ĐỘ PHỨC TẠP CỦA GIẢI THUẬT

Chương 1

NỘI DUNG

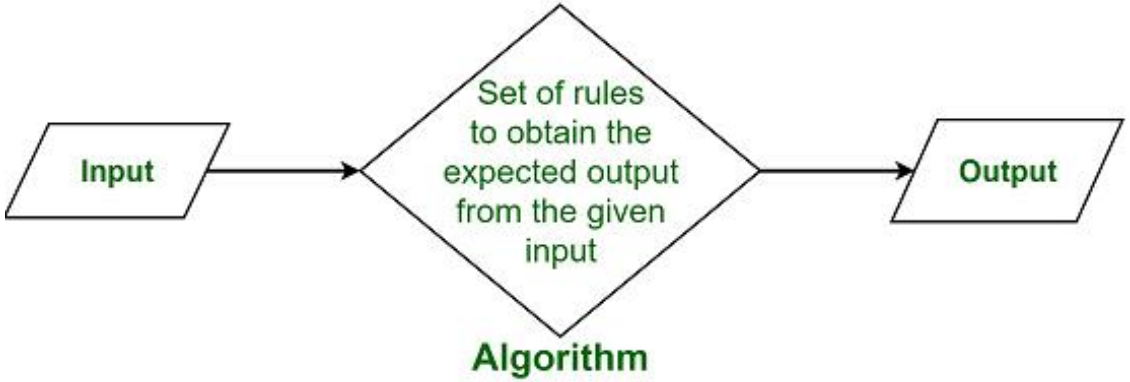
- Mô hình hóa
- Giải thuật
- Độ phức tạp của giải thuật
- Phương pháp tính độ phức tạp của giải thuật

MÔ HÌNH HÓA

- Mô hình là một dạng trừu tượng hóa của một hệ thống thực.
- Dùng mô hình để biểu diễn một hệ thống thực trong hệ thống máy tính.
  - Dữ liệu
  - Đối tượng
  - Chức năng
- Ví dụ: Bài toán tô màu bản đồ thế giới

GIẢI THUẬT LÀ GÌ?

- Giải thuật là chuỗi hữu hạn các thao tác để giải một bài toán nào đó.
- Độc lập với ngôn ngữ lập trình
- Tính chất của giải thuật:
  - Hữu hạn
  - Xác định
  - Hiệu quả

