- Cho mảng một chiều các số tự nhiên:
- 12, 7, 18, 15, 11, 5, 9, 21
- Minh hoạ giải thuật sắp xếp cho mảng trên bằng 3 thuật toán:
- Thuật toán sắp xếp nổi bọt (bubler sort)
- Thuật toán sắp xếp theo phương pháp chọn (Selection sort)
- Thuật toán sắp xếp theo phương pháp chèn (Insertion sort)

ThS.Nguyễn Thúy Loan

1. Viết chương trình nhập vào 1 dãy số nguyên
   (tối đa 1000 phần tử (nhập bằng random))
2. Xuất ra dãy số vừa nhập
3. Dùng cấu trúc swtch ( )
   chọn 1 trong 10 phương pháp sắp xếp.
4. Xuất ra dãy số sau khi sắp

1. Viết chương trình nhập quản lý SV
2. Thông tin về 1 SV gồm có : MSSV, Tên, lớp, giới tính, toan , lý .tin. (*Nhập tối đa 1000 phần tử)*
3. Xuất ra DS SV vừa nhập theo thứ tự tăng dần của cột MSSV (dùng PP chọn trực tiếp).
4 Xuất ra DS SV vừa nhập theo thứ tự tăng dần của cột tên (dùng PP chèn trực tiếp).
5. Xuất ra DS SV vừa nhập theo thứ tự tăng dần của cột toán (dùng PP quick sort).

Viết chương trình quản lý SV, thông tin về 1 SV gồm có : **MSSV,Tên, Giới tính, lớp, DTB.**

- Viết hàm nhập DS SV, lưu trên DSLKD 2 trỏ,(*Nhập cho đến chán thì thôi)*
- Xuất ra DS SV vừa nhập
- Xuất thông tin của sinh viên có tên “A”.
- Thêm một SV vào sau SV có điểm TB lớn nhất đầu tiên trong DS
- Xóa SV có điểm TB <5 đầu tiên trong DS
- Xuất ra DS SV theo thứ tự tăng dần của cột DTB (dùng PP chọn trực tiếp).
- Xuất ra DS SV vừa nhập theo thứ tự tăng dần của cột tên (dùng PP đổi chỗ trực tiếp).

- Vẽ cây nhị phân tìm kiếm nếu người ta lần lượt thêm vào các số dưới đây: 13, 12, 7, 28, 42, 31, 5, 9, 25, 18
- Vẽ cây nhị phân tìm kiếm ở câu 1 nếu xoá lần lượt 13 và 28
- Viết hàm tính chiều cao của cây nhị phân tìm kiếm
- Viết hàm đếm số nút lá ở mức k cho trước của cây nhị phân tìm kiếm
- Viết hàm đếm số nút có một con của cây nhị phân tìm kiếm

Đếm số nút lá

```
int  ktra ( Node *p )
{
        if ( p->pLeft == NULL && p->pRight == NULL) return 1;
        return 0 ;
}
//Dem nut la
int CountLeaf(Tree T)
 {
        if (T == NULL)  return 0 ;
        else
           if( ktra (T) == 1)  return 1;
           else
            return CountLeaf (T->pLeft) + CountLeaf (T->pRight);
     }
```

Chiều cao cây

```
int HeightTree(Tree T)
 {
        if( T == NULL )  return -1;
        else
        if(T->Left == NULL && T->Right == NULL )  return 0 ;
        else
        if(T->Left != NULL && T->Right != NULL)
         return Max(HeightTree(T->Left),HeightTree(T->Right)) +1;
        else
        if(T->Left == NULL)
           return 1 + HeightTree(T->Right);
        else
           return 1 + HeightTree(T->Left);
      }
```

2. Tìm cấu trúc dữ liệu để biểu diễn bài toán

Các tiêu chuẩn khi lựa chọn CTDL

- CTDL phải biểu diễn được đầy đủ các thông tin nhập và xuất của bài toán
- CTDL phải phù hợp với thao tác của thuật toán mà ta lựa chọn
- CTDL phải cài đặt được trên máy tính với ngôn ngữ lập trình đang sử dụng

