

# **CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN**

*Data Structures and Algorithms*

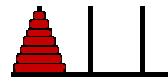


## **NguyỄN ĐỨC NGHĨA**

**Bộ môn Khoa học Máy tính  
Đại học Bách khoa Hà nội**

**Tel: 0438696121 (Off), 0903210111 (Mob)**

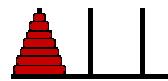
 [nghiand@soict.hut.edu.vn](mailto:nghiand@soict.hut.edu.vn)



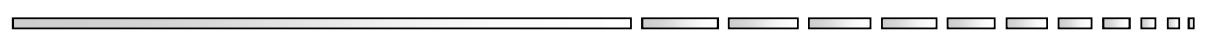
## Chương 1

# CÁC KIẾN THỨC CƠ BẢN

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



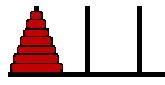
## NỘI DUNG



### 1.1. Ví dụ mở đầu

- 1.2. Thuật toán và độ phức tạp
- 1.3. Ký hiệu tiệm cận
- 1.4. Giả ngôn ngữ
- 1.5. Một số kỹ thuật phân tích thuật toán

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ mở đầu

- Bài toán tìm dãy con lớn nhất:

Cho dãy số

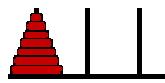
$$a_1, a_2, \dots, a_n$$

Dãy số  $a_i, a_{i+1}, \dots, a_j$  với  $1 \leq i \leq j \leq n$  được gọi là **dãy con** của dãy đã cho và  $\sum_{k=i}^j a_k$  được gọi là **trọng lượng** của dãy con này

**Bài toán đặt ra là:** Hãy tìm trọng lượng lớn nhất của các dãy con, tức là tìm cực đại giá trị  $\sum_{k=i}^j a_k$ . Để đơn giản ta gọi dãy con có trọng lượng lớn nhất là **dãy con lớn nhất**.

- **Ví dụ:** Nếu dãy đã cho là -2, **11**, **-4**, **13**, -5, 2 thì cần đưa ra câu trả lời là 20 (là trọng lượng của dãy con 11, -4, 13)

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Thuật toán trực tiếp

- Thuật toán đơn giản đầu tiên có thể nghĩ để giải bài toán đặt ra là: Duyệt tất cả các dãy con có thể

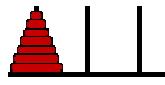
$$a_i, a_{i+1}, \dots, a_j \text{ với } 1 \leq i \leq j \leq n$$

và tính tổng của mỗi dãy con để tìm ra trọng lượng lớn nhất.

- Trước hết nhận thấy rằng, tổng số các dãy con có thể của dãy đã cho là

$$C(n,2) + n = n^2/2 + n/2 .$$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

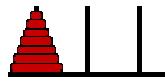


# Thuật toán trực tiếp

- Thuật toán này có thể cài đặt trong đoạn chương trình sau:

```
int maxSum = 0;
for (int i=0; i<n; i++) {
    for (int j=i; j<n; j++) {
        int sum = 0;
        for (int k=i; k<=j; k++)
            sum += a[k];
        if sum > maxSum
            maxSum = sum;
    }
}
```

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



# Thuật toán trực tiếp

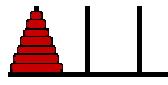
- **Phân tích thuật toán:** Ta sẽ tính số lượng phép cộng mà thuật toán phải thực hiện, tức là đếm xem dòng lệnh

Sum += a[k]

phải thực hiện bao nhiêu lần. Số lượng phép cộng sẽ là

$$\begin{aligned} \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} (j-i+1) &= \sum_{i=0}^{n-1} (1+2+\dots+(n-i)) = \sum_{i=0}^{n-1} \frac{(n-i)(n-i+1)}{2} \\ &= \frac{1}{2} \sum_{k=1}^n k(k+1) = \frac{1}{2} \left[ \sum_{k=1}^n k^2 + \sum_{k=1}^n k \right] = \frac{1}{2} \left[ \frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} \right] \\ &= \frac{n^3}{6} + \frac{n^2}{2} + \frac{n}{3} \end{aligned}$$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



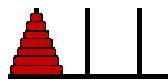
## Thuật toán nhanh hơn

- Để ý rằng tổng các số hạng từ  $i$  đến  $j$  có thể thu được từ tổng của các số hạng từ  $i$  đến  $j-1$  bởi 1 phép cộng, cụ thể là ta có:

$$\sum_{k=i}^j a[k] = a[j] + \sum_{k=i}^{j-1} a[k]$$

- Nhận xét này cho phép rút bớt vòng lặp for trong cùng.

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

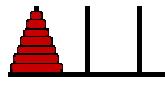


## Thuật toán nhanh hơn

- Ta có thể cài đặt như sau

```
int maxSum = a[0];
for (int i=0; i<n; i++) {
    int sum = 0;
    for (int j=i; j<n; j++) {
        sum += a[j];
        if sum > maxSum
            maxSum = sum;
    }
}
```

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



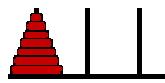
## Thuật toán nhanh hơn

- 
- **Phân tích thuật toán.** Ta lại tính số lần thực hiện phép cộng và thu được kết quả sau:

$$\sum_{i=0}^{n-1} (n-i) = n + (n-1) + \dots + 1 = \frac{n^2}{2} + \frac{n}{2}$$

- Để ý rằng số này là đúng bằng số lượng dãy con. Dường như thuật toán thu được là rất tốt, vì ta phải xét mỗi dãy con đúng 1 lần.

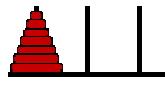
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Thuật toán đệ qui

- 
- Ta còn có thể xây dựng thuật toán tốt hơn nữa! Ta sẽ sử dụng kỹ thuật chia để trị. Kỹ thuật này bao gồm các bước sau:
    - Chia bài toán cần giải ra thành các bài toán con cùng dạng
    - Giải mỗi bài toán con một cách đệ qui
    - Tổ hợp lời giải của các bài toán con để thu được lời giải của bài toán xuất phát.
  - Áp dụng kỹ thuật này đối với bài toán tìm trọng lượng lớn nhất của các dãy con: Ta chia dãy đã cho ra thành 2 dãy sử dụng phần tử ở chính giữa và thu được 2 dãy số (gọi tắt là dãy bên trái và dãy bên phải) với độ dài giảm đi một nửa.

