CSC10004: Data Structure and Algorithms

Lecture 3: Recurrences

Lecturer: Bùi Văn Thạch
TA: Ngô Đình Hy/Lê Thị Thu Hiền
{bvthach,ndhy}@fit.hcmus.edu.vn, lthien@hcmus.edu.vn

Goals

1. To help students able to analyze recursive algorithms by using recurrence techniques.

Outline

1. Recursion trees

2. Characteristic equation

Outline

1. Recursion trees

2. Characteristic equation

Example 1: Merge Sort: preliminary analyses

Analyze the complexity of the merge sort algorithm

```
struct Node {
    int key;
    Node* pNext;
};

struct ListInt{
    Node first;
    Node last;
    int size;
} LIST;
```

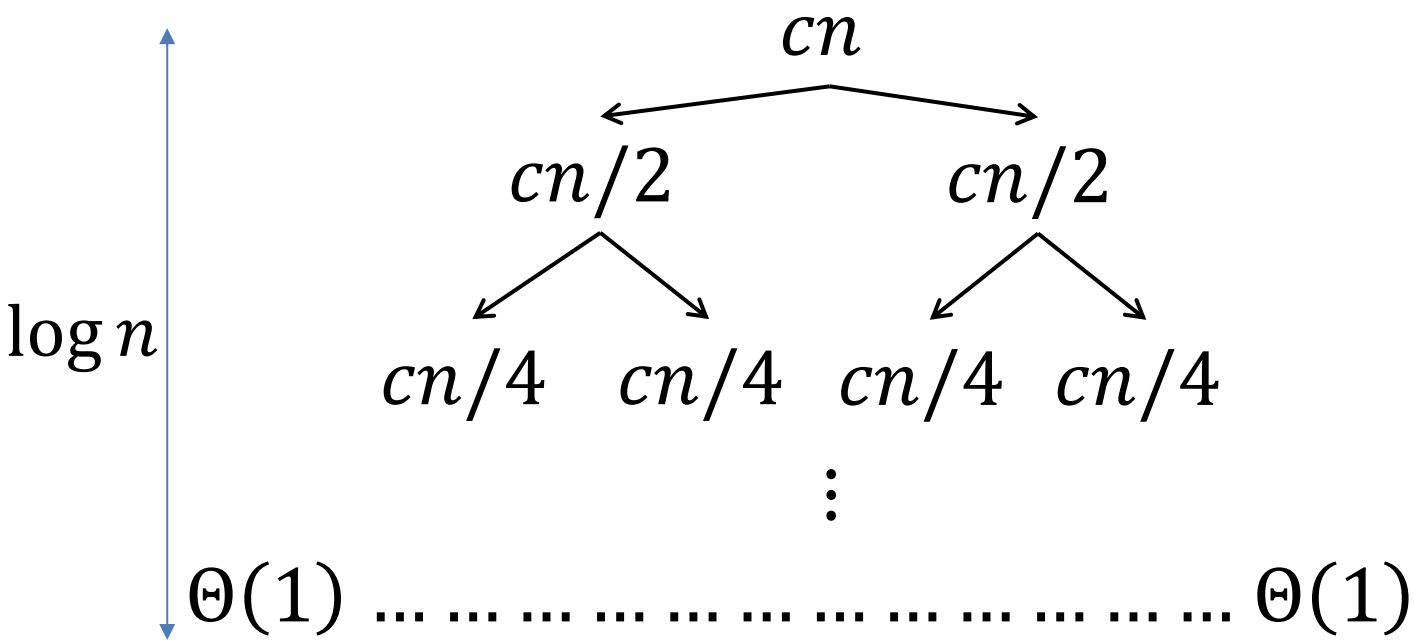
```
void mergeSort(LIST list)
    if (list->size == 1)
        return;                                          1

    first_size = floor(list->size/2);                    2

    first_list = LIST(list->first,
get_node_at_position(list, first_size),                  O(1)
first_size);
    second_list =
LIST(get_node_at_position(list, first_size+1),           O(n)
list->last, list->size-first_size);

    mergeSort(first_list);                               T(n_1)
    mergeSort(second_list);                              T(n − n_1)
    LIST = mergeTwoSortedLists(first_list,               Θ(n)
second_list);
}
```