1. Tìm thuật toán

Các đặc trưng của thuật toán

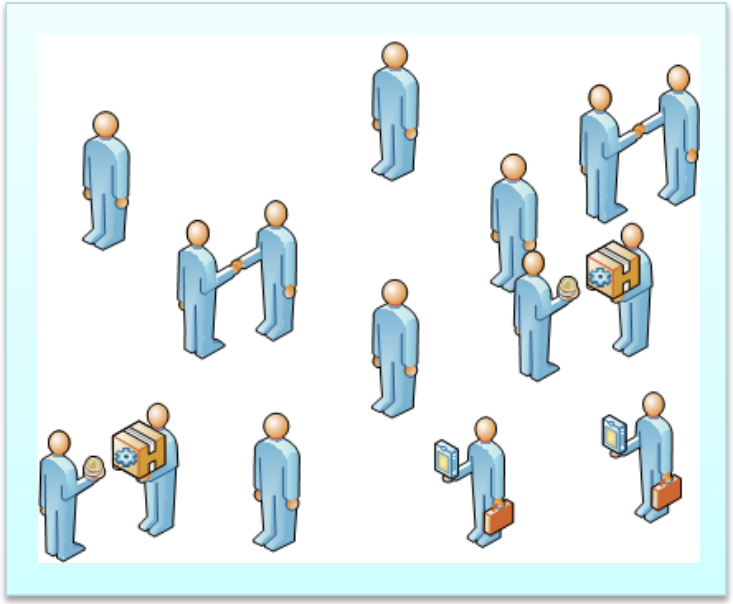
1. Tính đơn nghĩa
2. Tính dừng
3. Tính đúng
4. Tính phổ dụng
5. Tính khả thi

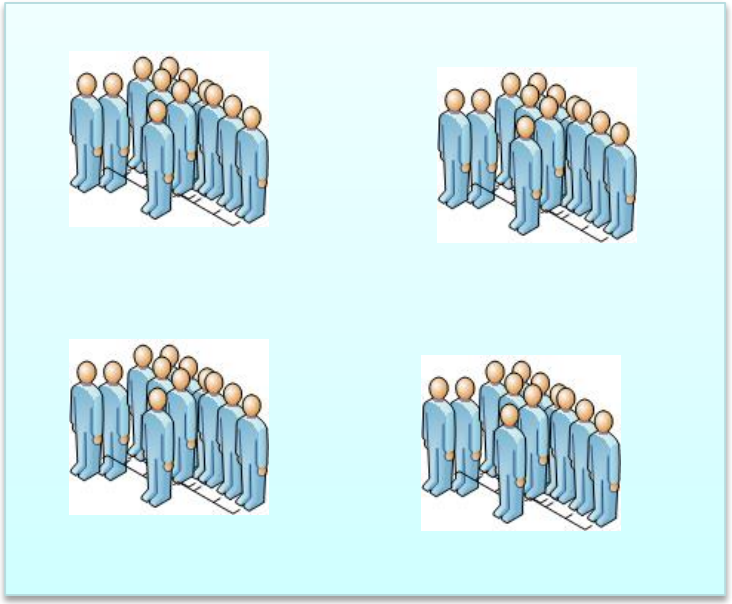
I. KHÁI NIỆM VỀ CẤU TRÚC DỮ LIỆU

1. Cấu trúc dữ liệu là gì ? :

I. KHÁI NIỆM VỀ CẤU TRÚC DỮ LIỆU

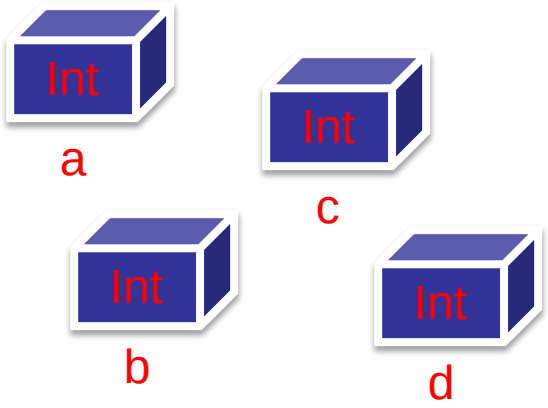
1. Cấu trúc dữ liệu là gì ? :