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Thuật toán đệ qui

- Để tổ hợp lời giải, nhận thấy rằng chỉ có thể xảy ra một trong 3 trường hợp:
  - Dãy con lớn nhất nằm ở dãy con bên trái (nửa trái)
  - Dãy con lớn nhất nằm ở dãy con bên phải (nửa phải)
  - Dãy con lớn nhất bắt đầu ở nửa trái và kết thúc ở nửa phải (giữa).
- Do đó, nếu ký hiệu trọng lượng của dãy con lớn nhất ở nửa trái là  $w_L$ , ở nửa phải là  $w_R$  và ở giữa là  $w_M$  thì **trọng lượng cần tìm sẽ là**

$$\max(w_L, w_R, w_M).$$

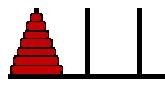
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Thuật toán đệ qui

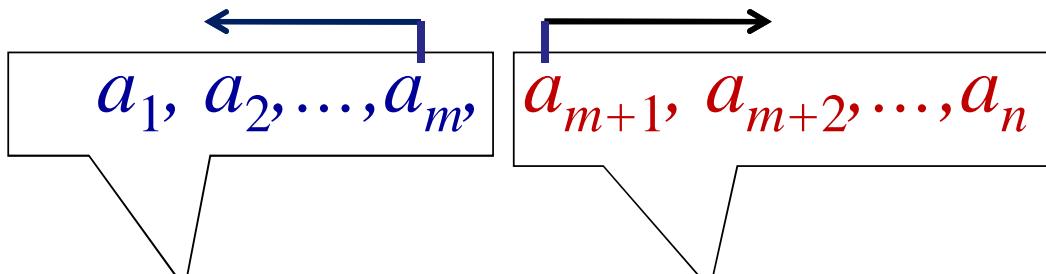
- Việc tìm trọng lượng của dãy con lớn nhất ở nửa trái ( $w_L$ ) và nửa phải ( $w_R$ ) có thể thực hiện một cách đệ qui
- Để tìm trọng lượng  $w_M$  của dãy con lớn nhất bắt đầu ở nửa trái và kết thúc ở nửa phải ta thực hiện như sau:
  - Tính trọng lượng của dãy con lớn nhất trong nửa trái kết thúc ở điểm chia ( $w_{ML}$ ) và
  - Tính trọng lượng của dãy con lớn nhất trong nửa phải bắt đầu ở điểm chia ( $w_{MR}$ ).
  - Khi đó  $w_M = w_{ML} + w_{MR}$ .

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Thuật toán đệ quy

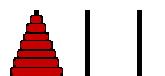
- $m$  – điểm chia của dãy trái,  $m+1$  là điểm chia của dãy phải



Tính  $W_{ML}$  của dãy con lớn nhất trong nửa trái kết thúc tại  $a_m$

Tính  $W_{MR}$  của dãy con lớn nhất trong nửa phải bắt đầu từ  $a_{m+1}$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

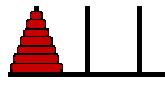


## Thuật toán đệ quy

- Để tính trọng lượng của dãy con lớn nhất ở nửa trái (từ  $a[i]$  đến  $a[j]$ ) kết thúc ở  $a[j]$  ta dùng thuật toán sau:

```
MaxLeft(a, i, j);
{
    maxSum = -∞; sum = 0;
    for (int k=j; k>=i; k--) {
        sum = sum+a[k];
        maxSum = max(sum, maxSum);
    }
    return maxSum;
}
```

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

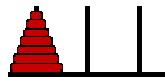


# Thuật toán đệ qui

- Để tính trọng lượng của dãy con lớn nhất ở nửa phải (từ  $a[i]$  đến  $a[j]$ ) bắt đầu từ  $a[i]$  ta dùng thuật toán sau:

```
MaxRight(a, i, j);
{
    maxSum = -∞; sum = 0;
    for (int k=i; k<=j; k++){
        sum = sum+a[k];
        maxSum = max(sum, maxSum);
    }
    return maxSum;
}
```

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

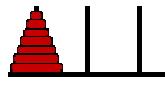


# Thuật toán đệ qui

- Sơ đồ của thuật toán đệ qui có thể mô tả như sau:

```
MaxSub(a, i, j);
{
    if (i = j) return a[i]
    else
    {
        m = (i+j)/2;
        wL = MaxSub(a, i, m);
        wR = MaxSub(a, m+1, j);
        wM = MaxLeft(a, i, m)+  
              MaxRight(a, m+1, j);
        return max(wL, wR, wM);
    }
}
```

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



# Thuật toán đệ qui

## • Phân tích thuật toán:

Ta cần tính xem lệnh gọi MaxSub(a,1,n) để thực hiện thuật toán đòi hỏi bao nhiêu phép cộng?

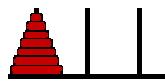
- Trước hết nhận thấy MaxLeft và MaxRight đòi hỏi

$$n/2 + n/2 = n \text{ phép cộng}$$

- Vì vậy, nếu gọi  $T(n)$  là số phép cộng cần tìm, ta có công thức đệ qui sau:

$$T(n) = \begin{cases} 0 & n=1 \\ T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + n = 2T\left(\frac{n}{2}\right) + n & n>1 \end{cases}$$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



# Thuật toán đệ qui

- Ta khẳng định rằng  $T(2^k) = k \cdot 2^k$ . Ta chứng minh bằng qui nạp

- Cơ sở qui nạp:** Nếu  $k=0$  thì  $T(2^0) = T(1) = 0 = 0 \cdot 2^0$ .

