CHƯƠNG 1

KỸ THUẬT
PHÂN TÍCH THUẬT TOÁN

Tuần 2: **Tính độ phức tạp của Chương trình đệ quy**

(Phương pháp truy hồi)

Võ Huỳnh Trâm

Phương pháp tính độ phức tạp

- Xét phương pháp tính độ phức tạp trong 3 trường hợp:

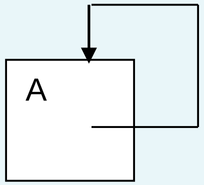
  (1) Chương trình **không gọi** chương trình con.

  (2) Chương trình **có gọi** chương trình con không đệ quy.

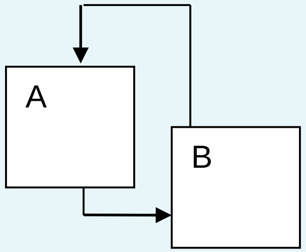
  (3) Chương trình **đệ quy.**

Phân tích các chương trình đệ qui

- 2 dạng chương trình đệ quy:



Đệ quy trực tiếp



Đệ quy gián tiếp

- Tính độ phức tạp chương trình đệ quy:

  (1) Thành lập phương trình đệ quy T(n)

  (2) Giải phương trình đệ quy tìm nghiệm

  → Suy ra tỷ suất tăng f(n) hay O(f(n)).

CANTHO UNIVERSITY

Chương trình đệ quy

- Chương trình đệ quy giải bài toán kích thước n, phải có ít nhất một *trường hợp dừng* ứng với một n cụ thể và *lời gọi đệ quy* giải bài toán kích thước k (k<n)
- **Ví dụ** : Chương trình đệ quy tính n!

```
int giai_thua(int n) {
    if (n == 0)
        return 1;
    else
        return n * giai_thua(n - 1);
}
```

- *Trường hợp dừng* n = 0 và *lời gọi đệ quy* k = n-1.

Thành lập phương trình đệ quy

- **Phương trình đệ quy** là phương trình biểu diễn mối liên hệ giữa T(n) và T(k), trong đó T(n) và T(k) là thời gian thực hiện chương trình có kích thước dữ liệu nhập tương ứng là n và k, với k < n.
- Để thành lập được phương trình đệ quy, phải căn cứ vào chương trình đệ quy.

  - ***Khi đệ quy dừng***: xem xét khi đó chương trình làm gì và tốn hết bao nhiêu thời gian (thông thường thời gian này là hằng số C(n))

  - ***Khi đệ quy chưa dừng***: xem xét có bao nhiêu lời gọi đệ quy với kích thước k thì sẽ có bấy nhiêu T(k).
- Ngoài ra, còn phải xem xét thời gian phân chia bài toán và tổng hợp các lời giải (chẳng hạn gọi thời gian này là d(n)).

Dạng phương trình đệ quy

- Dạng tổng quát của một phương trình đệ quy sẽ là:

$$T(n) = \begin{cases} C(n) \\ F(T(k)) + d(n) \end{cases}$$

- **C(n)**: thời gian thực hiện chương trình ứng với trường hợp đệ quy dừng.
- **F(T(k)):** hàm xác định thời gian theo T(k).
- **d(n)**: thời gian phân chia bài toán và tổng hợp các kết quả.

CANTHO UNIVERSITY

Ví dụ 1. Phương trình đệ quy của n!

n! = n*(n-1)*(n-2)*… *2*1

- Gọi T(n) là thời gian tính n!.
- Thì T(n-1) là thời gian tính (n-1)!.
- Trong trường hợp n = 0 thì chương trình chỉ thực hiện một lệnh *return 1*, nên tốn O(1), do đó ta có T(0) = C_1.
- Trong trường hợp n > 0 chương trình phải gọi đệ quy *giai_thua(n-1)*, việc gọi đệ quy này tốn T(n-1).
- Sau khi có kết quả của việc gọi đệ quy, chương trình phải nhân kết quả đó với n và trả về tích số. Thời gian để thực hiện phép nhân và trả kết quả về là một hằng C_2.
- Vậy ta có phương trình đệ quy như sau:

$$T(n) = \begin{cases} C_1 & \text{nếu n=0} \\ T(n-1) + C_2 & \text{nếu n>0} \end{cases}$$

