Linear Recurrence Equation in Analyzing the Complexity of Algorithms

Tang Van To
Faculty of Science and Technology
Assumption University, Bangkok
Thailand
tvto@scitech.au.edu

***Abstract*- Solving recurrence equation (Res) is an important technique in the analysis of algorithms. Especially for the divide and conquer algorithms, establishing the recurrence equations, solving them as well as finding the order of complexity will be discussed.**

***Keywords*- algorithm, Solving recurrence equation**

1. INTRODUCTION

Divide and Conquer method in general can be described as [1,2]

```
Divide-Conquer(S)
{ if ||S|| is small return DirectSolution(S)
  Else
  {Decompose S to smaller problems S1,S2...Sp;
  For i=1 to p
        Ri =Divide_and_Conquer (Si)

  Return R=Combine(R1,R2..Rp)

  }

}
```

Then the complexity of the algorithm can be written as

$$A(S)=\begin{cases} c \quad when \quad \|S\| \quad small \\ D(n)+\sum_{i=1}^{p} A(S_i)+C(n) \end{cases} \qquad (1)$$

Where n= ‖S‖ is the size of S, D(n), C(n) are time for decomposition S into smaller problems and time for combining subsolutions, respectively. Normally, these subproblems have the same size, let denote as n/b then (1) can be rewritten as

$$A(n)=\begin{cases} c \quad when \quad n \quad small \\ pA\left(\frac{n}{b}\right)+g(n) \end{cases} \qquad (2)$$

Here g(n)=D(n)+C(n).

2. LINEAR RECURRENCE (LIRE) EQUATION

Let $\{a_n\}$ be a sequence of numbers. A linear recurrence (RE) for $\{a_n\}$ has the form of

$$\sum_{i=0}^{k} c_i(n)a_{n-i} = r(n) \qquad (3)$$

If r(n)=0 then Eq. (3) is called homogenous and inhomogenous equation when r(n) ≠ 0. If $c_i(n)$ are constants then Eq. (1) is called a linear recurrence equation with constant coefficients (LIRE).

$$\sum_{i=0}^{k} c_i a_{n-i} = r(n) \qquad (4)$$

In this paper, we emphasize with r(n) has the form of

$$r(n)=\sum_{j=1}^{m} (b_i)^n P_i(n) \qquad (5)$$

Where $P_i(n)$ are polynomial of degree d_i on n.

Solving these inhomogenous linear recurrence equations with constant coefficients (ILIRE) using generating function [6] (GF) has been discussed in detail in [4,5]. The characteristic equation (CE) for ILIRE in Eq. (4) is

$$\left(\sum_{i=0}^{k} c_i x^{k-i}\right)\prod_{j=1}^{m}(x-b_i)^{d_i+1}=0 \qquad (6)$$

Let are roots of the CE with the multiplicities then the general solution (GS) for $\{a_n\}$ is

$$a_n = \sum_{i=1}^{q} \sum_{j=0}^{m_i - 1} c_{i,j} n^j r_i^n \quad (7)$$

Coefficients $c_{i,j}$ can be determined based on the initial values of a_i

3. SOME ILIRE EXAMPLES

3.1) Solving the following ILIRE

$$a_n = \begin{cases} n & n = 0,1 \\ 2a_{n-1} - a_{n-2} + 2^n n^2 & n \geq 2 \end{cases} \quad (8)$$

Move $2a_{n-1} - a_{n-2}$ to the LHS and find the CE for Eq. (8) as:

$$(x^2 - 2x + 1)(x - 2)^3 = 0 \quad (9)$$

CE in Eq. (9) has two roots of 1 and 2 with the multiplicities of 2 and 3 respectively.

Therefore GS for (8) is

$$a_n = c_1 + c_2 n + c_3 2^n + c_4 n 2^n + c_5 n^2 2^n$$

The order of a_n is $O(n^2 2^n)$. We can find the exact solution by replacing n=0,1…4 to find out all coefficients $c_1 \ldots c_5$. Another approach is replacing a_n, a_{n-1}, a_{n-2} to

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

And balancing two sides we get $c_5=4$, $c_4=-16$, $c_3=-32$. Other two values c_1 and c_2 are obtained by using a_0 and a_1. The exact solution for (8) is

$$a_n = -32 - 7n + 32x2^n - 16n2^n + 4n^2 2^n$$

3.2) Compute Integral Polynomial

Assume we want to compute

$$\sum_{i=0}^{n} P_k(i) \quad (10)$$

where $P_k(i)$ is a polynomial of degree k on i .

