02 March 2023

MAT 2250

§ 5.2. Order of magnitude of (linear) recurrence relations

If $\{a_n\}_{n=1}^{\infty}$ satisfies a (linear) recurrence, what is the growth of $\{a_n\}$?

Recall the theorem:

Let $a_{n+d} = \alpha_1 a_{n+d-1} + \dots + \alpha_d a_n$ be a linear recurrence satisfied by a sequence $\{a_n\}_{n=1}^{\infty}$, where $\alpha_1, \dots, \alpha_d \in \mathbb{C}$.

Let $p(t) = (1-\gamma_1 t)^{m_1} \dots (1-\gamma_k t)^{m_k}$

$= 1 - \alpha_1 t - \alpha_2 t^2 - \dots - \alpha_d t^d$,

where $\gamma_1, \dots, \gamma_k \in \mathbb{C}$ are roots of $p(t)$ & with multiplicities $m_1, \dots, m_k$.

- Then for $A(z) = a_0 + a_1 z + a_2 z^2 + \dots$

$\Rightarrow A(z) = \sum_{i=1}^{k} \sum_{j=0}^{m_i} \frac{\beta_{ij}}{(1-\gamma_i z)^j}$

where $\beta_{ij} \in \mathbb{C}$.

- $\exists$ constants $\kappa_{ij} \in \mathbb{C}$ $i = 1, \dots, k$, $j = 0, \dots, m_i - 1$ such that

$$a_n = \sum_{i=1}^{k} \sum_{j=0}^{m_i - 1} \kappa_{ij} n^j \gamma_i^n \qquad \square$$

So we can get a growth estimate for a_n, using this.

We don't need to find the generating function $A(z)$ in order to find $\{a_n\}_{n=1}^{\infty}$.

Example: Let $a_0 = 0$, $a_1 = 1$, & $a_{n+2} = 4a_{n+1} - 4a_n \ \forall\, n \geq 0$

First, find a closed formula for $\{a_n\}$.

$a_{n+2} = 4a_{n+1} - 4$

$\Rightarrow p(t) = 1 - 4t + 4t^2$

$= (1-2t)^2$

So $\gamma_1 = 2$ with multiplicity 2

By the theorem, $\exists$ constants $\beta_1, \beta_2 \in \mathbb{C}$:

$a_n = \beta_1 n 2^n + \beta_2 2^n \ \forall\, n$.

We need to find β_1 & β_2. To this end we use the initial conditions: $a_0 = 0$, $a_1 = 1$.

$0 = a_0 = \beta_1 \times 0 \times 2^0 + \beta_2 \times 2^0$

$= 0 + \beta_2 \Rightarrow \beta_2 = 0$

$1 = a_1 = \beta_1 \times 1 \times 2^1 + \beta_2 \times 2^1$

$= 2\beta_1 + 0 \Rightarrow \beta_1 = \frac{1}{2}$

$\Rightarrow a_n = \frac{1}{2} n 2^n \quad \forall n$

2^n { $a_n = \Omega(n 2^n)$

$= O(n 2^n)$

$= \Theta(n 2^n)$.

Example: $\sim$ Same but now $\frac{1}{2} n 2^n + 1$ $a_{n+2} = 4a_{n+1} - 4a_n$, $a_0 = 1$, $a_1 = 2$.

Again, $\exists \beta_1, \beta_2$ constants such that $a_n = \beta_1 n 2^n + \beta_2 2^n \quad \forall n$.

To find β_1 & β_2 we use the initial conditions: $a_0 = 1$ & $a_1 = 2$.

$1 = a_0 = \beta_1 \times 0 \times 2^0 + \beta_2 \times 2^0$

$= \beta_2$

$2 = a_1 = \beta_1 \times 1 \times 2^1 + \beta_2 \times 2^1$

$= 2\beta_1 + 2\beta_2$

$= 2\beta_1 + 2$

$\Rightarrow 0 = 2\beta_1$

$\Rightarrow 0 = \beta_1$

$a_n = 2^n \quad \forall n$.