Thuật toán MergeSort

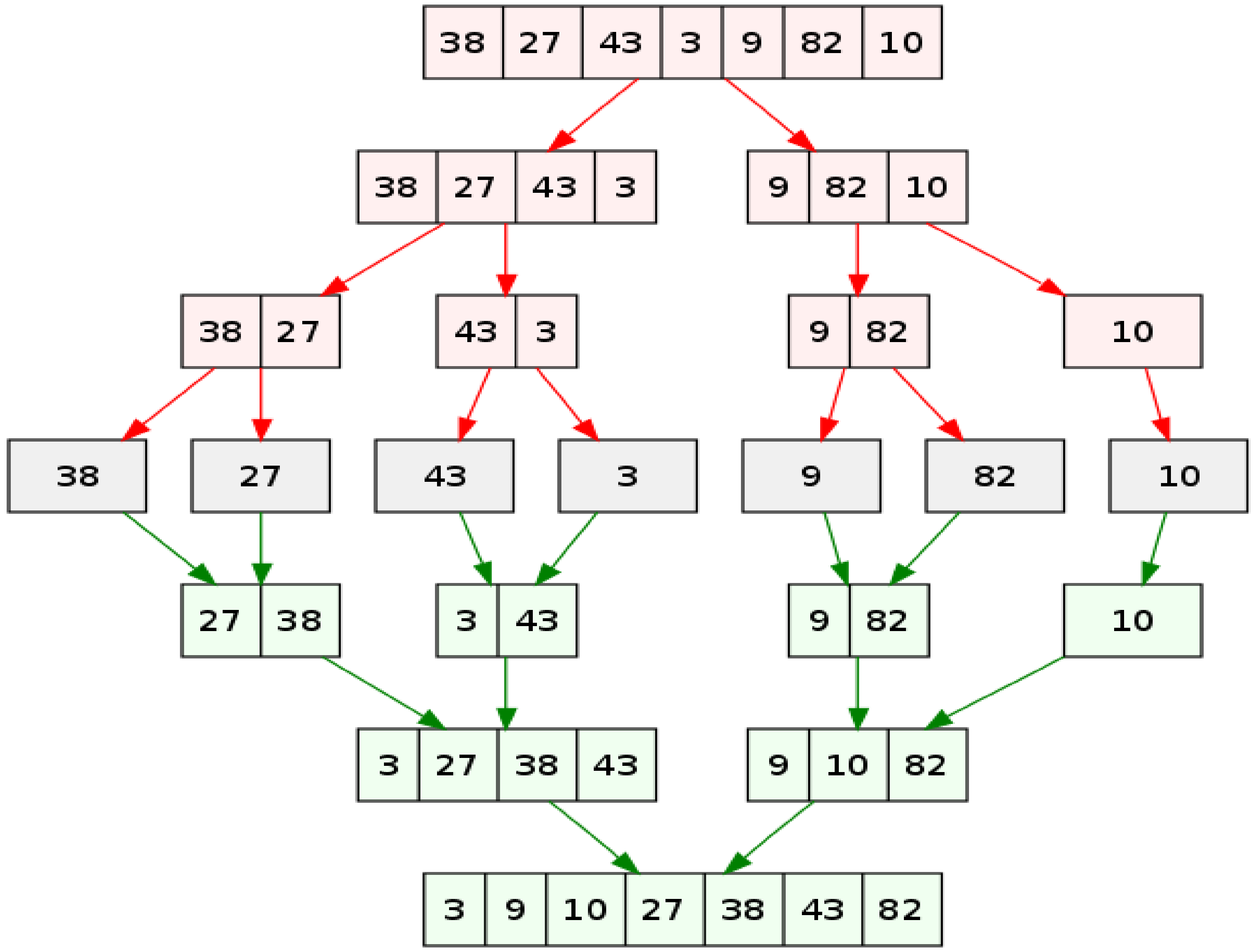
```
void mergeSort(int arr[], int l, int r) {
   if (l < r)    {
      int m = l+(r-l)/2;    // Tương tự (l+r)/2, nhưng cách này tránh tràn số khi l và r lớn
      // Gọi hàm đệ quy tiếp tục chia đôi từng nửa mảng
      mergeSort(arr, l, m);
      mergeSort(arr, m+1, r);
      merge(arr, l, m, r); }
   }
```

