: Introduction. preliminaries. the geometry of semi-algebraic sets. the decision problem for the existential theory of the reals. *J. Symb. Comput.*, 13:255–300, 1992.

[5] Reference to Listing 2