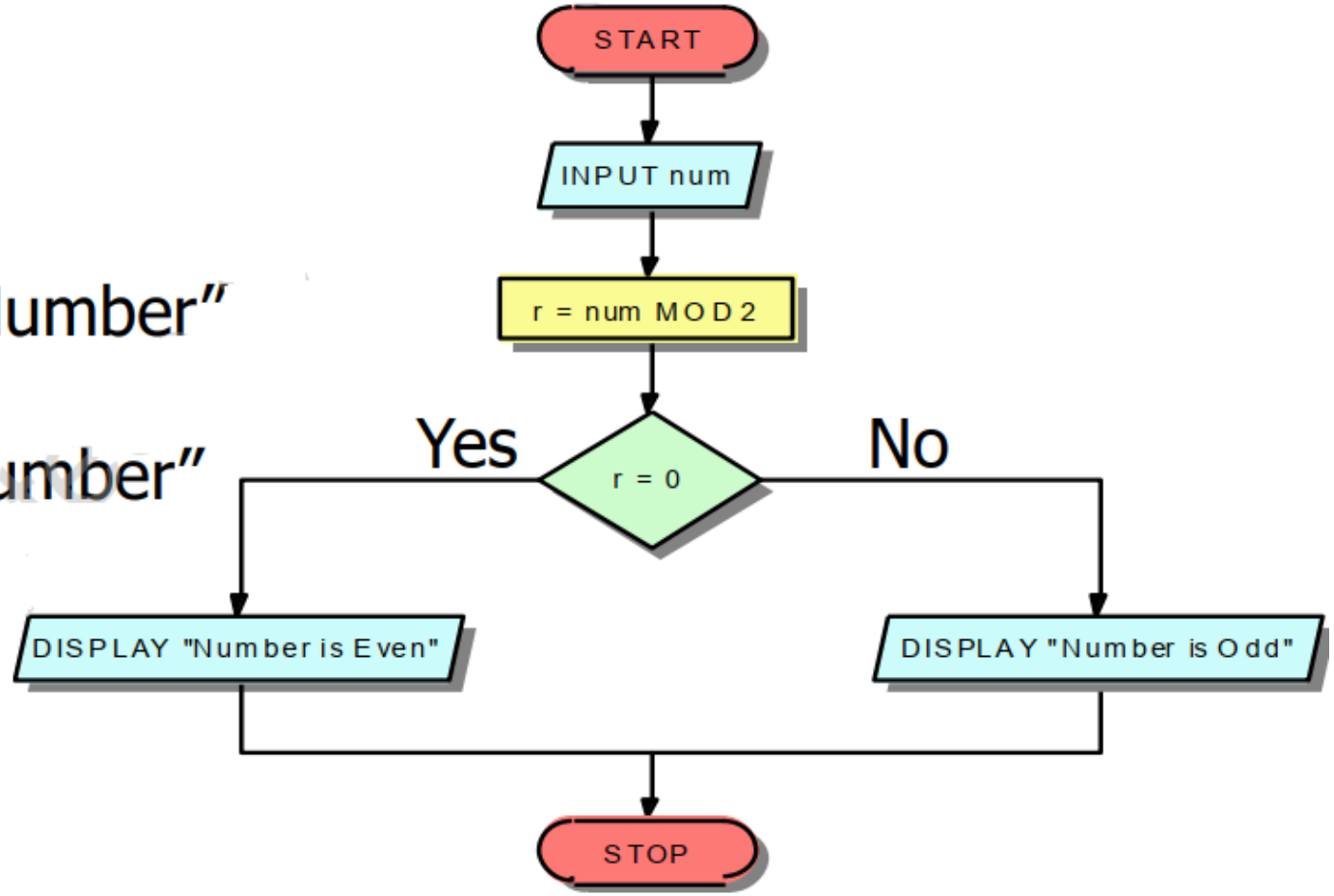


GIẢI THUẬT LÀ GÌ?

- Biểu diễn giải thuật:
  - Ngôn ngữ tự nhiên
  - Mã giả
  - Lưu đồ xử lý (flowchart)
  - Sơ đồ hoạt động (Activity diagram)
  - Ngôn ngữ lập trình

GIẢI THUẬT LÀ GÌ?

```
BEGIN
INPUT num
r=num MOD 2
IF r=0
   DISPLAY "Even Number"
ELSE
   DISPLAY "Odd Number"
END IF
END
```



CẤU TRÚC DỮ LIỆU LÀ GÌ?

- Dữ liệu:
  - Biểu diễn thông tin của bài toán
  - Là đối tượng mà các giải thuật cần xử lý
- Cấu trúc dữ liệu: là tập hợp các dữ liệu được tổ chức thành một cấu trúc có các mối quan hệ với nhau

ĐÁNH GIÁ GIẢI THUẬT (1-2)

- Tiêu chí đánh giá giải thuật:
  - Bộ nhớ → Độ phức tạp về bộ nhớ
  - Thời gian → Độ phức tạp về thời gian
- Cần phải phân tích, đánh giá giải thuật để:
  - Lựa chọn một giải thuật tốt nhất trong các giải thuật để cài đặt chương trình giải quyết bài toán đặt ra.
  - Cải tiến giải thuật hiện có để được một giải thuật tốt hơn.

ĐÁNH GIÁ GIẢI THUẬT (2-2)

- Thời gian thực thi giải thuật phụ thuộc vào:
  - Kích thước dữ liệu
  - Kiểu câu lệnh, tốc độ xử lý của máy tính, ngôn ngữ lập trình, trình biên dịch, ….
- Độ phức tạp của giải thuật về thời gian: là cách đánh giá thời gian thực hiện giải thuật độc lập với máy tính và các yếu tố liên quan máy tính.

THỜI GIAN THỰC HIỆN CHƯƠNG TRÌNH

- Là tổng thời gian / số phép tính toán mà chương trình cần để thực thi chương trình.
- Thời gian thực hiện một chương trình là một hàm của kích thước dữ liệu vào, ký hiệu T(n) trong đó n là kích thước (độ lớn) của dữ liệu vào.
- Ví dụ : Chương trình tính tổng của n số có thời gian thực hiện là T(n) = cn trong đó c là một hằng số.
- Thời gian thực hiện chương trình là một hàm không âm, tức là T(n) >= 0 $\forall$ n >= 0.