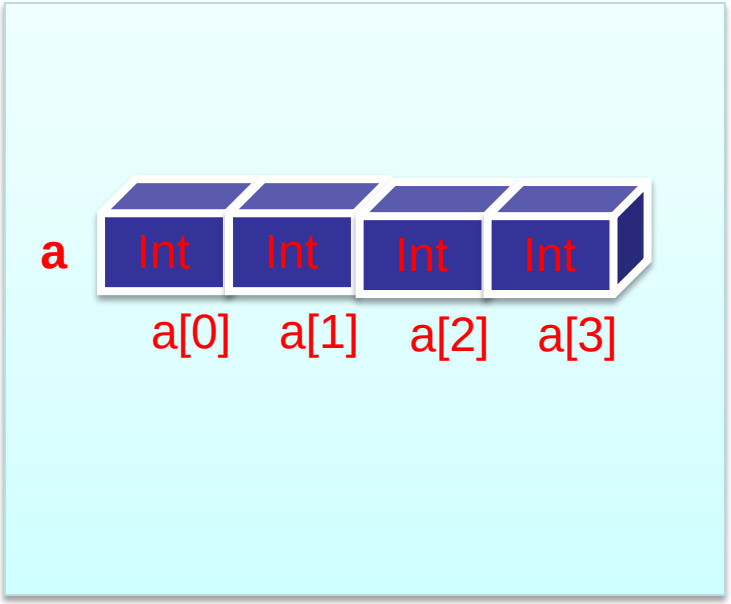




I. KHÁI NIỆM VỀ CẤU TRÚC DỮ LIỆU

1. Cấu trúc dữ liệu là gì ? :





- Hạt nhân của các chương trình máy tính là sự

Lưu trữ và Xử lý thông tin.

- Việc tổ chức dữ liệu như thế nào có ảnh hưởng rất lớn đến :

- *Cách thức xử lý dữ liệu đó*
- *Tốc độ thực thi*
- *Sự chiếm dụng bộ nhớ của chương trình.*

I. KHÁI NIỆM VỀ CẤU TRÚC DỮ LIỆU

1. Cấu trúc dữ liệu là gì ? :

- ❖ Tổ chức dữ liệu để lưu trữ.
- ❖ Mô hình dữ liệu để biểu diễn thông tin

❑ ***Dữ liệu không có cấu trúc (đơn giản):***

- ➢ Int, Char, Boolean, Float…
- ➢ Mỗi đối tượng dữ liệu là một phần tử đơn lẻ.

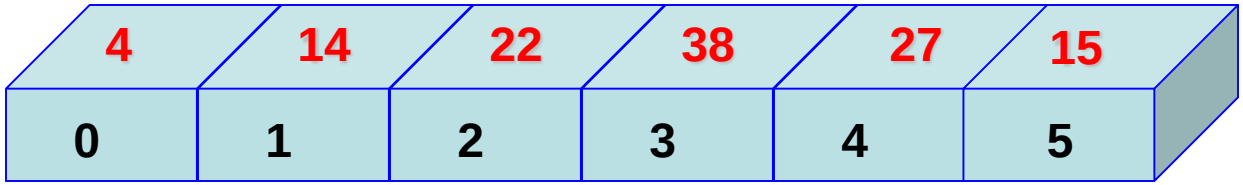
❑ ***Dữ liệu có cấu trúc:***

- ➢ Được cấu thành bởi các phần tử dữ liệu đơn giản.
- ➢ Mảng, Chuỗi, Danh sách, Tập tin.