This summation is a polynomial Q of degree k+1 on n+1, the coefficients of Q can be obtained as [7]:

$$Q_{k+1}(n+1) = \sum_{i=0}^{n} P_k(i) \quad (11)$$

Change n to n-1 we get

$$Q_{k+1}(n) = \sum_{i=0}^{n-1} P_k(i) \quad (12)$$

Subtract (12) from (11), it gives

$$Q_{k+1}(n+1) - Q_{k+1}(n) = P_k(n) \quad (13)$$

This equation can be used to obtain the coefficients for Q.

- Example we want to compute

$$\sum_{i=0}^{n} i^3 + 2i + 1 \quad (14)$$

then $Q_4(n) = a_4 n^4 + a_3 n^3 + a_2 n^2 + a_1$;

a_0=0 due to $a_0 = Q_{k+1}(0) = \sum_{i=0}^{-1} P_k(i) = 0$.

Replace $Q_4(n+1)$ and $Q_4(n)$ to (13) with $P_k(n)=n^3+2n+1$, and balance two sides it gives:

$$a_4 = 1/4, a_3 = -1/2, a_2 = 5/4, a_1 = 0$$

and then replace these coefficients to $Q_4(n+1)$ and simplify we obtain

$$\sum_{i=0}^{n} i^3 + 2i + 1 = \frac{(n+1)^2}{4}(n^2 + 4)$$

In order to use ILIRE to solve (10), denote f(n) as this summation then

$$f(n)-f(n-1)=P_k(n) \quad (15)$$

here f(n), f(n-1) play roles as $Q_{k+1}(n+1)$ and $Q_{k+1}(n)$

The CE for (15) is

$$(x-1)(x-1)^{k+1}=0 \quad (16)$$

Therefore the GS for f(n) is

$$f(n)=\sum_{i=0}^{k+1} c_i n^i \quad (17)$$

Theorem 1 : Summation of integral polynomial of degree k will be a polynomial of degree k+1

- For the summation in (12) we have:

$$f(n)-f(n-1)=n^3+2n+1 \quad (18)$$

Then the CE of (18) is

$$(x-1)(x-1)^4=0 \quad (19)$$

Then the GS as $f(n)=c_1+c_2n+c_3n^2+c_4n^3+c_5n^4$ and these coefficients can be obtained by the same methods given in sect. 3.1

3.3 Compute $\sum_{i=0}^{n}(i^2+1)3^i$

Let denote the above sum as f_n then the ILERE is obtained as

$$f_n - f_{n-1} = (n^2+1)3^n \quad (20)$$

and then the CE is $(x-1)(x-3)^3=0$ and the GS is $f_n = c_1 + c_2 3^n + c_3 n 3^n + c_4 n^2 3^n$ and we can use the same approach as in sect. 3.1 to find the result.

4. DC AND ILIRE

To find the order of the complexity in (2), we assume that $g(n)=O(n^k)$. More simple we assume that $g(n)=n^k$. Let $n=b^p$ and $s(p)=A(b^p)$ then Eq. (2) can be rewritten as

$$s(p)-ls(p-1)=b^{pk} \quad (21)$$

Then CE for s(p) will be

$$(x-l)(x-b^k)=0 \quad (22)$$

Then the GS solution for s(p) will be

$$S(p)=\begin{cases} c_1 l^p + c_2 (b^k)^p & if \quad l \neq b^k \\ c_1 l^p + c_2 p l^p & if \quad l = b^k \end{cases} \quad (23)$$

And $A(n)=s(\log_b(n))$ is given as

$$A(n)=\begin{cases} c_1 n^{\log_b(l)} + c_2 n^k & if \quad l \neq b^k \\ c_1 n^k + c_2 n^k \log_b n \end{cases} \quad (24)$$

If we just consider only the order of A(n), then

$$A(n)\begin{cases} \theta(n^k) & if \quad l < b^k \\ \theta(n^{\log_b(l)}) & if \quad l > b^k \\ \theta(n^k \log n) & if \quad l = b^k \end{cases} \quad (25)$$

5. SOME EXAMPLES

5.1 Find Median

In Fig. 1, assume that the array A has been decompose into several groups of p=5 elements, each group represents by one column, and the median of each group in the middle and assume that for each group the element above median are larger than median and the columns on the left of M* have median less than M*. Then the shading part on the top□right larger than M* and the shading part on the bottom□left less than M* will be removed and the median of the remaining part is the median of A.