THỜI GIAN THỰC HIỆN CHƯƠNG TRÌNH

- Ví dụ:

```
int n = 1000; 1
for(i=1; i<=n; i++)          1  n  n
    printf("%d", i);n
```

→ **T(n) = 1 + 1 + n + n + n = 2 + 3n**

THỜI GIAN THỰC HIỆN CHƯƠNG TRÌNH

- Thời gian thực hiện chương trình không chỉ phụ thuộc vào kích thước mà còn phụ thuộc vào tính chất của dữ liệu vào.
- Vì vậy thường ta coi T(n) là thời gian thực hiện chương trình trong trường hợp xấu nhất trên dữ liệu vào có kích thước n, tức là: T(n) là thời gian lớn nhất để thực hiện chương trình đối với mọi dữ liệu vào có cùng kích thước n.

THỜI GIAN THỰC HIỆN CHƯƠNG TRÌNH

- Trong những ứng dụng thực tiễn, chúng ta không cần biết chính xác hàm T(n) mà chỉ cần biết một ước lượng đủ tốt của nó.
- Ước lượng thường dùng nhất là ước lượng tiệm cận trên Big-O, Big-Omega, Big-Theta

BIG-O (1)

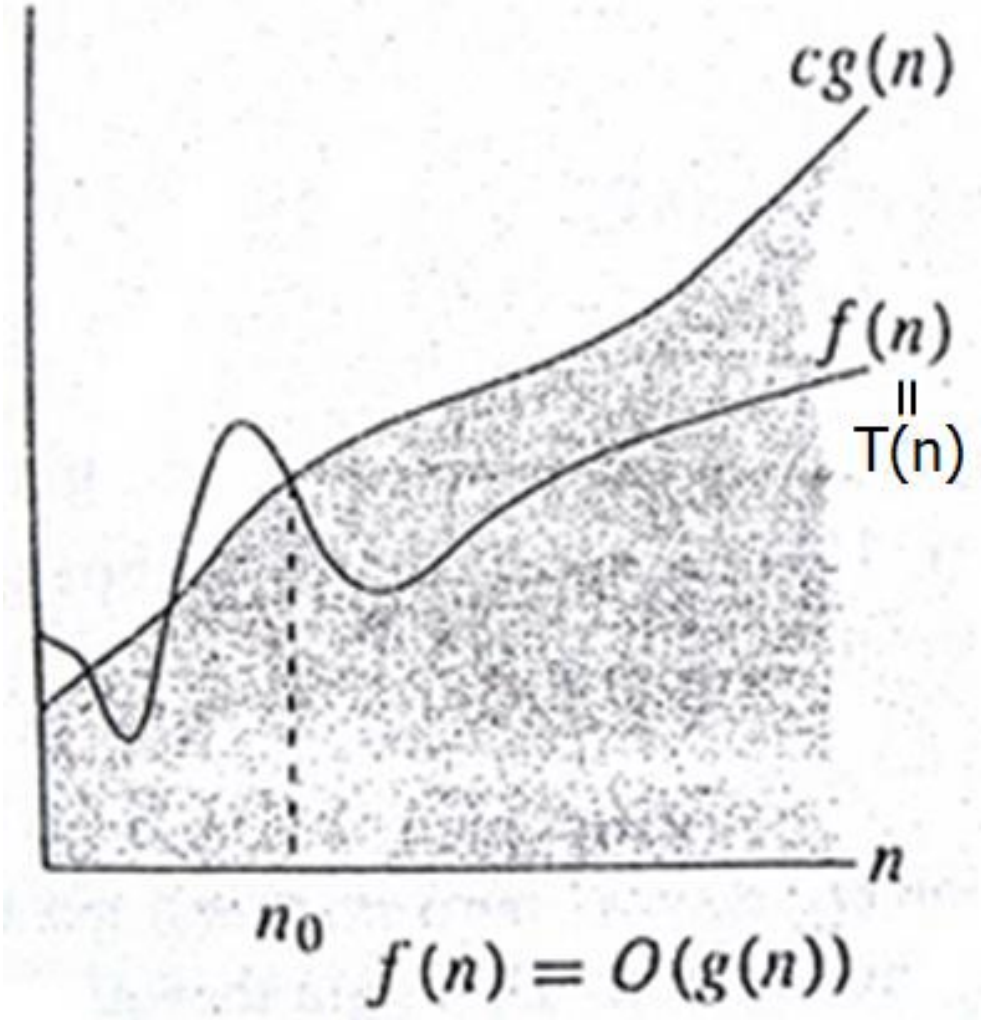
- Giả sử f(n) và g(n) là các hàm thực không âm của đối số nguyên không âm n.
- Ta nói "g(n) là O của f(n)"