$T(n)$, where $n = \text{list} \rightarrow \text{size}$

Example 1: Merge Sort

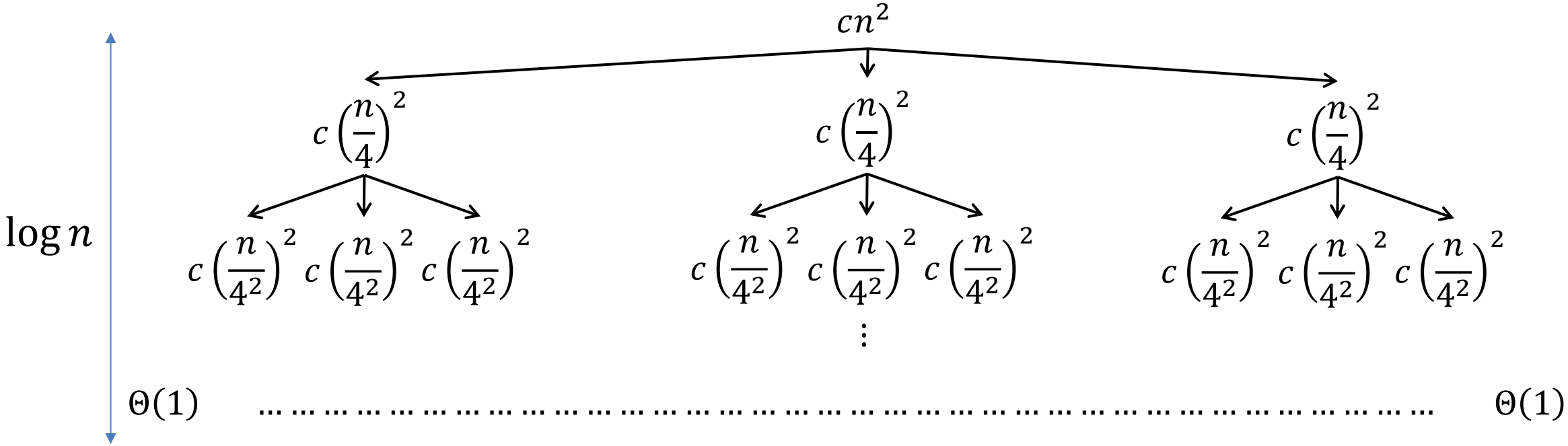
$$T(n) = \begin{cases} \Theta(1), \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n), \text{otherwise} \end{cases}$$

1. Depth of the leaves (the tree height): $\log n$.
2. Sum for each level: cn.
3. Sum across all levels: $cn \log n = O(n \log n)$.



Example 2: Advanced Merge Sort (1/2)

$$T(n) = \begin{cases} \Theta(1), \text{if } n = 1 \\ 3T\left(\frac{n}{4}\right) + \Theta(n^2), \text{otherwise} \end{cases}$$



Example 2: Advanced Merge Sort (2/2)

1. Depth of the leaves (the tree height): $\log n$.
2. Sum for each level: $3^i c\left(n/4^i\right)^2 = (3/16)^i cn^2$.
3. Sum across all levels:

$$T(n) = \sum_{i=0}^{\log_4 n} \left(\frac{3}{16}\right)^i cn^2 \leq cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i = \frac{1}{1-3/16} cn^2 = O(n^2)$$

Moreover, $T(n) = 3T\left(\frac{n}{4}\right) + \Theta(n^2) = \Omega(n^2)$ then $T(n) = \Theta(n^2)$.