I. KHÁI NIỆM VỀ CẤU TRÚC DỮ LIỆU

2. Một số ví dụ

A



Array 1 chiều

I. KHÁI NIỆM VỀ CẤU TRÚC DỮ LIỆU

2. Một số ví dụ

Cột

Dòng	0	1	2	3	4	5
0	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	[0][5]
1	[1][0]	[1][1]	[1][2]	[1][3]	[1][4]	[1][5]
2	[2][0]	[2][1]	[2][2]	[2][3]	[2][4]	[2][5]

Array 2 chiều

I. KHÁI NIỆM VỀ CẤU TRÚC DỮ LIỆU

2. Một số ví dụ

```
struct   <TenCauTruc >
 {
        <Kiểu> Biến 1;
        <Kiểu> Biến 2;
 };
```

Sinh viên (H_ Tên, MSSV, nsinh)

```
typedef   struct SINH_VIÊN
{
        char    H_Tên [ 255 ];
        char    MSSV  [3];
        int     năm_sinh;
} ;
```

Danh sách

I. KHÁI NIỆM VỀ CẤU TRÚC DỮ LIỆU

3. Vai trò cấu trúc dữ liệu trong lập trình

Cấu Trúc Dữ Liệu + Giải Thuật = Chương trình

(Data Structures + Algorithms = Program)

II. GIẢI THUẬT

1. Khái niệm giải thuật :

Khái niệm giải thuật hay thuật giải mà nhiều khi còn được gọi là thuật toán dùng để chỉ phương pháp hay cách thức (method) để giải quyết vần đề.

Giải thuật có thể được minh họa bằng ngôn ngữ tự nhiên (natural language), bằng lưu đồ (flow chart) hoặc bằng mã giả (pseudo code).