  và viết là: g(n) = O(f(n))

  nếu tồn tại các hằng số dương C

  và n_0 sao cho f(n) <= C * g(n)

  với mọi n >= n0



BIG-O (2)

- Chú ý: O(C.f(n))=O(f(n))

  với C là hằng số.

  Đặc biệt O(C)=O(1)

- Ví dụ:

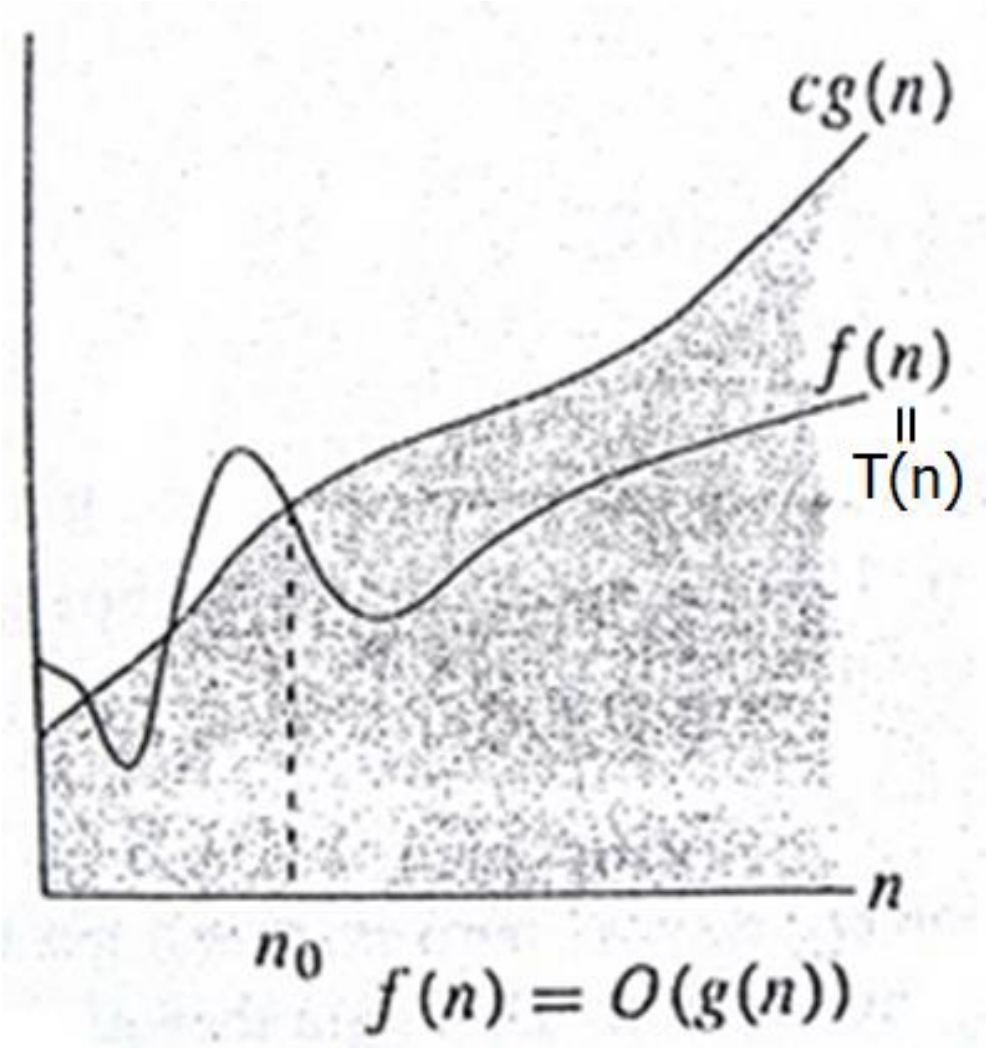
  $f(n) = 10n^2 + 100$

  ➔ $f(n) <= 10n^2 + 100n^2$ (n>=1)