Recurrent relation: definition

1. A first- or second-order linear homogeneous recurrence relations

$$T(n) = \begin{cases} a \cdot T(n-1) + f(n) \\ a \cdot T(n-1) + b \cdot T(n-2) + f(n) \end{cases}$$

2. Their corresponding characteristic functions.

$$\begin{cases} x = a \\ x^2 = ax + b \end{cases}$$

Recurrent relation: definition

1. A first- or second-order linear homogeneous recurrence relations

$$T(n) = \begin{cases} a \cdot T(n-1) + f(n) \\ a \cdot T(n-1) + b \cdot T(n-2) + f(n) \end{cases}$$

2. Their corresponding characteristic functions.

$$\begin{cases} x = a \\ x^2 = ax + b \end{cases}$$

Solving the first-order characteristic equation

$$T(n) = a \cdot T(n-1) + f(n)$$
$$= a\big(a \cdot T(n-2) + f(n-1)\big) + f(n)$$
$$= a^2 T(n-2) + af(n-1) + f(n)$$
$$= a^{n-1}T(1) + \sum_{i=0}^{n-2} a^i f(n-i)$$

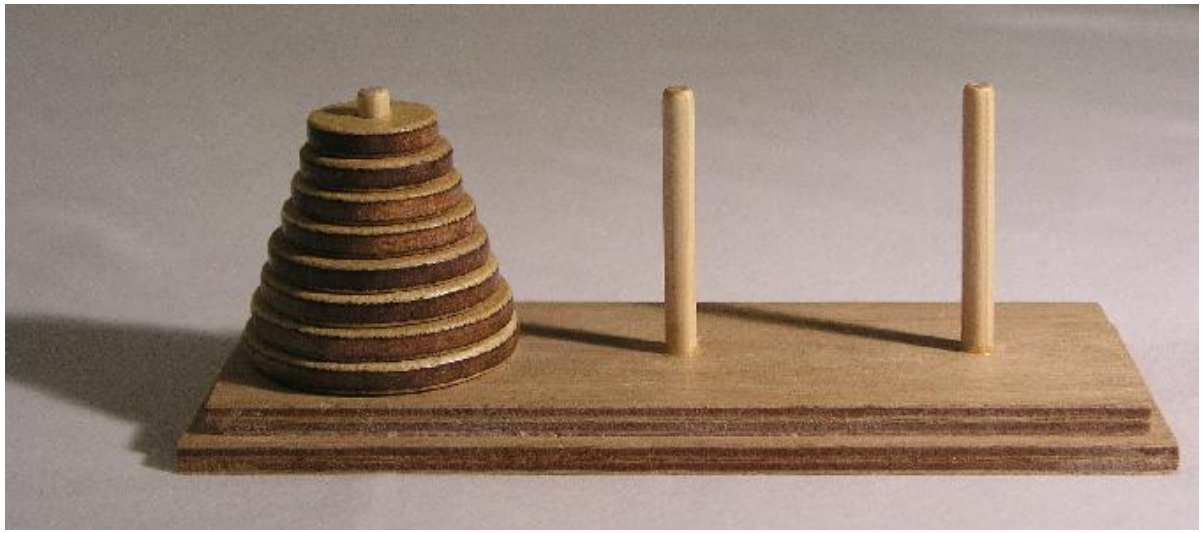
Example: Hanoi's Tower (1/6)

A mathematical puzzle consisting of three rods and a number of disks of various diameters, which can slide onto any rod. The puzzle begins with the disks stacked on one rod in order of decreasing size, the smallest at the top, thus approximating a conical shape. The objective of the puzzle is to move the entire stack to one of the other rods, obeying the following rules:

1. Only one disk may be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or an empty rod.
3. No disk may be placed on top of a disk that is smaller than it.

Allardice, R. E., and A. Y. Fraser. "La tour d'Hanoï." *Proceedings of the Edinburgh Mathematical Society* 2 (1883): 50-53.

Example: Hanoi's Tower (2/6)



A model set of the Tower of Hanoi with 8 disks

Example: Hanoi's Tower (3/6)

According to an old Indian legend, the Brahmins have been following each other for a very long time on the steps of the alter in the Temple of Bernares, carrying out the moving of the Sacred Tower of Brahma with sixty-four levels in fine gold, trimmed with diamonds from Golconda. When all is finished, the Tower and the Brahmins will fall, and that will be the end of the world!