- ***Thủ tục merge*** nhận 2 danh sách đã có thứ tự, trộn chúng lại với nhau để được một danh sách có thứ tự.

Mô hình minh hoạ Mergesort

- Sắp xếp danh sách L gồm 7 phần tử :

38, 27, 43, 3, 9 82, 10



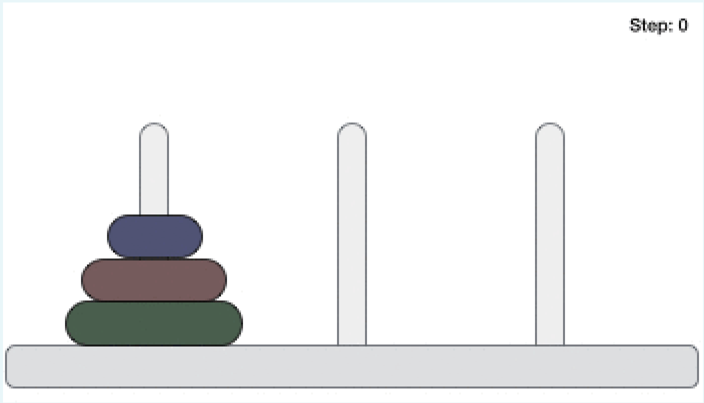
Ví dụ 2. Phương trình đệ quy của thuật toán MergeSort

- $T(n)$ là thời gian thực hiện **mergeSort** danh sách n, $T(n/2)$ là thời gian thực hiện **mergeSort** một danh sách n/2 phần tử.
- Khi L có độ dài 1 (n = 1): chương trình chỉ làm một việc duy nhất là *return(L)*, việc này tốn $O(1) = C_1$ thời gian.
- Khi L có độ dài n > 1: chương trình gọi đệ quy **mergeSort** 2 lần cho 2 danh sách con với độ dài n/2, do đó thời gian để gọi 2 lần đệ quy này là $2T(n/2)$.
- Ngoài ra, còn phải tốn thời gian chia danh sách thành 2 nửa bằng nhau và trộn hai danh sách kết quả (Merge): cần thời gian $O(n) = nC_2$.
- Vậy ta có phương trình đệ quy như sau:

$$T(n) = \begin{cases} C_1 & \text{nếu } n=1 \\ 2T\left(\frac{n}{2}\right) + nC_2 & \text{nếu } n>1 \end{cases}$$

Ví dụ 3. Phương trình đệ quy của Thủ tục tháp Hà nội số tầng n

```
void ThapHN(int n, char A, char B, char C)
{  if(n > 1)
   {
     ThapHN(n-1,A, C, B);
     printf("Chuyen tu %c sang %c\n", A, C);
     ThapHN(n-1,B, A, C);
   }
}
```