So $a_n = O(2^n)$

$= \Omega(2^n)$

$= \Theta(2^n)$

$\sim 2^n$

{ $n 2^n$

HW: The same $a_{n+2} = 4a_{n+1} - 4a_n$ with $a_0 = 2$, $a_1 = 3$. Compare a_n & 2^n & $n 2^n$. □

If the length of the linear recurrence is not constant, so what is the growth?

If $\exists$ constants $\alpha_1, \alpha_2, \ldots, \alpha_d$ so that for all n, we have relation: (*)

$a_{n+d} = \alpha_1 a_{n+d-1} + \cdots + \alpha_d a_n \quad \forall n$

then we call d the length of the linear recurrence (*).

One example where the length varies is:

$a_n = \alpha a_{\lfloor \frac{n}{b} \rfloor} + f(n)$,

$a_0 = 0, a_1 = c$, $b > 0$ an integer, $f(n)$ a function. $\lfloor \cdot \rfloor$ floor function

More specifically, if say $b = 3$ & $c = 1$, $f(n) = n$

$\lfloor 0.1 \rfloor = 0$

$\lfloor 1.9 \rfloor = 1$ ...

$a_0 = 0,\ a_1 = 1$

$a_2 = a_{\lfloor \frac{2}{3} \rfloor} + f(2)$
$= a_0 + 2 = 2$

$a_3 = a_{\lfloor \frac{3}{3} \rfloor} + f(3)$
$= a_1 + 3 = 4$

$a_4 = a_{\lfloor \frac{4}{3} \rfloor} + f(4)$
$= a_1 + 4 = 5$

$a_5 = a_{\lfloor \frac{5}{3} \rfloor} + f(5)$
$= a_1 + 5 = 6$

$a_6 = a_{\lfloor \frac{6}{3} \rfloor} + f(6)$
$= a_2 + 6 = 8$

Theorem 5.2: if $\alpha \geq 1$, $b > 1$, & $a_n = \alpha a_{\lfloor \frac{n}{b} \rfloor} + f(n)$.

a) if $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then $a_n = O(n^{\log_b a})$.

b) if $f(n) = O(n^{\log_b a})$ then $a_n = O(n^{\log_b a} \times \log n)$

c) If $f(n) = O(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, & $\alpha f(\frac{n}{b}) \leq c f(n)$ with $c < 1$ & $n \geq n_0$, then $a_n = \Theta(f(n))$ □

We won't prove the theorem, but here is some explanation:

The point is to compare the sequence $n^{\log_b \alpha} = n^{\mathcal{K}}$ with $\mathcal{K} = \log_b \alpha$ Why?

Let's say if $f(n) = 0\ \forall n$, what happens?

$a_n = \alpha a_{\lfloor \frac{n}{b} \rfloor}$

$a_b = \alpha a_1$

$a_{b^2} = \alpha a_{\lfloor \frac{b^2}{b} \rfloor} = \alpha a_b = \alpha^2 a_1$

By induction: if $n = b^k$ $\Rightarrow a_n = \alpha^k a_1 = \Theta(\alpha^k)$

Claim: $\alpha^k = n^{\log_b \alpha}$ ($n^{\log_b \alpha} = (b^{k \log_b \alpha}) = b^{k \log_b \alpha}$)

$n = b^k$

$\Rightarrow n^{\log_b \alpha} = b^{k \log_b \alpha}$
$= b^{(\log_b \alpha) k}$
$= (b^{\log_b \alpha})^k = \alpha^k$.

Here we showed that $a_n = \Theta(n^{\log_b \alpha})$ if $n = b^k$ for $k \in \mathbb{N}$.

HW: Show that we have similar estimates if $n \neq b^k$.

§5.3: Running times of algorithms

How to compute $n!$?