Allardice, R. E., and A. Y. Fraser. "La tour d'Hanoï." *Proceedings of the Edinburgh Mathematical Society* 2 (1883): 50-53.

Example: Hanoi's Tower (4/6)

1. The minimal number of moves required to solve a Tower of Hanoi puzzle with n disks is $2^n - 1$. [1]
2. If the old Indian legend were true, and if the priests were able to move disks at a rate of one per second, using the smallest number of moves, it would take them $2^{64} - 1$ seconds or roughly 585 billion years to finish, which is about 42 times the estimated current age of the universe (14 billions years old).

[1] Petkovi_, Miodrag. *Famous puzzles of great mathematicians*. American Mathematical Soc., 2009.

Example: Hanoi's Tower (5/6): source code

Analyze the number of printings

```
void TowerOfHanoi(int n, char source, char target, char auxiliary) {
    if (n > 0) {
        // Move n-1 disks from source to auxiliary using target as the temporary peg
        TowerOfHanoi(n - 1, source, auxiliary, target);

        // Move the nth disk from source to target
        cout << "Move disk " << n << " from " << source << " to " << target << endl;

        // Move n-1 disks from auxiliary to target using source as the temporary peg
        TowerOfHanoi(n - 1, auxiliary, target, source);
    }
}
```

$$T(n) = T(n-1) + 1 + T(n-1)$$

Example: Hanoi's Tower (6/6)

1. We have

$$T(n) = 2T(n-1) + 1$$

$\Rightarrow T(1) = 1, a = 2, \text{and } f(n) = 1.$

$\Rightarrow T(n)$

$$= a^{n-1}T(1) + \sum_{i=0}^{n-2} a^i f(n-i) = 2^{n-1} \cdot 1 + \sum_{i=0}^{n-2} 2^i = 2^{n-1} + \frac{(2^{n-1}-1)}{2-1} = 2^n - 1$$

Recurrent relation: definition

1. A first- or second-order linear homogeneous recurrence relations

$$T(n) = \begin{cases} a \cdot T(n-1) + f(n) \\ a \cdot T(n-1) + b \cdot T(n-2) + f(n) \end{cases}$$

2. Their corresponding characteristic functions.

$$\begin{cases} x = a \\ x^2 = ax + b \end{cases}$$

Recurrent relation: solving the second-order characteristic equation

Let x_1 and x_2 be the roots of $x^2 = ax + b$. Then $\alpha + \beta = a$ and $\alpha\beta = b$. Then we have:

$$T(n) = a \cdot T(n-1) + b \cdot T(n-2) + f(n) = (\alpha+\beta)T(n-1) + \alpha\beta T(n-2) + f(n)$$

$$\Leftrightarrow T(n) - \alpha T(n-1) = \beta\big(T(n-1) - \alpha T(n-2)\big) + f(n)$$

$$= \beta\Big(\beta\big(T(n-2) - \alpha T(n-3)\big) + f(n-1)\Big) + f(n)$$

$$= \beta^2\big(T(n-2) - \alpha T(n-3)\big) + \sum_{i=0}^{1} \beta^i f(n-i)$$

$$= \beta^{n-1}\big(T(1) - \alpha T(0)\big) + \sum_{i=0}^{n-2} \beta^i f(n-i) = \beta^{n-1}C + \sum_{i=0}^{n-2} \beta^i f(n-i)$$

Recurrent relation: solving the second-order characteristic equation

$$T(n) - \alpha T(n-1) = \beta^{n-1} C + \sum_{i=0}^{n-2} \beta^i f(n-i)$$

$$\Rightarrow T(n) = \alpha^{n-1} T(1) + C \sum_{j=0}^{n-2} \alpha^j \beta^{n-1-j} + \sum_{j=0}^{n-2} \left(\alpha^j \sum_{i=0}^{n-2-j} \beta^i f(n-j-i) \right)$$