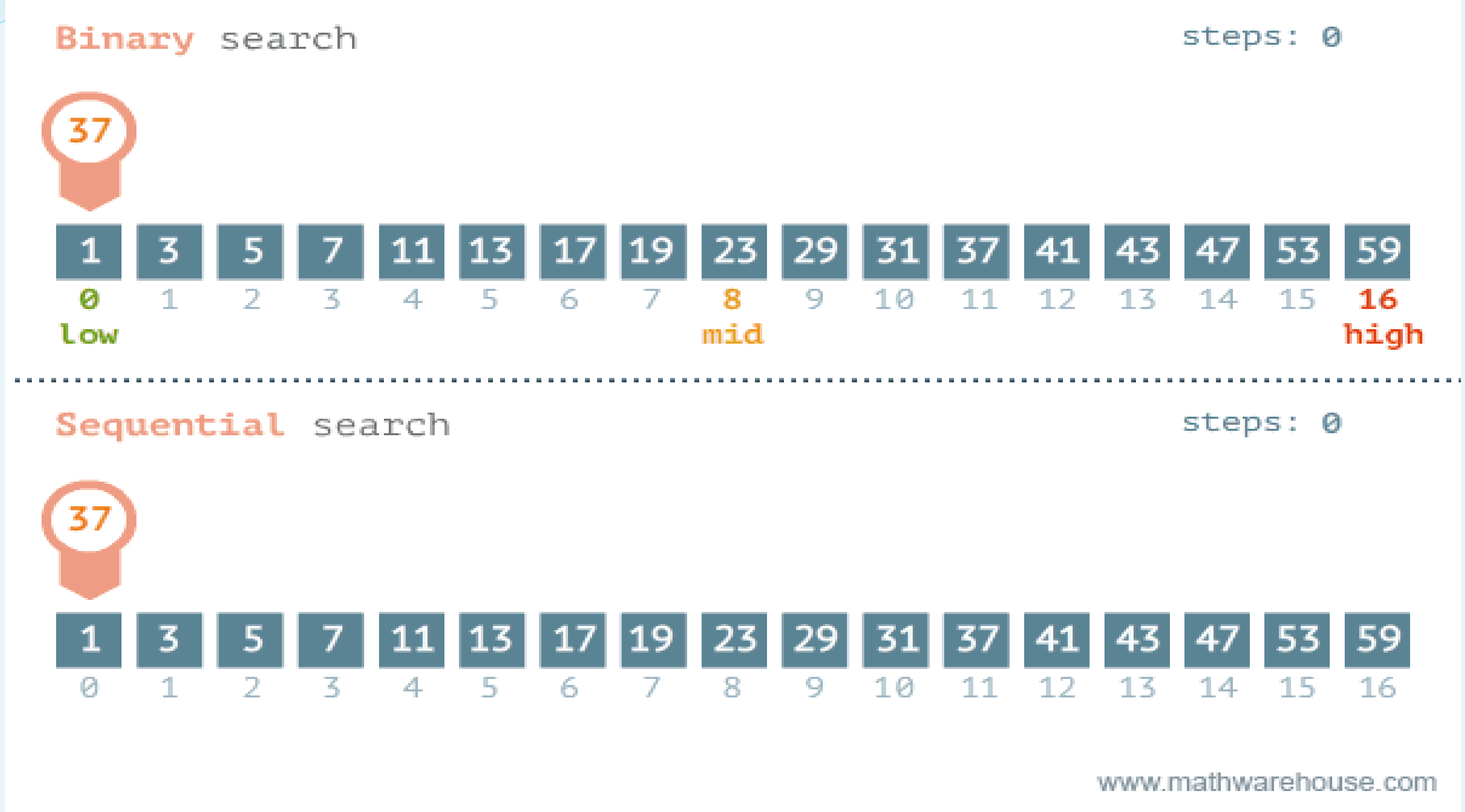
Phương trình đệ quy : $$T(n) = \begin{cases} C1 & n = 1 \\ 2T(n-1) + C2 & n > 1 \end{cases}$$

Ví dụ 4. Phương trình đệ quy của thuật toán tìm kiếm nhị phân dãy n thứ tự

```
int binarysearch(int x, int *a, int left, int right)
{
    if(left > right)    return -1;
    int mid = (left + right)/2;
    if(x == a[mid])     return mid;
    if (x < a[mid])     return     binarysearch(x,a,left,mid-1);
    else return   binarysearch(x,a,mid+1,right);
}
```

Phương trình đệ quy : $T(n) = \begin{cases} 1 & n = 1 \\ T(n/2) + 1 & n > 1 \end{cases}$

Ví dụ 4. Phương trình đệ quy của thuật toán tìm kiếm nhị phân dãy n thứ tự



Giải phương trình đệ quy

- Có 3 phương pháp giải phương trình đệ quy:

(1) Phương pháp truy hồi.