Algorithm:

$P = 1$

For $2 \le i \le n$:

$P = P * i$ (guess)

What is the complexity of this algorithm?

We need to do $n-1$ multiplications.

Complexity

We can say roughly that the complexity of an algorithm is the number of operations it needs to do from an input to get the desired output: plus, minus, multiplication, division, change two numbers, compare two numbers.

We want the complexity is "minimal" compared to the size of the input.

In this $n!$ question:

Input: a number n $\Rightarrow$ size $= \log_2 n$

Algorithm $\Rightarrow$ Complexity: $n-1$

Output: $n!$ $= O(n)$

In computers, we use binary system to store data. Example:

$11 = 1011_2$ in base 2

we need $\log_2 n$ bits to store the number n.

Problem: Find a good algorithm to multiply 2 matrices.

Why important? Most important applied mathematics question

NVIDIA chips / processors / GPU: Crucial in games, Deep learning (ChatGPT)

good multiplication method.

Even in theoretical study, also matrix multiplication is important.

Look on Wikipedia for Complexity of matrix multiplication.

Let's first consider the problem of multiplying 2 matrices of size 2×2.

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

$C = A \times B$:

Most simple algorithm:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj} \quad \forall i, j.$$

$n = 2$:

$c_{11} = a_{11} b_{11} + a_{12} b_{21}$ (2 multiplications, 1 addition)

$c_{12} = a_{11} b_{12} + a_{12} b_{22}$

$c_{21} = a_{21} b_{11} + a_{22} b_{21}$

$c_{22} = a_{21} b_{12} + a_{22} b_{22}$

$\Rightarrow$ Complexity = 8 (multiplications) + 4 (additions)

Usually, multiplication is more complicated than addition.

For general n:

$n^2 \times [$ n multiplications + $n-1$ additions $]$

Complexity $= O(n^3)$.

Strassen method (1968): improved the complexity can be $< n^3$.

$n=2$: $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$

$C = A \times B$

Define

$D_1 = (a_{11} + a_{22})(b_{11} + b_{22})$

• 1 multiplication + 2 additions

$D_2 = (a_{21} + a_{22}) b_{11}$

1 multiplication + 1 addition

$D_3 = a_{11}(b_{12} - b_{22})$

1 multiplication + 1 subtraction / addition

$D_4 = a_{22}(-b_{11} + b_{21})$

1 multiplication + 1 subtraction / addition

$D_5 = (a_{11} + a_{12}) b_{22}$

1 M + 1 A

$D_6 = (-a_{11} + a_{21})(b_{11} + b_{12})$

1 M + 2 A

$D_7 = (a_{12} - a_{22})(b_{21} + b_{22})$

1 M + 2 A

$C = A \times B$

$C_{11} = D_1 + D_4 - D_5 + D_7$

3 A

$C_{21} = D_2 + D_4$

1 A

$C_{12} = D_3 + D_5$

1 A

$C_{22} = D_1 + D_3 - D_2 + D_6$

3 A

Complexity: 7 Multiplications + 18 additions

For general n, we use the block method.

Example: Say $n = 4$

$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$

$= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

A_{ij}, B_{ij} is a 2×2 matrix.

But we can apply Strassen's method to $\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$

(think about them as 2×2 matrices).

Important remark: Since A_{11}, B_{11} are not numbers any more, be careful that $A_{11} \times B_{11} \neq B_{11} \times A_{11}$.

$$D_1 = (A_{11} + A_{22}) \leftarrow 2 \times 2 \text{ matrix}$$
$$\times (B_{11} + B_{22})$$
$$\vdots$$

So we can apply Strassen's method again to compute D_1 in this case.

Similar for $n = 2^k$.

Complexity $= O(n^{\log_2 7}) \approx O(n^{2.807})$

$\Rightarrow$ If n is not 2^k, add rows & columns of 0's to make it 2^k.