روش‌های حل معادلات بازگشتی

برای حل معادلات بازگشتی روش‌های زیر پیشنهادشده است:

- روش استقراء (Induction Method)
- روش تکرار (Iteration Method)
- روش معادله مشخصه (Characteristic Equation)
- روش قضیه اصلی (The Master Theorem Method)
- درخت بازگشتی (The Recursion Tree)

نکته: روش درخت بازگشتی برای آنالیز و حل معادلات بازگشتی، معمولا در درس ساختمان‌داده‌ها، به طور مفصل توضیح‌داده‌می‌شود.

۱- روش استقراء

1. Estimate the form of the solution
2. Prove by mathematical induction

مثال: محاسبه فاکتوریل به روش بازگشتی

Fact(n) = Fact(n-1)*n;

Number of *: T(n) = T(n-1) + 1 (Recurrence equation)
Proof:

Induction base: for n=0 → T(0) = 0;
Induction hypothesis: for arbitrary positive integer n → T(n) = n (Guess)
Induction step: to show → T(n+1) = n+1

Replace n by n+1 in the recurrence equation, we get: T(n+1) = T(n) +1 = n+1 (It's correct!)
So this completes the proof that T(n) is correct → T(n) = Θ(n)

۲- روش تکرار

مثال: محاسبه تابع پیچیدگی جستجوی دودویی در بدترین حالت

$$\begin{aligned} W(n) &= 1 + W(\lfloor n/2 \rfloor) \\ &= 1 + (1 + W(\lfloor \lfloor n/2 \rfloor / 2 \rfloor)) = 2 + W(\lfloor n/4 \rfloor) \\ &= 2 + (1 + W(\lfloor n/8 \rfloor)) = 3 + W(\lfloor n/8 \rfloor) \\ &\ldots \\ &= k + W(\lfloor n/2^k \rfloor) = k + W(1) = k + 1 \\ &= \lfloor \lg n \rfloor + 1 \in \Theta(\lg n) \end{aligned}$$

البته قبلاً اشاره شده بود که $W(n) = \lfloor \lg n \rfloor + 1$

۳- روش معادله مشخصه

الف- معادله بازگشتی خطی همگون(Homogeneous Linear Recurrence)

مثال: $t_n - 4t_{n-1} + 3t_{n-2} = 0 \quad n > 1$ $t_0 = 0$ $t_1 = 1$ $Characteristic: \quad r^2 - 4r + 3 = 0$ $\Rightarrow r_1 = 3, r_2 = 1 \Rightarrow t_n = c_1 3^n + c_2 1^n \Rightarrow$ $t_0 = 0 = c_1 + c_2$ $t_1 = 1 = c_1 3 + c_2$ $\rightarrow c_1 = 0.5, c_2 = -0.5$ $\rightarrow t_n = \frac{1}{2}3^n - \frac{1}{2}$	The **homogeneous linear** recurrence equation with constant coefficients: $a_0 t_n + a_1 t_{n-1} + ... + a_k t_{n-k} = 0$ Its characteristic equation is: $a_0 r^k + a_1 r^{k-1} + ... + a_k r^0 = 0$ If there are k distinct solutions r1, r2, …, rk, the only solution to the recurrence is: $t_n = c_1 r_1^n + c_2 r_2^n + ... + c_k r_k^n$

ب- معادله بازگشتی خطی ناهمگون(Non-Homogeneous Linear Recurrence)

مثال(معادله بازگشتی مرتب‌سازی سریع در بدترین حالت)**:** $t_n = t_{n-1} + n - 1 \quad n > 1$ $t_0 = 0$ $t_n - t_{n-1} = n - 1$ $Characteristic:$ $(r-1)(r-1)^2 = 0$ $(r-1)^3 = 0$ $\Rightarrow r = 1 \ (multiplicity\ 3) \Rightarrow t_n = c_1 1^n + c_2 n 1^n + c_3 n^2 1^n$ $\rightarrow t_n = \frac{n(n-1)}{2}$	The **Non- homogeneous linear** recurrence equation: $a_0 t_n + a_1 t_{n-1} + ... + a_k t_{n-k} = b^n P(n)$ Its characteristic equation is: $(a_0 r^k + a_1 r^{k-1} + ... + a_k)(r-b)^{d+1} = 0$ solution to the recurrence is: $t_n = ---$

۴- روش قضیه اصلی

Let f be an increasing function that satisfies the recurrence relation

$$f(n) = a * f(\frac{n}{b}) + c * n^d$$

whenever $n = b^k$, where k is a positive integer, $a > 1$, b is an integer greater than 1, and c and d are numbers with c positive and d nonnegative.