  ➔ $f(n) <= 110n^2$ (n>=1)

  Chọn c = 110, $n_0 = 1$

  Vậy $f(n) = O(n^2)$



BIG-O (3)

Tính Big-O cho các hàm sau:

$T(n) = 7n-2$ ➔ $T(n) = O(n)$

$T(n) = 5n^3 + 600n^2 + 10000$ ➔ $T(n) = O(n3)$

$T(n) = 4\log n + 1000$ ➔ $T(n) = O(\log n)$

BIG-O (4)

CÁCH TÍNH BIG-0 của giải thuật:

- Tính tổng thời gian thực hiện giải thuật T(n)
- Loại tất cả hệ số, chọn số hạng có bậc cao nhất

BIG-O (5)

- Ví dụ:

```
int n = 1000;
for(i=1; i<=n; i++)
    printf("%d", i);
```

→ **T(n) = 1 + 1 + n + n + n = 2 + 3n**

→ Độ phức tạp: T(n) = **O (n)**

BIG-O (6)

- Độ lớn của dữ liệu đầu vào → độ phức tạp.

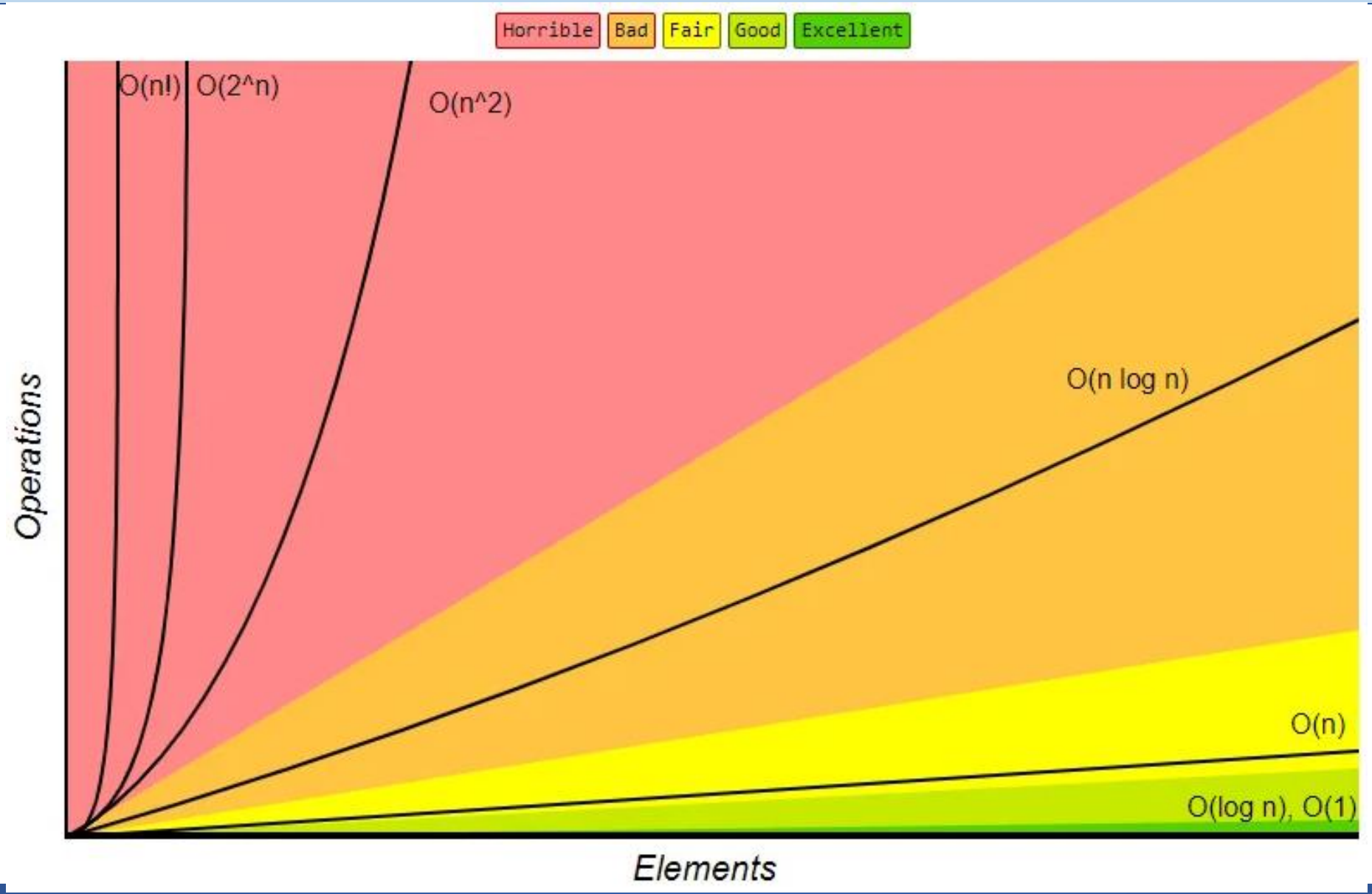
Big O	Name	Mô tả
O(1)	Constant	Số phép tính là hằng số, không phụ thuộc vào dữ liệu đầu vào. vd: a + b.
O(log n)	Logarithmic	Số phép tính (thời gian thực hiện) tăng theo kích thước dữ liệu theo hàm logarit. vd: tìm kiếm nhị phân, tìm ước số chung lớn nhất.
O(n)	Linear	Số phép tính phụ thuộc vào dữ liệu đầu vào theo 1 biến. vd: tìm kiếm tuyến tính, tính tổng các phần tử của một mảng một chiều.
O(n log n)	Linearithmic	Tổng hợp của n và logn (lồng nhau). vd: Heap sort, Merge sort.
$O(n^2)$	Quadratic	Vòng lặp lồng nhau. Ví dụ tìm xem trong tập hợp có phần tử trùng nhau không.