- Chuyển qui nạp:** Nếu  $k>0$ , giả sử rằng  $T(2^{k-1}) = (k-1)2^{k-1}$  là đúng. Khi đó

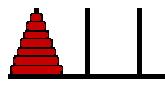
$$T(2^k) = 2T(2^{k-1}) + 2^k = 2(k-1) \cdot 2^{k-1} + 2^k = k \cdot 2^k.$$

- Quay lại với ký hiệu  $n$ , ta có

$$T(n) = n \log n .$$

- Kết quả thu được là tốt hơn thuật toán thứ hai !**

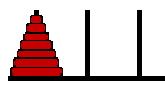
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## So sánh các thuật toán

- Cùng một bài toán ta đã đề xuất 3 thuật toán đòi hỏi số lượng phép toán khác nhau và vì thế sẽ đòi hỏi thời gian tính khác nhau.
- Các bảng trình bày dưới đây cho thấy thời gian tính với giả thiết máy tính có thể thực hiện  $10^8$  phép cộng trong 1 giây

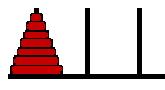
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



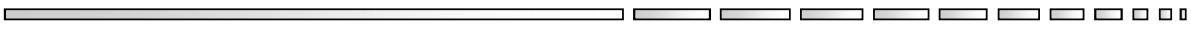
## Thời gian tính

Số phép toán	n=10	0 ms	n=100	0 ms
$\log(n)$	3.32	$3.3 \times 10^{-8}$ sec	6.64	$6 \times 10^{-8}$ sec
$n \log(n)$	33.2	$3.3 \times 10^{-7}$ sec	664	$6.6 \times 10^{-6}$ sec
$n^2$	100	$10^{-6}$ sec	$10^{10}$	$10^{-4}$ sec
$n^3$	$1 \times 10^3$	$10^{-5}$ sec	$1 \times 10^9$	$10^{-2}$ sec
$2^n$	$2.2 \times 10^4$	$2 \times 10^{-4}$ sec	$2.48 \times 10^{13}$	$> 10^{26}$ thế kỷ

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



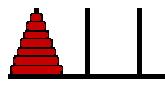
# Thời gian tính



Số phép toán	n=100000	Time	n=10 <sup>6</sup>	Time
$\log(n)$	13.3	$10^{-6}$ sec	10.9	$<10^{-5}$ sec
$\log_2(n)$	$1.39 \times 10^5$	$10^{-5}$ sec	$1.39 \times 10^7$	$2 \times 10^{-1}$ sec
$\pi$	$1 \times 10^0$	1 sec	$1 \times 10^{12}$	$2.77 \times 10^0$
$\pi$	$1 \times 10^{12}$	2.7 giờ	$1 \times 10^{15}$	118 ngày
$e$	$8.31 \times 10^{4342}$	$> 10^{4327}$ thế kỷ		

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

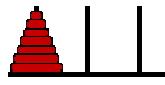
## Bảng quy đổi thời gian



- Bảng sau đây dùng để tính thời gian thực hiện

Thời gian	Số phép toán
1 giây	$10^0$
1 phút	$6 \times 10^3$
1 giờ	$3.6 \times 10^6$
1 ngày	$8.64 \times 10^8$
1 năm	$3.1536 \times 10^{10}$
1 thế kỷ	$3.1536 \times 10^{17}$

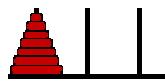
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Bài toán mở đầu

- Với  $n$  nhỏ thời gian tính là không đáng kể.
- Vấn đề trở nên nghiêm trọng hơn khi  $n > 10^6$ . Lúc đó chỉ có thuật toán thứ ba là có thể áp dụng được trong thời gian thực.
- Còn có thể làm tốt hơn nữa không?
- Có thể đề xuất thuật toán chỉ đòi hỏi  $n$  phép cộng!

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

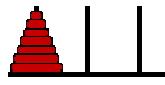


## Thuật toán Quy hoạch động

Việc phát triển thuật toán dựa trên DP bao gồm 3 giai đoạn:

1. **Phân rã:** Chia bài toán cần giải thành những bài toán con nhỏ hơn có cùng dạng với bài toán ban đầu.
2. **Ghi nhận lời giải:** Lưu trữ lời giải của các bài toán con vào một bảng.
3. **Tổng hợp lời giải:** Lần lượt từ lời giải của các bài toán con kích thước nhỏ hơn tìm cách xây dựng lời giải của bài toán kích thước lớn hơn, cho đến khi thu được lời giải của bài toán xuất phát (là bài toán con có kích thước lớn nhất).

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Thuật toán QHĐ

- **Phân rã.** Giải  $s_i$  là trọng lượng của dãy con lín nhết trong dãy  $a_1, a_2, \dots, a_i$ ,  $i = 1, 2, \dots, n$ .  
Rất rõ ràng  $s_n$  là giá trị cần tìm.
- **Tổng hợp lêii giới.**

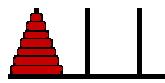
– Trước hết, ta có

$$s_1 = a_1.$$

– Giả sử  $i > 1$  và  $s_k$  là giá trị biêt với  $k = 1, 2, \dots, i-1$ . Ta cần tính  $s_i$  là trọng lượng của dãy con lín nhết của dãy

$$a_1, a_2, \dots, a_{i-1}, a_i.$$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Thuật toán QHĐ

- Do dãy con lớn nhất của dãy này hoặc là có chứa phần tử  $a_i$  hoặc là không chứa phần tử  $a_i$ , nên nó chỉ có thể là một trong hai dãy:
  - Dãy con lớn nhất của dãy  $a_1, a_2, \dots, a_{i-1}$
  - Dãy con lớn nhất của dãy  $a_1, a_2, \dots, a_{i-1}$  kết thúc tại  $a_i$ .
- Từ đó suy ra

$$s_i = \max \{s_{i-1}, e_i\}, i = 2, \dots, n.$$

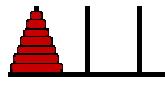
trong đó  $e_i$  là trọng lượng của dãy con lớn nhất của dãy  $a_1, a_2, \dots, a_i$  kết thúc tại  $a_i$ .