$$f(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log_b n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

الگوریتم ضرب ماتریس استراسن (Strassen's Matrix Multiplication Algorithm)

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$m_1 = (a_{11} + a_{22})(b_{11} + b_{22})$$
$$m_2 = (a_{21} + a_{22})b_{11}$$
$$m_3 = a_{11}(b_{12} - b_{22})$$
$$m_4 = a_{22}(b_{21} - b_{11})$$
$$m_5 = (a_{11} + a_{12})b_{22}$$
$$m_6 = (a_{21} - a_{11})(b_{11} + b_{12})$$
$$m_7 = (a_{12} - a_{22})(b_{21} + b_{22})$$

$$C = \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

تعداد ضرب‌ها: ۷

تعداد جمع و تفریق‌ها: ۱۸

Problem: Find the product of two n×n matrices where n is a power of 2.
Inputs: an integer n that is power of 2, and two matrices A and B.
Outputs: the product C of A and B.

```
void strassen (int n, n×n_matrix A, B,
n×n_matrix& C)
{
  if (n ≤ threshold)
    compute C = A×B using the standard algorithm;
  else
  {
    partition A
          into four submatrices A11, A12, A21, A22 ;
    partition B
          into four submatrices B11, B12, B21, B22 ;
    compute C = A×B using the Strassen's method;
    }
}
```

Alg. 4-1

Every-Case Time Complexity (no. of multiplications)

$$T(n) = 7T(\tfrac{n}{2}) \quad \text{for } n > 1, \; n \; \text{ a power of } 2$$
$$T(1) = 1.$$

By Master Theorem :

$$\because a = 7, \; b = 2, \; d = 0 \Rightarrow a > b^d$$
$$\therefore T(n) = n^{\log_2 7} \in \Theta(n^{2.81})$$

Every-Case Time Complexity (no. of +/−)

$$T(n) = 7T(\tfrac{n}{2}) + 18(\tfrac{n}{2})^2 \quad \text{for } n > 1,$$
$$n \; \text{ a power of } 2$$
$$T(1) = 1.$$

By Master Theorem :

$$\because a = 7, \; b = 2, \; d = 2 \Rightarrow a > b^d$$
$$\therefore T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{2.81})$$

ضرب اعداد صحیح بزرگ (Larger Integer Multiplication)

When do we need to multiply two very large numbers? In Cryptography and Network Security,

```
           X        ************
                    ************
--------------------------------
                    ************
                   ************
                  ************
                 ************
                ************
               ************
              ************
             ************
            ************
           ************
          ************
         ************
--------------------------------
     ***********************
```

$\Theta(n^2)$ bit operations

Divide And Conquer

X =	a	b
Y =	c	d

$$X = a2^{n/2} + b,\ Y = c2^{n/2} + d$$

$$XY = ac2^{n} + (ad + bc)2^{n/2} + bd$$

الگوریتم اول(با استفاده از روش تقسیم وحل)

MULT(X,Y)
if |X| = |Y| = 1 **then do return** XY
else
return $\mathrm{MULT}(a,c)2^{n} + \left(\mathrm{MULT}(a,d) + \mathrm{MULT}(b,c)\right)2^{n/2} + \mathrm{MULT}(b,d)$

Alg. 4-2

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 4T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

By the Master Theorem:

$$T(n) = \Theta\left(n^2\right)$$

پرسش: آیا می‌توان مرتبه الگوریتم ۴-۲ را بهبود بخشید؟ چگونه؟ Karatsuba 1962

معادله گاوس:

$$ad + bc = (a + b)(c + d) - ac - bd$$

الگوریتم دوم(بهبود یافته الگوریتم اول)

MULT(X,Y)
if |X| = |Y| = 1 **then do return** XY
else
A1 = MULT(*a*,*c*);
A2 = MULT(*b*,*d*);
A3 = MULT((*a*+*b*)(*c*+*d*));

$$A_1 2^{n} + \left(A_3 - A_1 - A_2\right)2^{n/2} + A_2$$

Alg. 4-3

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

By the Master Theorem:

$$T(n) = \Theta\left(n^{\log_2 3}\right) = \Theta\left(n^{1.58}\right)$$

تمرین‌ها

تمرین۴-۱: برای محاسبه توان، دو الگوریتم یکی با روش بازگشتی و دیگری با روش تکراری، ارائه‌کنید و سپس مرتبه هر الگوریتم را محاسبه نماید.

تمرین۴-۲: برای مساله برج هانوی، یک الگوریتم بازگشتی ارائه و مرتبه آن را محاسبه کنید.