$O(n^3)$	Cubic	Ba vòng lặp lồng nhau. Ví dụ tính bài toán: 3*x + 5*y + 9 * z = 100
$O(2^n)$	Exponential	Thời gian chạy theo cấp số nhân (cơ số 2) có nghĩa là các phép tính được thực hiện bởi một thuật toán sẽ tăng gấp đôi mỗi khi đầu vào tăng lên.
O(n!)	Factorial	Thời gian chạy giai thừa như hoán đổi một chuỗi.

ĐỘ PHỨC TẠP CỦA GIẢI THUẬT - O (2)

- Các hàm thể hiện độ phức tạp có các dạng thường gặp sau: 1, logn, n, nlogn, n^2, n^3, **2n, n!, n^n**.

- Ba hàm cuối cùng ta gọi là dạng hàm mũ, các hàm khác gọi là hàm đa thức. Một giải thuật có độ phức tạp là một hàm đa thức thì chấp nhận được, còn các giải thuật có độ phức tạp hàm mũ thì phải tìm cách cải tiến giải thuật.

BIG-O (7)


Horrible
Bad
Fair
Good
Excellent
O(n!)
O(2^n)
O(n^2)
O(n log n)
O(n)
O(log n), O(1)
Operations
Elements

BIG-O: CÁC QUY TẮC (8)

Quy tắc: n là độ lớn dữ liệu đầu vào

1. Quy tắc bỏ hằng số:

$T(n) = O(c.f(n)) = O(f(n))$ với c là một hằng số dương

BIG-O: CÁC QUY TẮC (9)

2. Quy tắc cộng (tuần tự):

$$T1(n) = O(f(n)) , \qquad T2(n) = O(g(n))$$

$$T(n) = T1(n) + T2(n) = O(f(n) + g(n))$$

3. Quy tắc lấy giá trị max:

$$T(n) = O(f(n)+ g(n)) = O(max(f(n), g(n)))$$

BIG-O: CÁC QUY TẮC (10)