- Để tính  $e_i$ , ta cũng có thể sử dụng công thức đê qui sau:

$$- e_1 = a_1;$$

$$- e_i = \max \{a_i, e_{i-1} + a_i\}, i = 2, \dots, n.$$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



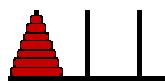
# Thuật toán QHĐ

```
MaxSub(a);  
{  
    smax = a[1];      (* smax – trọng lượng của dãy con lõi n nhỏ nhất *)  
    maxendhere = a[1]; (* maxendhere – trọng lượng của dãy con lớn nhất kết thúc tại a[i] *)  
    imax = 1;          (* imax - vị trí kết thúc của dãy con lõi n nhỏ nhất *)  
    for i = 2 to n {  
        u = maxendhere + a[i];  
        v = a[i];  
        if (u > v) maxendhere = u  
        else maxendhere = v;  
        if (maxendhere > smax)then {  
            smax := maxendhere;  
            imax := i;  
        }  
    }  
}
```

## Phân tích thuật toán:

Dễ thấy số phép toán công phải thực hiện trong thuật toán (số lần thực hiện câu lệnh **u = maxendhere + a[i];**) là  $n$ .

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



# NỘI DUNG

1.1. Ví dụ mở đầu



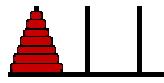
1.2. Thuật toán và độ phức tạp

1.3. Ký hiệu tiệm cận

1.4. Giả ngôn ngữ

1.5. Một số kỹ thuật phân tích thuật toán

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



# Khái niệm bài toán tính toán

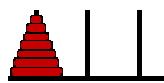
- Sinh nghĩa.** Bại to, n tñnh to, n F lµ , nh x<sup>1</sup> tõ tÆp c,c x°u nhþ ph°n ®é dµi h÷u h<sup>1</sup>n vµo tÆp c,c x°u nhþ ph°n ®é dµi h÷u h<sup>1</sup>n:

$$F : \{0, 1\}^* \rightarrow \{0, 1\}^*.$$

## Ví dô:

- Mci sè nguyªn x ®Ùu cã thÓ biÙu diÔn dưới d¹ng x°u nhþ ph°n lµ c, ch viÙt trong hÖ ®Ùm nhþ ph°n cña nã.
- HÖ phuong trænh tuyÙn tñnh Ax = b cã thÓ biÙu diÔn dưới d¹ng x°u lµ ghØp nèi cña c,c x°u biÙu diÔn nhþ ph°n cña c,c thµnh phÇn cña ma trÈn A vµ vect¬ b.
- Ša thøc mét biÙn P(x) = a<sub>0</sub> + a<sub>1</sub> x + ... + a<sub>n</sub> x<sup>n</sup> hoµn toµn x,c ®Þnh bëi d· y sè n, a<sub>0</sub>, a<sub>1</sub>, ..., a<sub>n</sub>, mµ ®Ù biÙu diÔn d· y sè nµy chong ta cã thÓ sö dông x°u nhþ ph°n.

Câu trùc dñ li u và thuật toán - N.Đ. Nghia. Bộ môn KHMT



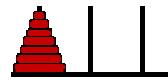
# Khái niệm thuật toán

- Sinh nghĩa.** Ta hiÙu thuÆt to, n gi¶i bµi to, n ®Æt ra lµ mét thñ t¢c x,c ®Þnh bao g m mét d· y h÷u h<sup>1</sup>n c,c bu c c n th c hiÙn ®Ó thu đu c ®Çu ra cho mét ®Çu vµo cho tr c cña bµi to, n.
- ThuÆt to, n cã c,c ®Æc truong sau ®©y:
  - ŠÇu vµo (Input):** ThuÆt to, n nh n d· liÙu vµo tõ mét tÆp nµo ® .
  - ŠÇu ra (Output):** Ví mci tÆp c,c d· liÙu ®Çu vµo, thuÆt to, n đưa ra c,c d· liÙu t ong v  i l i gi¶i cña bµi to, n.
  - Ch nh x,c (Precision):** C,c bu c cña thuÆt to, n đu c m c t¶ ch nh x,c.
  - H÷u h<sup>1</sup>n (Finiteness):** ThuÆt to, n c n ph¶i đưa đu c ®Çu ra sau mét s  h÷u h<sup>1</sup>n (cã thÓ r t l n) bu c v  i m i ®Çu vµo.
  - Š¬n tr  (Uniqueness):** C,c k t qu¶i trung gian cña t ng bu c th c hiÙn thuÆt to, n đu c x,c ®Þnh mét c,c ch ®¬n tr  v i ch  ph  thu c vµo ®Çu vµo v i c,c k t qu¶i cña c,c bu c tr c.
  - T ng qu,t (Generality):** ThuÆt to, n cã thÓ p d ng ®Ó gi¶i m i bµi to, n cã d¹ng ®. cho.

Câu trùc dñ li u và thuật toán - N.Đ. Nghia. Bộ môn KHMT

# Giải bài toán là gì?

What is Problem Solving?



- **Problem solving**

- Là quá trình đặt bài toán và phát triển chương trình máy tính để giải bài toán đặt ra

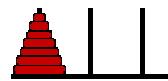
- **Lời giải bài toán bao gồm:**

- Thuật toán (Algorithms)
    - *Algorithm: là dãy các bước cần thực hiện để từ dữ liệu vào (input) đưa ra kết quả đầu ra (output) của bài toán trong thời gian hữu hạn.*
  - Cấu trúc dữ liệu:
    - *Cách tổ chức lưu trữ dữ liệu vào - ra*