- Triển khai T(n) theo T(n-1), rồi T(n-2), … cho đến T(1) hoặc T(0)

- Suy ra nghiệm

(2) Phương pháp đoán nghiệm.

- Dự đoán nghiệm f(n)

- Áp dụng định nghĩa tỷ suất tăng và chứng minh f(n) là tỷ suất tăng của T(n)

(3) Phương pháp lời giải tổng quát.

Phương pháp truy hồi

- Dùng đệ quy để thay thế T(m) với m < n (vào phía phải phương trình) cho đến khi tất cả T(m) với m>1 được thay thế bởi biểu thức của T(1) hoặc T(0).

Triển khai T(n) theo
T(n-1) rồi đến
T(n-2) tiếp đến
...
cho đến T(1)

- Vì T(1) và T(0) là hằng số nên công thức T(n) chứa các số hạng chỉ liên quan đến n và hằng số.
- Từ công thức đó suy ra nghiệm của phương trình.

Ví dụ 1. Giải phương trình đệ quy bằng phương pháp truy hồi (n!)

Hàm tính Giai thừa n !

$$T(n) = \begin{cases} C_1 & \text{nếu } n=0 \\ T(n-1) + C_2 & \text{nếu } n>0 \end{cases}$$

Ta có :

$T(n) = T(n-1) + C_2$

$T(n) = [T(n-2) + C_2] + C_2 = T(n-2) + 2C_2$

$T(n) = [T(n-3) + C_2] + 2C_2 = T(n-3) + 3C_2$

……

$T(n) = T(n-i) + iC_2$

- Quá trình truy hồi kết thúc khi n - i = 0 hay i = n.
- Khi đó ta có $T(n) = T(0) + nC_2 = C_1 + nC_2 =$ **O(n)**

Ví dụ 2. Giải phương trình đệ quy bằng phương pháp truy hồi (MergeSort)

Hàm MergeSort

$$T(n) = \begin{cases} C_1 & \text{nếu } n=1 \\ 2T(\frac{n}{2}) + nC_2 & \text{nếu } n>1 \end{cases}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + C_2 n$$

$$T(n) = 2\left[2T\left(\frac{n}{4}\right) + C_2\frac{n}{2}\right] + C_2 n = 4T\left(\frac{n}{4}\right) + 2C_2 n$$

$$T(n) = 4\left[2T\left(\frac{n}{8}\right) + C_2\frac{n}{4}\right] + 2C_2 n = 8T\left(\frac{n}{8}\right) + 3C_2 n$$

………………

$$T(n) = 2^i T\left(\frac{n}{2^i}\right) + iC_2 n$$

Quá trình truy hồi kết thúc khi $n/2^i = 1$ hay $2^i = n$ và do đó $i = \log n$.
Khi đó ta có: $T(n) = nT(1) + \log n C_2 n = C_1 n + C_2 n\log n =$ **O(nlogn)**.

CANTHO UNIVERSITY

Ví dụ 3. Giải phương trình đệ quy bằng phương pháp truy hồi (Tháp Hà nội)

$T(1)=C_1$, $T(n)=2T(n-1)+C_2$

$T(n) = 2T(n-1) + C2$

$= 2[2T(n-2) + C2] + C2 = 4T(n-2) + 3C2$

$= 4[2T(n-3) + C2] + 3C2 = 8T(n-3) + 7C2$

....

$= 2^i T(n-i) + (2^i - 1)C2$

Quá trình truy hồi dừng khi n - i = 1 hay i= n -1.