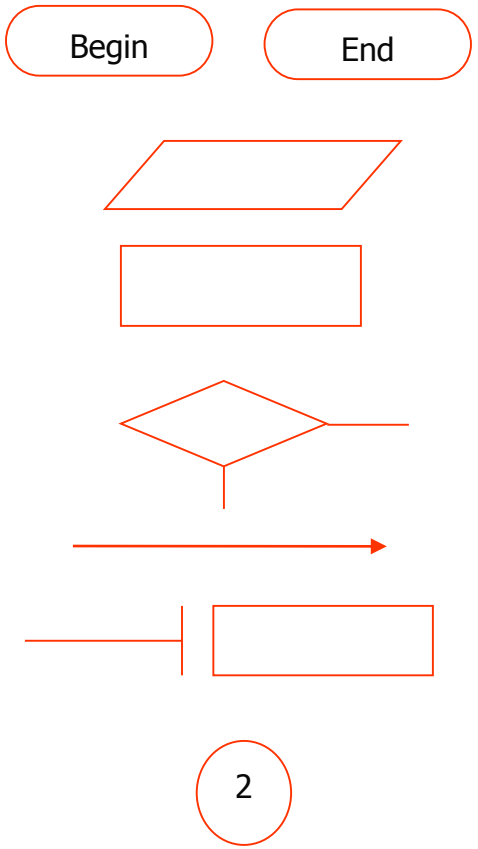
II. GIẢI THUẬT

2. Cách viết một giải thuật (Biểu diễn giải thuật)

A. **Ngôn ngữ tự nhiên**

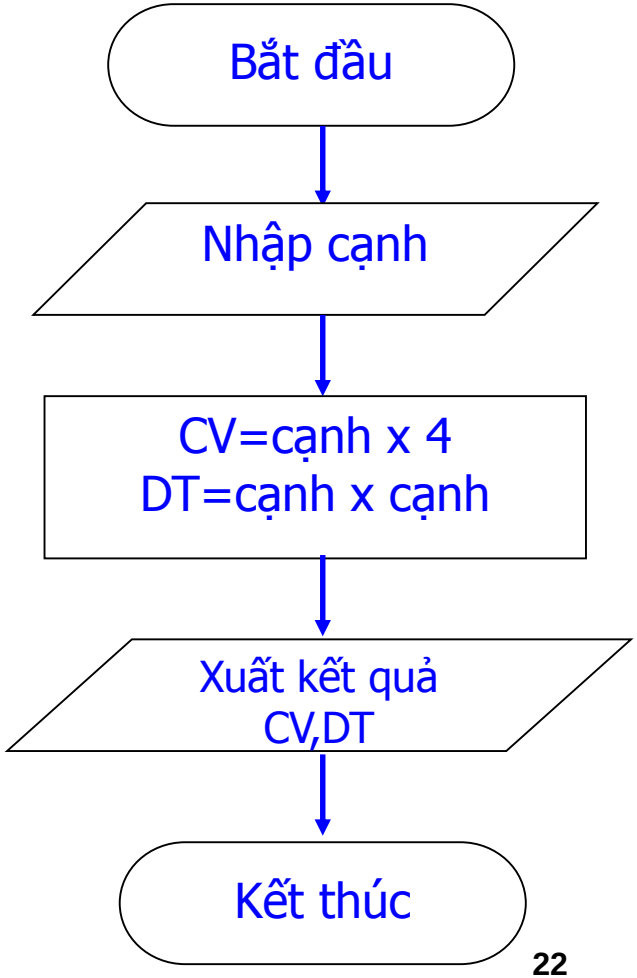
B. **Lưu đồ (flow chart)**

C. **Mã giả (pseudo code)**



- Điểm bắt đầu / Kết thúc giải thuật
- Thao tác nhập/ xuất dữ liệu
- Thao tác xử lý
- Thao tác lựa chọn
- Đường tiến trình
- Chú thích
- Ký hiệu kết nối cùng trang hay sang trang khác

Ví dụ: Hãy dùng lưu đồ mô tả bài toán tính Chu vi và Diện tích hình vuông khi người sử dụng cho biết số đo cạnh của nó.



III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

1. Dạng tổng quát

f (n)

n : kích cỡ đầu vào của dữ liệu

III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

2. Vòng lặp đơn

```
1   i = 1
2   while ( i <= 1000 )
    {
            application code
            i = i + 1
    }
```

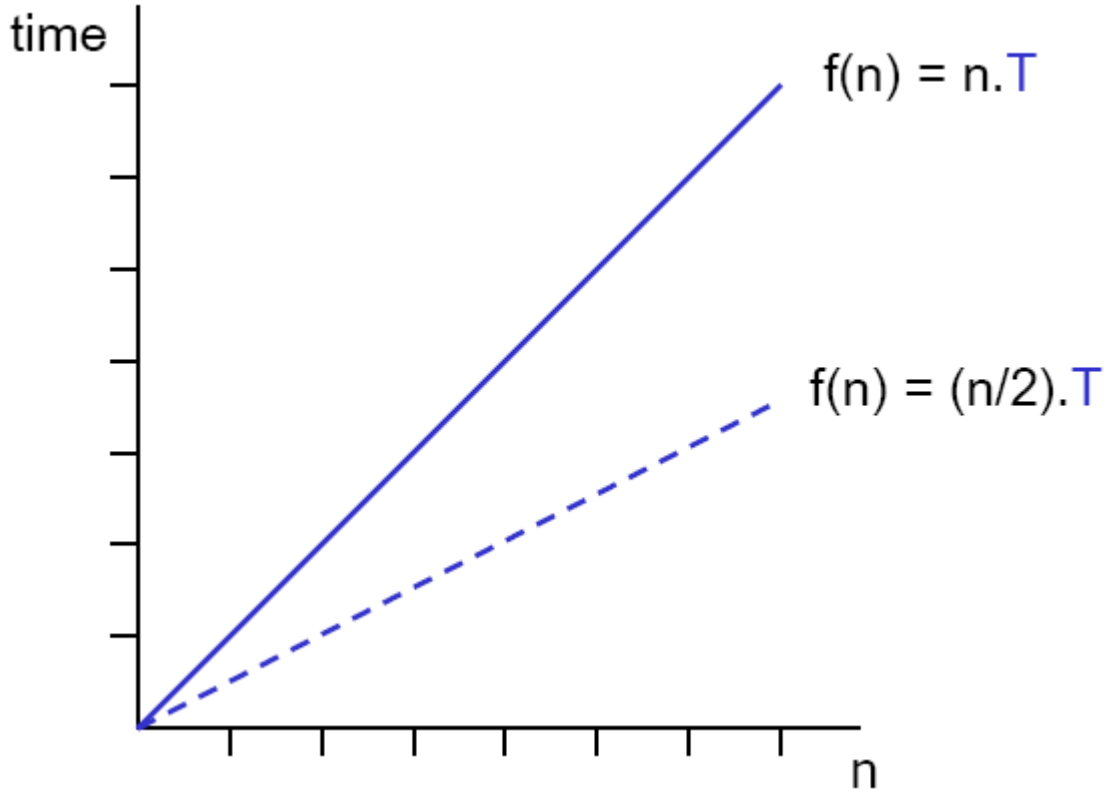
1000

```
1   i = 1
2   while ( i <= 1000 )
    {
            application code
            i = i + 2
    }
```