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

# Vòng đời của phần mềm

The Life Cycle of Software



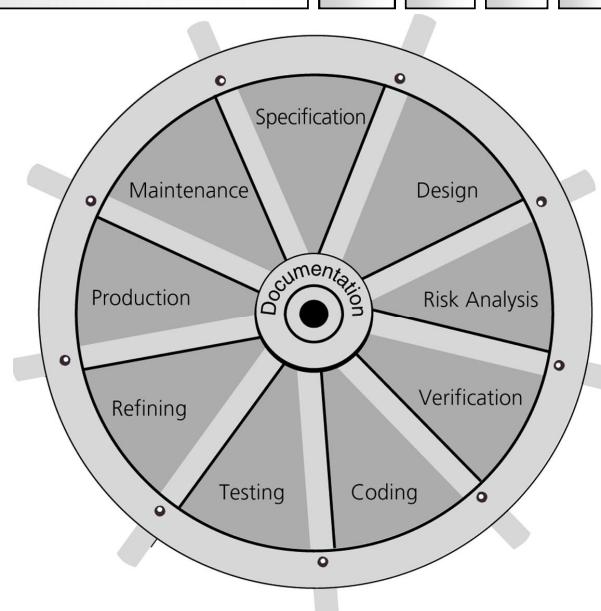
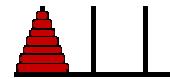
- **Vòng đời của phần mềm**

- Là một quá trình dài và liên tục
  - Đòi hỏi để phát triển một phần mềm có chất lượng tốt
  - Lập trình viên có thể di chuyển từ một pha trong vòng đời sang bất kỳ pha nào còn lại

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

# Vòng đời của phần mềm

The Life Cycle of Software



Vòng đời của phần mềm như là một bánh lái có thể quay từ một pha đến một pha khác bất kỳ.

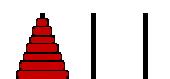
Nguyễn Đức Nghĩa - Bộ

1-35

Cấu trúc dữ liệu và thuật toán - N.D.Nghĩa - Bộ môn KHMT  
môn KHMT ĐHBKHN

# Vòng đời của phần mềm

The Life Cycle of Software



## 9 pha:

- Phase 1: Chỉ rõ đặc điểm kỹ thuật (Specification) (đặc tả)
- Phase 2: Thiết kế (Design)
- Phase 3: Phân tích rủi ro (Risk Analysis)
- Phase 4: Kiểm thử (Verification)
- Phase 5: Lập trình (Coding)
- Phase 6: Test thử (Testing)
- Phase 7: Tinh chế lời giải (Refining the Solution)
- Phase 8: Sản xuất (Production)
- Phase 9: Bảo trì (Maintenance)

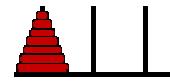
Nguyễn Đức Nghĩa - Bộ

1-36

Cấu trúc dữ liệu và thuật toán - N.D.Nghĩa - Bộ môn KHMT  
môn KHMT ĐHBKHN

# Vòng đời của phần mềm

The Life Cycle of Software



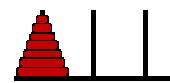
## • Phase 1: Đặc tả (Specification)

- Các khía cạnh của bài toán cần chỉ rõ:
  - *Dữ liệu đầu vào là gì (What is the input data?)*
  - *Dữ liệu nào là đúng đắn, là không đúng đắn?*
  - *Ai là người sử dụng phần mềm và giao diện người dùng cần được thiết kế như thế nào?*
  - *Cần phát hiện những lỗi gì và cần thông báo như thế nào về chúng?*
  - *Có thể có các giả thiết nào?*
  - *Có những trường hợp đặc biệt nào?*
  - *Dạng của dữ liệu đưa ra như thế nào?*
  - *Cần có các tài liệu gì?*
  - *Cái gì cần phát triển trong tương lai?*
- Chương trình mẫu (Chương trình mô phỏng dáng điệu của một phần của sản phẩm phần mềm cần phát triển)

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

# Vòng đời của phần mềm

The Life Cycle of Software



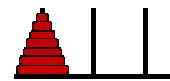
## • Phase 2: Thiết kế (Design). Quá trình thiết kế bao gồm:

- Chia chương trình ra thành các modules (Modules: là các đơn vị chương trình độc lập)
- Chỉ rõ mục đích của mỗi module
- Chỉ rõ dòng dữ liệu trong các modules
- Xác định giao diện (Interfaces - Cơ cấu giao tiếp giữa các module)

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

# Vòng đời của phần mềm

The Life Cycle of Software

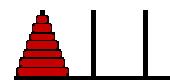


- Phase 3: Phân tích rủi ro (Risk Analysis)
- Phase 4: Kiểm thử (Verification)
  - *Chứng minh tính đúng đắn của thuật toán bằng các phương pháp hình thức, ...*
- Phase 5: Cài đặt (Coding)
  - Liên quan đến việc chuyển thiết kế sang một ngôn ngữ lập trình cụ thể
  - Loại trừ các lỗi ngữ pháp
- Phase 6: Test thử (Testing)
  - Liên quan đến việc loại bỏ các lỗi logic
  - Dữ liệu test phải bao gồm:
    - *Dữ liệu đúng đắn với kết quả biết trước*
    - *Dữ liệu không đúng đắn*
    - *Dữ liệu ngẫu nhiên*
    - *Dữ liệu thực tế*

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

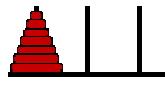
# Vòng đời của phần mềm

The Life Cycle of Software



- Phase 7: Tinh chế lời giải (Refining the Solution)
  - Do chương trình được phát triển với những giả thiết nhất định nên cần tìm cách giảm nhẹ các giả thiết được bổ sung đối với đầu vào, đầu ra
  - Bổ sung thêm các chức năng
  - Tăng các biện pháp kiểm tra lỗi
- Phase 8: Xuất xưởng (Production)
  - Bàn giao sản phẩm cho người dùng.
  - Người dùng sử dụng phần mềm
- Phase 9: Bảo trì (Maintenance)
  - Sửa chữa các lỗi do người sử dụng phát hiện.
  - Bổ sung thêm chức năng.
  - Cải tiến một số bộ phận để đáp ứng yêu cầu của người dùng tốt hơn

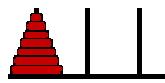
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Độ phức tạp của thuật toán

- Đánh giá **độ phức tạp tính toán** của thuật toán là đánh giá **lượng tài nguyên các loại** mà thuật toán đòi hỏi sử dụng. Có hai loại tài nguyên quan trọng đó là **thời gian** và **bộ nhớ**. Trong giáo trình này ta đặc biệt quan tâm đến đánh giá thời gian cần thiết để thực hiện thuật toán mà ta sẽ gọi là *thời gian tính* của thuật toán.
- Thời gian tính phụ thuộc vào dữ liệu vào.
- Sinh nghĩa.** Ta giải thích thước đ $\ddot{u}$  li $\ddot{u}$   $\text{v} \mu \text{o}$  (hay độ dài dữ liệu vào) l $\mu$  s $\ddot{e}$  b $\ddot{u}$ t c $\ddot{C}$ n thi $\ddot{u}$ t  $\text{b}\ddot{\text{i}}$   $\text{b}\ddot{\text{i}}$   $\text{d}\ddot{\text{i}}$ n n $\ddot{a}$ .
- Ta sẽ tìm cách đánh giá thời gian tính của thuật toán bởi một **hàm của độ dài dữ liệu vào**.