$\Rightarrow T(n) = 2^{n-1} T(1) + (2^{n-1} - 1)C2$)

$= 2^{n-1} C1 + (2^{n-1} - 1)C2 = \mathbf{O(2^n)}$

Bài tập

Giải các phương trình đệ quy sau với $T(1) = 1$ và:

1. $T(n) = T(n/2) + 1$

2. $T(n) = 3T(n/2) + n$

3. $T(n) = 4T(n/2)+ n^2$

Bài giải (1)

- **$T(1)=1, T(n)=T(n/2)+1$**

- $T(n)=T(n/2)+1$
  $= [T(n/2^2)+1]+1$
  $= T(n/2^2)+2$
  $= [T(n/2^3)+1]+2.....$
  $= T(n/2^3)+3$
  ....
  $= T(n/2^i)+i$

Quá trình truy hồi dừng lại khi $n/2^i = 1$ hay $2^i = n$ và do đó $i=\log n$.

=> $T(n) = T(1) + \log n =$ **O (log n)**

Bài giải (2)

- **$T(1)=1, T(n)=3T(n/2)+n$**

- $T(n)= 3T(n/2)+n$

  $= 3[3T(n/ 2^2)+n/2]+n =3^2 T(n/ 2^2)+5n/2$

  $= 3^2 [3T(n/ 2^3)+n/ 2^2]+5n/2 =3^3 T(n/ 2^3)+19n/ 2^2$

  ....

  $= 3^i T(n/ 2^i)+(3^i – 2^i)n/ 2^{i-1}$

Quá trình truy hồi dừng lại khi $n/2^i = 1$ hay $2^i = n$ và $i=\log n$.

$\Rightarrow T(n) = 3^{\log n} T(1)+(3^{\log n} – 2^{\log n})n/ 2^{\log n-1}$

$= 3^{\log n} + (3^{\log n} – 2^{\log n})n/ 2^{\log n-1}$

$=$ **$O(3^{\log n})$**

Bài giải (3)

- **$T(1)=1, T(n)=4T(n/2) + n^2$**

Giải phương trình đệ quy

$$T(1)=1$$

$$T(n)=4T\left(\frac{n}{2}\right)+n^2$$

Ta có:

$$T(n)=4T\left(\frac{n}{2}\right)+n^2=2^2T\left(\frac{n}{2}\right)+n^2=2^2\left(2^2T\left(\frac{n}{2^2}\right)+\left(\frac{n}{2}\right)^2\right)+n^2$$

$$=2^4T\left(\frac{n}{2^2}\right)+2n^2=2^4\left(2^2T\left(\frac{n}{2^3}\right)+\left(\frac{n}{2^2}\right)^2\right)+2n^2$$

$$=2^6T\left(\frac{n}{2^3}\right)+3n^2= \quad \ldots \quad =2^{2i}T\left(\frac{n}{2^i}\right)+in^2 \quad ,\forall i\in N^+$$

Quá trình trên kết thúc khi $n=2^i \Rightarrow i=\log n$. Khi đó,

$T(n)=2^{2\log n}T(1)+n^2\log n=2^{\log n^2}+n^2\log n=n^2+n^2\log n$.

Vậy độ phức tạp của thuật toán là $T(n)=O(n^2\log n)$.

Một số công thức Logarit

10. Logarit : $0 < N_1, N_2, N$ và $0 < a, b \neq 1$ ta có

$\log_a N = M \Leftrightarrow N = a^M$	$\log_a\left(\frac{N_1}{N_2}\right) = \log_a N_1 - \log_a N_2$
$\log_a a^M = M$	$\log_a N^\alpha = \alpha \log_a N$
$a^{\log_a N} = N$	$\log_{a^\alpha} N = \frac{1}{\alpha}\log_a N$
$N_1^{\log_a N_2} = N_2^{\log_a N_1}$	$\log_a N = \frac{\log_b N}{\log_b a}$
$\log_a(N_1.N_2) = \log_a N_1 + \log_a N_2$	$\log_a b = \frac{1}{\log_b a}$