Results on Linear Recurrences

Suvaditya Sur

suvaditya.sur@gmail.com

9 September 2020

Theorem 1. *Any recurrence of the form,*

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + c_3 a_{n-3} + \cdots + c_k a_{n-k}$$

has the solution,

$$\sum_{n=\sum\limits_{\substack{1\le i\le k\\ 1\le j\le c_i}} i*f_{ij}} \frac{\left(\sum\limits_{\substack{1\le i\le k\\ 1\le j\le c_i}} f_{ij}\right)!}{\prod\limits_{\substack{1\le i\le k\\ 1\le j\le c_i}} f_{ij}!} \tag{1}$$

where,

$$0 \le f_{ij} \le \left\lfloor \frac{n}{i} \right\rfloor$$
$$c_i \ge 0$$

The base cases can be defined as:

$$a_i = \begin{cases} 0, & \textit{if } i < 0 \\ 1, & \textit{if } i = 0, \sum\limits_{i,j} f_{ij} \ge 0 \\ 0, & \textit{if } i = 0, \sum\limits_{i,j} f_{ij} > 0 \\ 0, & \textit{if } 0 < i \le j-1 \\ c_j, & \textit{if } i = j \end{cases} \tag{2}$$

where, $c_i = 0$ *for* $0 < i \le j-1$ *and* $c_j > 0$

Proof. We can represent the recurrence as the number of ways to generate strings of length $= n$, using substrings s_{ij} of length $1, 2, \ldots, k$ where every substring of length i has a multiplicity c_i *i.e.*, there are c_i distinct substrings of length i.

We can represent the above linear recurrence as counting number of n length strings in the regular expression R^* or R^+ where

$$R = ((0_1+0_2+\cdots+0_{c_1})+(11_1+11_2+\cdots+11_{c_2})+\cdots+((k-1)^k_1+(k-1)^k_2+\cdots+(k-1)^k_{c_k}))$$

because taking 0_i contributes length $= 1$ to the string, thereby reducing to a_{n-1}, taking 11_i contributes length 2 and reducing the remaining length to a_{n-2}. The multiplicity c_i is handled by considering 0_i and 0_j as distinct strings of length 1, where $i \neq j$.

To generate strings of length n, we can, at a step take each symbol from the alphabet R. Let f_{ij} be the number of occurrences of the symbol whose length is i and $1 \leq j \leq c_i$.

This reduces to a problem of rearranging groups f_{ij} of distinct substrings to make a string of length n, the total number of groups is $N = \sum\limits_{i,j} f_{ij}$. Now for a given set of values for f_{ij} we have the number of n length strings as,

$$g(f_{ij}) = \binom{N}{f_{i1}}\binom{N-f_{11}}{f_{12}}\cdots\binom{f_{kc_k}}{f_{kc_k}} = \frac{N!}{\prod\limits_{i,j} f_{ij}!}$$

and since f_{ij} contributes length i to the string of length n, we have the constraints,

$$n = \sum_{1\leq i\leq k, 1\leq j\leq c_i} i * f_{ij}$$

Taking all possible solutions to the above constraint, we get the desired result.

Base Cases.

Since we are counting strings, the base cases will be defined on the minimum length string l_m accepted by R^+. If in

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + c_3 a_{n-3} + \cdots + c_k a_{n-k}$$

$c_i = 0$ for $0 < i <= j-1$ and $c_j > 0$, then the minimum length string accepted is of length j. If we consider R^+, we get

$$a_i = \begin{cases} 0, & \text{if } i < 0 \\ 0, & \text{if } i = 0 \\ 0, & \text{if } 0 < i <= j-1 \\ c_j, & \text{if } i = j \end{cases} \tag{3}$$

If we include ϵ, *i.e.,* R^*, the minimum length accepted is 1, and we get

$$a_i = \begin{cases} 0, & \text{if } i < 0 \\ 1, & \text{if } i = 0 \\ 0, & \text{if } 0 < i <= j-1 \\ c_j, & \text{if } i = j \end{cases} \tag{4}$$

$\square$

Example 0.1. Consider the example of fibonacci sequence

$$a_n = a_{n-1} + a_{n-2}$$

applying Theorem 1, we get,

$$\sum_{\substack{n=f_{i1}+2f_{i2} \\ 0 \le f_{i1} \le n \\ 0 \le f_{i2} \le \frac{n}{2}}} \frac{(f_{i1}+f_{i2})!}{f_{i1}!f_{i2}!}$$

simplifying a bit, we can reduce to one constraint instead of two,

$$\sum_{0 \le f_{i2} \le \frac{n}{2}} \frac{(n-2f_{i2}+f_{i2})!}{(n-2f_{i2})!f_{i2}!}$$

here's the snippet for programmers,

```
// combinations(n, r): computes nCr
int combinations(int n, int r)
{
  if (r > n || r < 0)
  {
    return 0;
  }
  int ans = 1;
  for (int i = 1; i <= r; i++)
  {
    ans = (ans*(n-i+1))/i;
  }
  return ans;
}

// return n'th fibonacci number using the defined theorem
int fibo(int n)
{
  int ans = 0;
  for (int i = 0; 2*i <= n; i++)
  {
    int j = n-2*i;
```

```
    ans += combinations(i+j, i);
  }
  return ans;
}
```