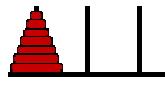
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Phép toán cơ bản

- Đo thời gian tính bằng đơn vị đo nào?
- Sinh nghĩa.** Ta giải phép toán  $c - b$  l $\mu$  phép toán c $\ddot{a}$  th $\ddot{u}$  th $\ddot{c}$  hi $\ddot{u}$ n v $\ddot{i}$  t $\ddot{h}\acute{e}$ i gian b $\ddot{u}$  ch $\ddot{a}$ n b $\ddot{e}$ i m $\ddot{e}$ t h $\ddot{u}$ ng s $\ddot{e}$  kh $\ddot{u}$ ng phô thu $\ddot{c}$  v $\mu \text{o}$  k $\ddot{y}$ ch thước đ $\ddot{u}$  li $\ddot{u}$ .
- Şó t $\ddot{y}$ nh toán t $\ddot{h}\acute{e}$ i gian t $\ddot{y}$ nh c $\ddot{a}$ na thu $\ddot{E}$ t toán ta s $\ddot{I}$   $\text{b}\ddot{\text{O}}$ m s $\ddot{e}$  phép toán  $c - b$  l $\mu$  m $\ddot{a}$  ph $\ddot{u}$ i th $\ddot{u}$ c hi $\ddot{u}$ n.

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

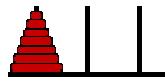


# Các loại thời gian tính

## Chóng ta sỹ quan tcm ®Ön

- Thời gian tèi thiØu cçn thiØt ®Ó thùc hiØn thuËt to,n ví i mäi bé d÷ liØu ®Çu vµo kÝch th\u00f3r\u00f3c n. Thời gian nh\u00f3 vÆy sÃ đ\u00f9i c g\u00e1i l\u00f9 th\u00f3i gian t\u00fdnh t\u00f3t nh\u00e1t c\u00e1n thuËt to,n ví i ®Çu vµo kÝch th\u00f3r\u00f3c n.
- Thời gian nh\u00f3u nh\u00e1t cçn thiØt ®Ó thùc hiØn thuËt to,n ví i mäi bé d÷ liØu ®Çu vµo kÝch th\u00f3r\u00f3c n. Thời gian nh\u00f3 vÆy sÃ đ\u00f9i c g\u00e1i l\u00f9 th\u00f3i gian t\u00fdnh t\u00f3i nh\u00e1t c\u00e1n thuËt to,n ví i ®Çu vµo kÝch th\u00f3r\u00f3c n.
- Thời gian trung b\u00e1nh cçn thiØt ®Ó thùc hiØn thuËt to,n tr\u00e1n t\u00e9p h\u00f3u h\u00e1n c,c ®Çu vµo kÝch th\u00f3r\u00f3c n. Thời gian nh\u00f3 vÆy sÃ đ\u00f9i c g\u00e1i l\u00f9 th\u00f3i gian t\u00fdnh trung b\u00e1nh c\u00e1n thuËt to,n.

C\u00e1u tr\u00fc d\u00f9r li\u00e9u v\u00e0 thu\u00e1t to\u00e1n - N.\u00c4. Ngh\u00e1ia. B\u00f3m KHMT



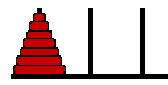
## N\u00f3I DUNG

- 1.1. Ví dụ m\u00f2i đ\u00e1u
- 1.2. Giải bài toán v\u00e0 công nghệ phần mềm
- 1.3. Thu\u00e1t to\u00e1n v\u00e0 độ phức tạp
-  1.4. Ký hiệu ti\u00e9m c\u00e1n
- 1.5. Giả ngôn ngữ
- 1.6. Một số k\u00ed thu\u00e1t ph\u00e1n t\u00fich thu\u00e1t to\u00e1n

C\u00e1u tr\u00fc d\u00f9r li\u00e9u v\u00e0 thu\u00e1t to\u00e1n - N.\u00c4. Ngh\u00e1ia. B\u00f3m KHMT

# Ký hiệu tiệm cận

Asymptotic Notation

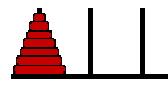


## $\Theta$ , $\Omega$ , $O$

- Được sử dụng để mô tả thời gian tính của thuật toán
- Thay vì nói chính xác thời gian tính, ta nói  $\Theta(n^2)$
- Được xác định đối với các hàm nhận giá trị nguyên không âm
- Dùng để so sánh tốc độ tăng của hai hàm

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

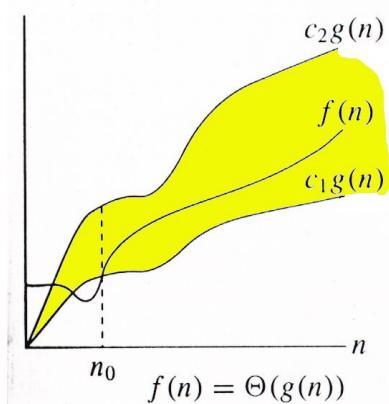
## Ký hiệu $\Theta$



Đối với hàm  $g(n)$ , ta ký hiệu  $\Theta(g(n))$  là tập các hàm

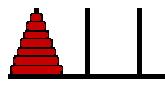
$\Theta(g(n)) = \{f(n): \text{tồn tại các hằng số } c_1, c_2 \text{ và } n_0 \text{ sao cho}$

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \text{ với mọi } n \geq n_0\}$$