Example: Fibonacci

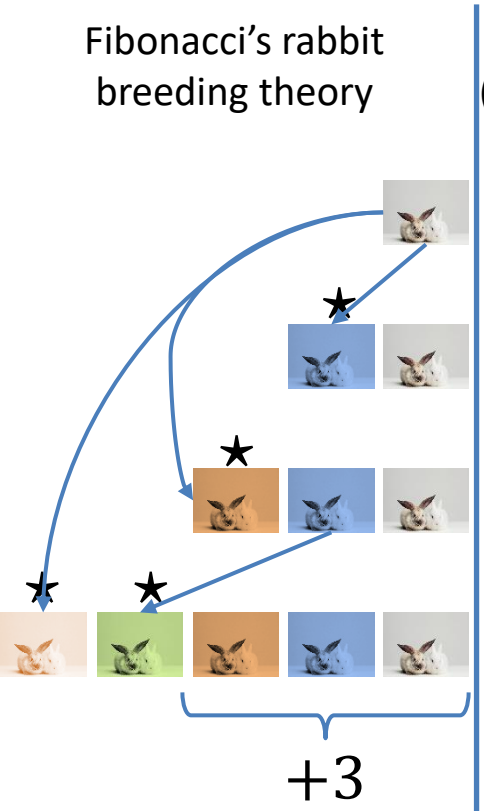
1. The Fibonacci sequence is a sequence in which each number is the sum of the two preceding ones.

$$F(n) = F(n-1) + F(n-2) \text{ and } F(0) = 0 \text{ and } F(1) = 1$$

2. The Fibonacci numbers were first described in Indian mathematics as early as **200 BC** in a work by Pingala on enumerating possible patterns of Sanskrit poetry formed from syllables of two lengths.
3. Introduced to Western European mathematics in Leonardo of Pisa's book titled Liber Abaci in 1202 [2].

[1] Singh, Parmanand. "The so-called Fibonacci numbers in ancient and medieval India." *Historia Mathematica* 12.3 (1985): 229-244.
[2] Sigler, Laurence. *Fibonacci's Liber Abaci: a translation into modern English of Leonardo Pisano's book of calculation*. Springer Science & Business Media, 2003.

Example: Fibonacci in Animal Breeding Patterns



1. Rabbits follow the Fibonacci sequence in their reproductive pattern.
2. A single pair of rabbits (one male and one female) produces another pair in the second month.
3. From the third month onwards, each pair produces a new pair every month.
4. The sequence of pairs of rabbits is 1, 1, 2, 3, 5, 8, 13, and so on.
5. The breeding pattern reflects the natural balance of reproduction and population growth.
6. The population grows slowly initially as the breeding pairs are young and need time to mature.
7. As the population increases, more pairs are available to reproduce, leading to an exponential growth rate that follows the Fibonacci sequence.

Example: Fibonacci numbers: source code

Analyze the number of returns

```
int Fibonacci(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;

    return Fibonacci(n-1) + Fibonacci(n-2);
}
```

$T(n)$
=

$T(n-1) + T(n-2)$

Example: Fibonacci numbers (1/2)

1. We have

$$T(n) = T(n-1) + T(n-2)$$

$\Rightarrow \alpha = \frac{1+\sqrt{5}}{2}, \beta = \frac{1-\sqrt{5}}{2}$ which are roots of $x^2 = x + 1$ and

$f(n) = 0$

Example: Fibonacci numbers (2/2)

$$T(n) = \alpha^{n-1}T(1) + \big(T(1) - \alpha T(0)\big)\sum_{j=0}^{n-2}\alpha^j\beta^{n-1-j} = \alpha^{n-1} + \beta^{n-1}\sum_{j=0}^{n-2}\left(\frac{\alpha}{\beta}\right)^j$$

$$= \alpha^{n-1} + \beta^{n-1}\cdot\frac{1-\left(\frac{\alpha}{\beta}\right)^{n-1}}{1-\frac{\alpha}{\beta}} = \alpha^{n-1} + \frac{\beta^n}{\beta-\alpha} - \frac{\beta}{\beta-\alpha}\alpha^{n-1}$$

$$= \frac{\alpha^n - \beta^n}{\alpha - \beta}$$

$$= \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Exercise 1: Karatsuba's algorithm