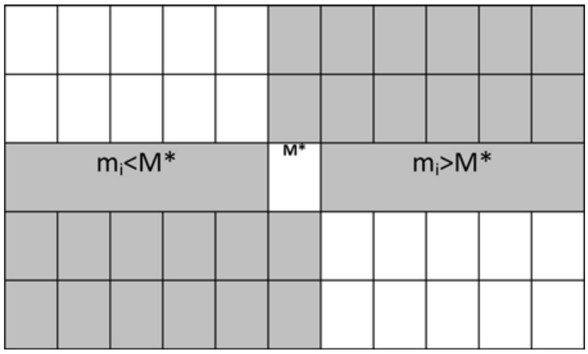

Fig. 1: Find Median.

The general algorithm is given as:

The general algorithm is given as:

```
Median(A[1..n])
{if n small return SimpleMed(A)
 Else
   { Decompose A into p=n/b groups of b
       elements, let denote as B1..Bp
     For i=1 to p
        M[i]=SimpleMed(Bi )
     M*=Median(M); Put M* to C
     For each group Bi that M[i]<M* pick up
       elements larger than M[i] and put them to C
     For each group Bi that M[i]>M* pick up elements
       smaller than M[i] and put them to C
Return Median(C)
   }
}
```

Then the number of elements in C is $\frac{n-b+3}{2}-\frac{n}{2b}$. Then the complexity for the median is

$$A(n)=px\ SimpleMed(b)+A(n/b)+A(\|C\|) \quad (26)$$

Here the first term obtained from finding medians for p groups, the second term is the median of medians, normally finding the median also arrange which elements are smaller median and which one are larger than median, the last term is the comparison need for finding median(C). The values of simpleMed(3)=3 and SimpleMed(5)=6 is obtained from [3]. Some typical values of b as

- b=3, p=n/3, SimpleMed(3)=3 , $\|C\| = \frac{n}{3}$ and

$$A3(n)=2A3(n/3)+n \qquad (27)$$

Apply Eq(19) to (21) we get A3(n)=$\theta(n)$

- b=5, p=n/5, SimpleMed(5)=6, $\|C\| = \frac{2n}{5} - 1 \approx 0.4n$ and

$$A5(n)=A5(n/5)+A5(0.4n)+6n/5 \qquad (28)$$

Use constructive proof we get A5(n)=3n.

Theorem 1: Median of an array can be done in linear time.

Really, median of an array can be found by with the complexity

$$A(n)=\frac{3}{2}(n-1) \text{ comparisons [3]}$$

5.2 Worst case Analysis of the Modified Quicksort

Quick sort get the worst case when the pivot element is too small or too large. In order to avoid the worst case the modified Quicksort algorithm use the median as pivot element.

```
Modified_Quicksort(A[1..n])

{ If n small  return Simplesort(A)

 Else {    Med=(A);

           Pivot(A,Med,P_loc)

           Modified_Quicksort(A[1..P_loc-1)

           Modified_Quicksort(A[P_loc+1..n)

             }

}
```

The median can be done in linear time, therefore the complexity of Modified_Quicksort can be written as

$$A(n)=2A(n/2)+g(n) \qquad (29)$$

Where g(n)=O(n), therefore A(n)=$\theta(n\lg n)$.

Theorem 2: The worst case of Modified Quicksort is $\theta(n\lg n)$

6. CONCLUSION

In design algorithm, DC plays an important role. Furthermore, DC methods lead to recursive algorithms, and then their complexity will be LIRE. Solving LIRE can be done by using generating function or characteristic equation. Solving the LIRE with non-constant coefficient will be considered in future.

REFERENCES

[1] G. Brassard & P. Bratley, 1996, Fundamentals of Algorithms, Prentice Hall

[2] U. Manber,1989, Introduction to Algorithms: A creative Approach, Addison Wesley

[3] S. Baase, 1988, Computer Algorithms: Introduction to Design and Analysis, 2nd Ed., Addison Wesley

[4] T.Minh and TV. To, 2006, Rational Generating Functions and the Solution of Linear Inhomogeneous Recurrence Equation, WSEAS Transactions on Mathematics, Issue 11, Vol. 5, Nov 2006, ISSN 1109-2769, pp 1191-1196

[5] T.Minh and TV.To,2006, Using Generating Functions to Solve Linear Inhomogeneous Recurrence Equations, Proceeding 6th WSEAS Int. Conf. On Simulation, Modelling and Optimization (SMO 06), Lisbon Portugal, Sept 22-24,2006

[6] H.S. Wilf, 1994, Generatingfunctionology, 2nd. Ed., Academic Press

[7] TV.To, 1992, Polar form approach to Geometric Modelling, AIT Disertation.