Ta nói rằng  $g(n)$  là đánh giá tiệm cận đúng cho  $f(n)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ

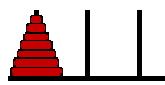
$$10n^2 - 3n = \Theta(n^2) ?$$

- Với giá trị nào của các hằng số  $n_0$ ,  $c_1$ , và  $c_2$  thì bất đẳng thức trong định nghĩa là đúng?
- Lấy  $c_1$  bé hơn hệ số của số hạng với số mũ cao nhất, còn  $c_2$  lấy lớn hơn, ta có

$$n^2 \leq 10n^2 - 3n \leq 11n^2, \text{ với mọi } n \geq 1.$$

- Đối với hàm đa thức: Để so sánh tốc độ tăng cần nhìn vào số hạng với số mũ cao nhất

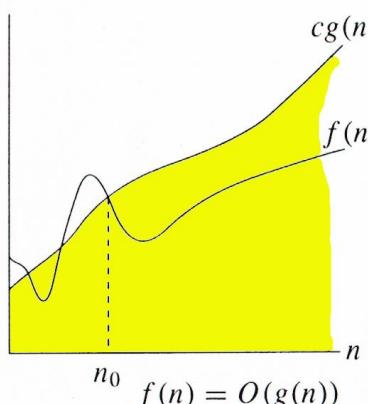
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ký hiệu $O$ (đọc là ô lớn - big O)

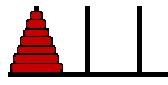
Đối với hàm  $g(n)$  cho trước, ta ký hiệu  $O(g(n))$  là tập các hàm  $O(g(n)) = \{f(n): \text{tồn tại các hằng số dương } c \text{ và } n_0 \text{ sao cho:}$

$$f(n) \leq cg(n) \text{ với mọi } n \geq n_0\}$$



Ta nói  $g(n)$  là **cận trên tiệm cận** của  $f(n)$

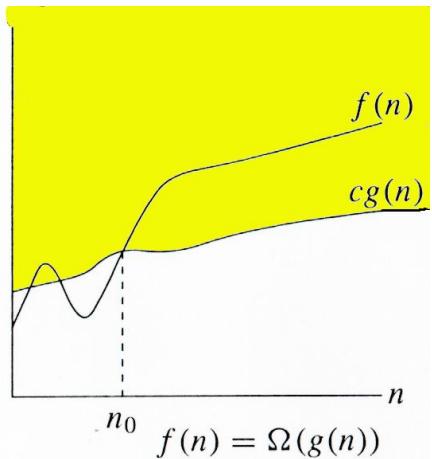
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ký hiệu $\Omega$

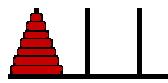
Đối với hàm  $g(n)$  cho trước, ta ký hiệu  $\Omega(g(n))$  là tập các hàm  $\Omega(g(n)) = \{f(n): \text{tồn tại các hằng số dương } c \text{ và } n_0 \text{ sao cho:}$

$$cg(n) \leq f(n) \text{ với mọi } n \geq n_0\}$$



Ta nói  $g(n)$  là **cận dưới tiệm cận** cho  $f(n)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Liên hệ giữa $\Theta$ , $\Omega$ , $O$

Đối với hai hàm bất kỳ  $g(n)$  và  $f(n)$ ,

$$f(n) = \Theta(g(n))$$

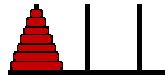
khi và chỉ khi

$$f(n) = O(g(n)) \text{ và } f(n) = \Omega(g(n))$$

tức là

$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$

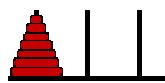
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



# Cách nói về thời gian tính

- Nói “Thời gian tính là  $O(f(n))$ ” hiểu là: Đánh giá trong tình huống tồi nhất (worst case) là  $O(f(n))$ . Thường nói: “Đánh giá thời gian tính trong tình huống tồi nhất là  $O(f(n))$ ”
  - Nghĩa là thời gian tính trong tình huống tồi nhất được xác định bởi một hàm nào đó  $g(n) \in O(f(n))$
- “Thời gian tính là  $\Omega(f(n))$ ” hiểu là: Đánh giá trong tình huống tốt nhất (best case) là  $\Omega(f(n))$ . Thường nói: “Đánh giá thời gian tính trong tình huống tốt nhất là  $\Omega(f(n))$ ”
  - Nghĩa là thời gian tính trong tình huống tốt nhất được xác định bởi một hàm nào đó  $g(n) \in \Omega(f(n))$

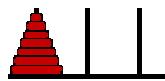
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ

- *Sắp xếp chèn (Insertion sort)* đòi hỏi thời gian  $\Theta(n^2)$  trong tình huống tồi nhất, vì thế bài toán sắp xếp có thời gian là  $O(n^2)$ .
- Mọi thuật toán sắp xếp đều đòi hỏi duyệt qua tất cả các phần tử, vì thế bài toán sắp xếp có thời gian  $\Omega(n)$  trong tình huống tốt nhất.
- Trên thực tế, sử dụng (chẳng hạn) *sắp xếp trộn (merge sort)*, bài toán sắp xếp có thời gian  $\Theta(n \log n)$  trong tình huống tồi nhất.