1. Input: Two positive numbers a, b.
2. Output: The product $f(a, b) = a \times b$

Compute the complexity of the following pseudocode:

- Step 1. If both a and b have only one binary digit, then return $a \times b$.
- Step 2. $n \leftarrow \max\{\lceil \log_2 a \rceil, \lceil \log_2 b \rceil\}$.
- Step 3. Split a into 2 parts (a_l, a_h) where a_l contains $\left\lfloor \frac{n}{2} \right\rfloor$ lower binary digits, and a_h contains $\lceil \log_2 a \rceil - \left\lfloor \frac{n}{2} \right\rfloor$ higher binary digits. Split b into 2 parts (b_l, b_h) where b_l contains $\left\lfloor \frac{n}{2} \right\rfloor$ lower binary digits, and b_h contains $\lceil \log_2 b \rceil - \left\lfloor \frac{n}{2} \right\rfloor$ higher binary digits.
- Step 4. $z_0 \leftarrow f(a_l, b_l)$; $z_1 \leftarrow f(a_h + a_l, b_h + b_l)$; $z_2 \leftarrow f(a_h, b_h)$.
- Step 5. Return $z_2 \times 2^{2\left\lfloor \frac{n}{2} \right\rfloor} + (z_1 - z_2 - z_0) \times 2^{\left\lfloor \frac{n}{2} \right\rfloor} + z_0$

Solution

- Let $n = \max\{\lceil \log_2 a \rceil, \lceil \log_2 b \rceil\}$.
- Let $T_1(n)$ be the number of multiplications in Step 1 of Karatsuba's algorithm. Then, $T_1(1) = 1$ and $T_1(n) = 3T_1\left(\frac{n}{2}\right)$.
- Let $T_2(n)$ be the number of binary digits needing splitting in Step 3 of Karatsuba's algorithm. Then, $T_2(1) = 0$ and $T_2(n) = 3T_1\left(\frac{n}{2}\right) + n + n$.
- Let $T_3(n)$ be the number of binary digits needing adding in Step 4 of Karatsuba's algorithm. Then, $T_3(1) = 0$ and $T_3(n) = 3T_3\left(\frac{n}{2}\right) + \frac{n}{2} + \frac{n}{2}$.
- Let $T_4(n)$ be the number of binary digits needing adding in Step 5 of Karatsuba's algorithm. Then, $T_4(1) = 0$ and $T_4(n) = 3T_4\left(\frac{n}{2}\right) + 2.n + 2.2n$.
- Therefore, if $T(n) = T_1(n) + T_2(n) + T_3(n) + T_4(n)$, then we have:

$$\begin{cases} T(1) = 1 \\ T(n) = 3T\left(\frac{n}{2}\right) + 9n \end{cases}$$

- Because $9n = \Theta(n) = \Theta\left(n^{\log_2 3 - \epsilon}\right)$ where $\epsilon = \log_2 3 - 1 > 0$, it implies that
- $T(n) = \Theta\left(n^{\log_2 3}\right)$

Exercise 2: Strassen's algorithm

Input: Two matrices $A, B \in \mathbb{R}^{n \times n}$

Output: The product $A \times B$.

The idea:

- $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = A \times B$

$$M_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22}) \times B_{11}$$
$$M_3 = A_{11} \times (B_{12} - B_{22})$$
$$M_4 = A_{22} \times (B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12}) \times B_{22}$$
$$M_6 = (A_{21} - A_{11}) \times (B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22}) \times (B_{21} + B_{22})$$

- Then, we have:

$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

- Let $T(n)$ be the number of multiplications and additions two elements of matrices. Estimate $T(n)$.

Solution

- $T(1) = 1$

$$T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2 = 7T\left(\frac{n}{2}\right) + \frac{9}{2}n^2$$

- Let $f(n) = \frac{9}{2}n^2 = \Theta(n^2) = \Theta\left(n^{\log_2 7 - \epsilon}\right)$ where $\epsilon = \log_2 7 - 2 > 0$.
- Therefore, $T(n) = \Theta\left(n^{\log_2 7}\right)$.

Q & A