500

III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

2. Vòng lặp đơn



III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

2. Vòng lặp đơn

```
1   i = 1
2   while ( i <= 1000 )
    {
            application code
            i = i + 1
    }
```

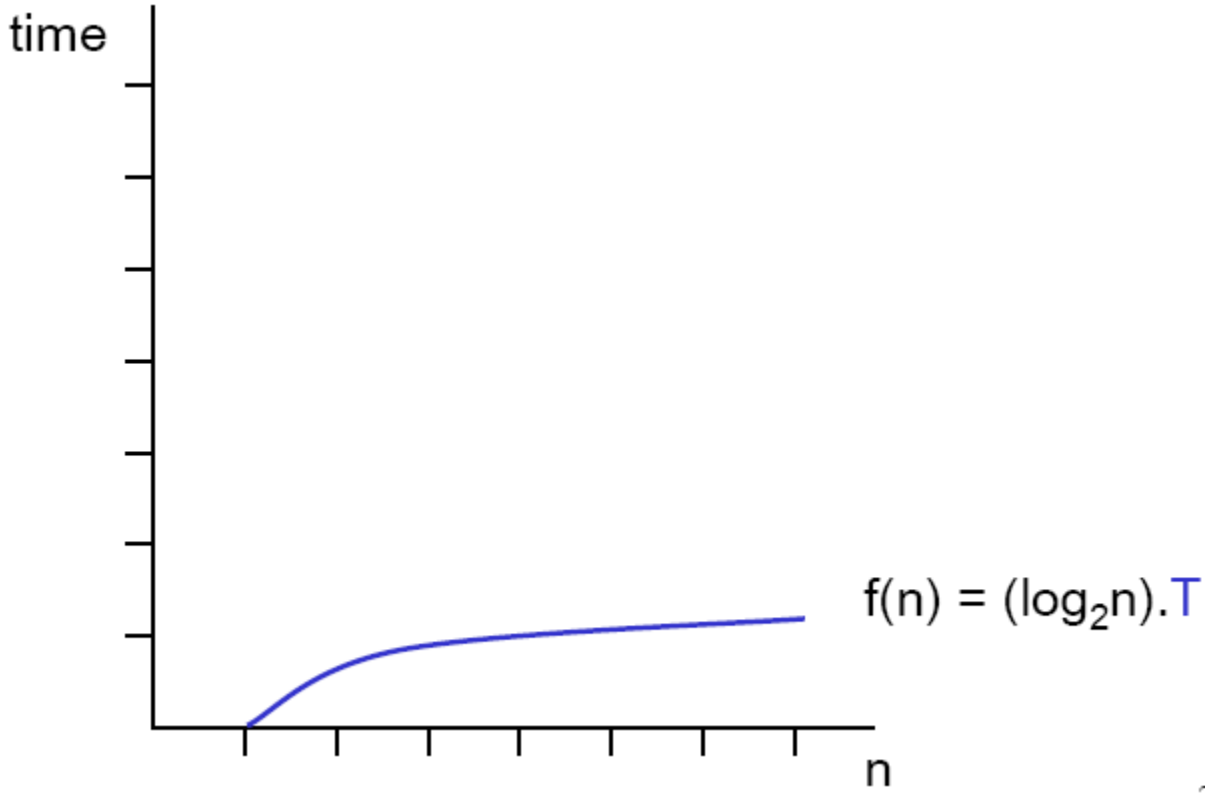
1000

```
1   i = 1
2   while ( i <= 1000 )
    {
            application code
            i = i x 2
    }
```