4. Quy tắc nhân (lồng nhau):

$T(n)=O(f(n))$, T(n) thực hiện k(n) lần; $k(n) = O(g(n))$

→ $T(n) = O(g(n).f(n))$

BIG-O (9)

Lưu ý:

- Thời gian thực hiện mỗi lệnh gán, nhập, xuất, so sánh, phép tính toán cơ bản, … là O(1)
- Thời gian thực hiện mệnh đề if…else là thời gian lớn nhất thực hiện khối lệnh trong if hoặc trong else. Thời gian kiểm tra điều kiện là O(1)
- Thời gian thực hiện chuỗi lệnh tuần tự được xác định bằng quy tắc cộng. Tức là thời gian thực hiện một lệnh lâu nhất trong chuỗi lệnh.
- Thời gian thực hiện vòng lặp là tổng số lần thực hiện khối lệnh trong thân vòng lặp. Nếu thời gian thực hiện khối lệnh trong thân vòng lặp không đổi thì thời gian thực hiện vòng lặp là tích của số bước thực hiện vòng lặp và thời gian thực hiện thân vòng lặp.

BÀI TẬP: TÍNH BIG-O

```
int firstElement(arr[])
{
        return arr[0]; 1
}
```

→ T(n) = **O (1)**

BÀI TẬP: TÍNH BIG-O

```
int arr[n];                    // 1
for(i=0; i<n; i++)             // 1, n, n
    s = s + arr[i];            // n, n
```

→ T(n) =1+1+n+n+n+n = **O (n)**

BÀI TẬP: TÍNH BIG-O

```
public void tinhtong(int[][] a)
{
        int tong=0;                                   ----------1
        for( int i=0;i<n;i++)                         ----------n
                for (int j=0;j<n;j++)                 --------n
                        tong+=a[i][j];
        return tong;                                  ----------1
}
```

→ T(n) = **O (n^2)**

BÀI TẬP: TÍNH BIG-O

```
int dem(int n)
{
        int count = 0;
        for(i=1; i<n; i*=2)
                count++;
        return count;
}
```

→ T(n) = **O (logn)**

BÀI TẬP: TÍNH BIG-O

```
for(i=1;  i<=n;  i++)
        printf("%d", i*2);
for(j=1;  j<=n;  j++)
        printf("%d", j*3);
```

→ T(n) = **O (n)**

BÀI TẬP: TÍNH BIG-O

```
int arr1[n], int arr2[n];
for(i=0; i<n; i++)
    for(j=n-1; j>=0; j--)
        arr2[j] = arr1[i];
for(i=0;i<n; i++)
    printf("%d ", arr2[i]);
```

→ T(n) = **O (n^2)**

BÀI TẬP: TÍNH BIG-O

```
Sum=0;
for(i=1;i<=n;i++)  {
 scanf(“%d”,&x);
 Sum=Sum+x;
}
```

→ T(n) = **O (n)**

BÀI TẬP: TÍNH BIG-O

```
Sum=0;
for(i=1;i<=n;i++)  {
 if (i%2==0)
  Sum=Sum+i;
}
```

→ T(n) = **O (n)**

BÀI TẬP: TÍNH BIG-O

```
Sum=0;
for(i=1;i<=n;i=i*2)
{
  scanf(“%d”,&x);
  Sum=Sum+x;
}
```

→ T(n) = **O (logn)**

BÀI TẬP: TÍNH BIG-O

```
Sum1=0;
k=1;
while (k<=n)
{
    for(j=1;j<=n;j++)
        Sum1=Sum1+1;
    k=k*2;
}
```

→ T(n) = **O (nlogn)**

BÀI TẬP: TÍNH BIG-O

```
s = 0;
for (i=0; i<=n;i++){
    p =1;
    for (j=1;j<=i;j++)
        p=p * x / j;
    s = s+p;
}
```

→ T(n) = **O (n^2)**

BÀI TẬP: TÍNH BIG-O

```
s = 1; p = 1;
for (i=1;i<=n;i++) {
    p = p * x / i;
    s = s + p;
}
```

→ T(n) = **O (n)**

TÓM TẮT

- Mô hình hóa
- Giải thuật
- Độ phức tạp của giải thuật
- Phương pháp tính độ phức tạp của giải thuật