$\text{Log}_2\ 1000$

III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

2. Vòng lặp đơn



III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

3. Vòng lặp lồng nhau

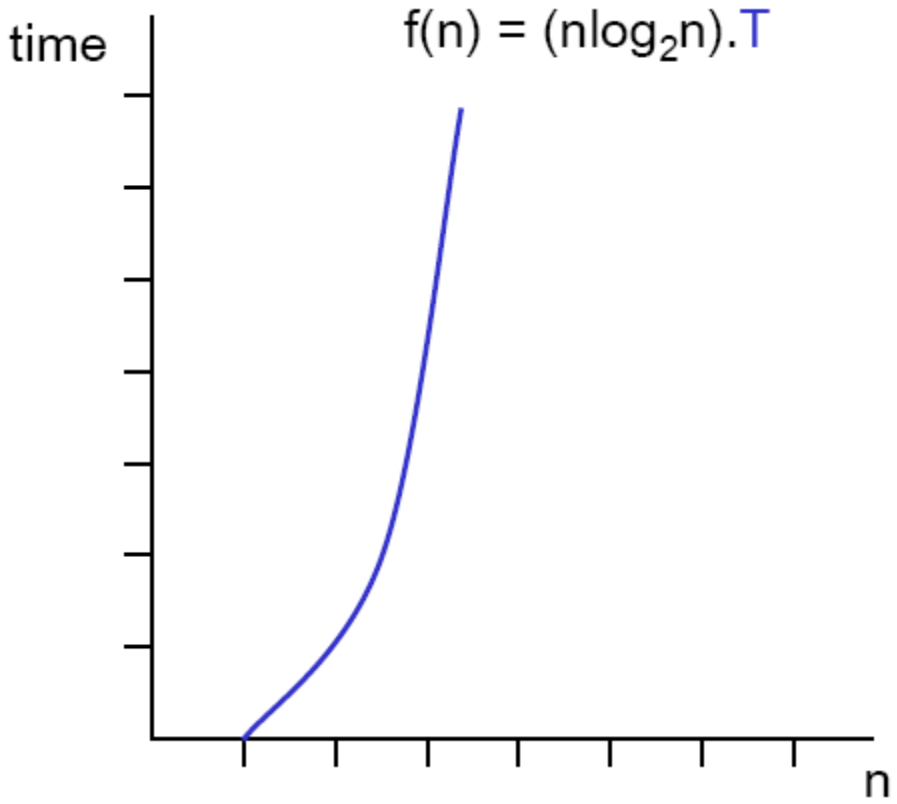
Số lần lặp = Outer loop x Inner loop

```
1   i = 1
2   While ( i <= 10 )
    {       j = 1
            while  (j <= 10)
            {           application code
                        j = j x 2
            }
    }
3   i = i + 1
```

$n \log_2 n$

III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

3. Vòng lặp lồng nhau



III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

4. Đánh giá tổng kết

$\log_2 n \quad n \quad n\log_2 n \quad n^2 \quad n^3 \quad \dots \quad n^k \quad \dots \quad 2^n \quad n!$

III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

4. Đánh giá tổng kết

n = 10,000

Efficiency	Big-O	Iterations	Est. Time
logarithmic	$O(\log_2 n)$	14	microseconds
linear	$O(n)$	10,000	.1 seconds
linear logarithmic	$O(n\log_2 n)$	140,000	2 seconds
quadratic	$O(n^2)$	$10{,}000^2$	15-20 min.
polynomial	$O(n^k)$	$10{,}000^k$	hours
exponential	$O(2^n)$	$2^{10{,}000}$	intractable
factorial	$O(n!)$	10,000!	intractable

III. ĐỘ PHỨC TẠP CỦA GIẢI THUẬT (Algorithm Efficiency)

4. Đánh giá tổng kết