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ký hiệu tiệm cận trong các đẳng thức

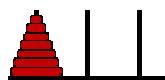
- Được sử dụng để thay thế các biểu thức chứa các toán hạng với tốc độ tăng chậm

- Ví dụ,

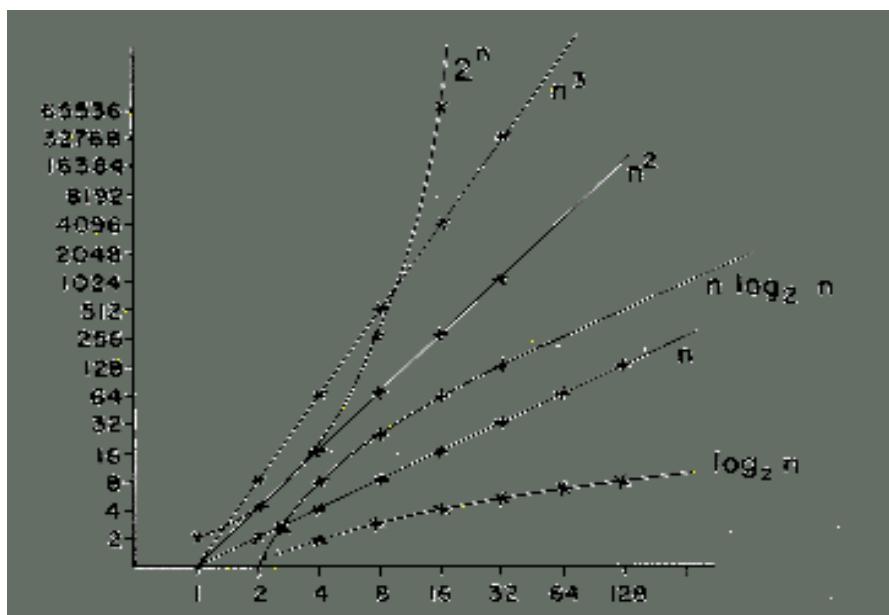
$$\begin{aligned}4n^3 + 3n^2 + 2n + 1 &= 4n^3 + 3n^2 + \Theta(n) \\&= 4n^3 + \Theta(n^2) = \Theta(n^3)\end{aligned}$$

- Trong các đẳng thức,  $\Theta(f(n))$  thay thế cho một *hàm nào đó*  $g(n) \in \Theta(f(n))$ 
  - Trong ví dụ trên,  $\Theta(n^2)$  thay thế cho  $3n^2 + 2n + 1$

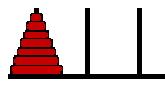
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Đồ thị của một số hàm cơ bản



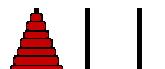
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Giá trị của các hàm cơ bản

$n$	$\log n$	$n$	$n \log n$	$n^2$	$n^3$	$2^n$
4	2	4	8	16	64	16
8	3	8	24	64	512	256
16	4	16	64	256	4,096	65,536
32	5	32	160	1,024	32,768	4,294,967,296
64	6	64	384	4,094	262,144	$1.84 * 10^{19}$
128	7	128	896	16,384	2,097,152	$3.40 * 10^{38}$
256	8	256	2,048	65,536	16,777,216	$1.15 * 10^{77}$
512	9	512	4,608	262,144	134,217,728	$1.34 * 10^{154}$
1024	10	1,024	10,240	1,048,576	1,073,741,824	$1.79 * 10^{308}$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Sự tương tự giữa so sánh các hàm số và so sánh các số

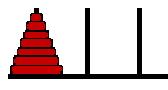
$$f \leftrightarrow g \approx a \leftrightarrow b$$

$$f(n) = O(g(n)) \approx a \leq b$$

$$f(n) = \Omega(g(n)) \approx a \geq b$$

$$f(n) = \Theta(g(n)) \approx a = b$$

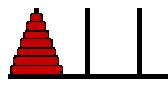
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Liên hệ với khái niệm giới hạn

- $\lim_{n \rightarrow \infty} [f(n) / g(n)] = 0 \Rightarrow f(n) \in o(g(n))$
- $\lim_{n \rightarrow \infty} [f(n) / g(n)] < \infty \Rightarrow f(n) \in O(g(n))$
- $0 < \lim_{n \rightarrow \infty} [f(n) / g(n)] < \infty \Rightarrow f(n) \in \Theta(g(n))$
- $0 < \lim_{n \rightarrow \infty} [f(n) / g(n)] \Rightarrow f(n) \in \Omega(g(n))$
- $\lim_{n \rightarrow \infty} [f(n) / g(n)] = \infty \Rightarrow f(n) \in \omega(g(n))$
- $\lim_{n \rightarrow \infty} [f(n) / g(n)]$  không xác định  $\Rightarrow$  không thể nói gì

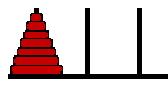
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Các tính chất

- **Truyền ứng (Transitivity)**  
 $f(n) = \Theta(g(n)) \& g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$   
 $f(n) = O(g(n)) \& g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$   
 $f(n) = \Omega(g(n)) \& g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$
- **Đối xứng (Symmetry)**  
 $f(n) = \Theta(g(n))$  khi và chỉ khi  $g(n) = \Theta(f(n))$
- **Đối xứng chuyển vị (Transpose Symmetry)**  
 $f(n) = O(g(n))$  khi và chỉ khi  $g(n) = \Omega(f(n))$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ

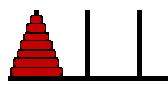
A

B

- $5n^2 + 100n$        $3n^2 + 2$

- $\log_3(n^2)$        $\log_2(n^3)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ

A

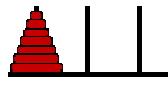
B

- $5n^2 + 100n$        $3n^2 + 2$        $A \in \Theta(B)$

$$A \in \Theta(n^2), n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$$

- $\log_3(n^2)$        $\log_2(n^3)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



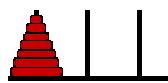
## Ví dụ

A

B

- $5n^2 + 100n$        $3n^2 + 2$        $A \in \Theta(B)$   
 $A \in \Theta(n^2), n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$
- $\log_3(n^2)$        $\log_2(n^3)$        $A \in \Theta(B)$   
 $\log_b a = \log_c a / \log_c b; A = 2\lg n / \lg 3, B = 3\lg n, A/B = 2/(3\lg 3)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



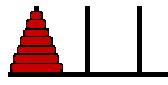
## Ví dụ

A

B

- $5n^2 + 100n$        $3n^2 + 2$        $A \in \Theta(B)$   
 $A \in \Theta(n^2), n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$
- $\log_3(n^2)$        $\log_2(n^3)$        $A \in \Theta(B)$   
 $\log_b a = \log_c a / \log_c b; A = 2\lg n / \lg 3, B = 3\lg n, A/B = 2/(3\lg 3)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

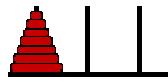


## Ví dụ

- 
- Với mỗi cặp hàm sau đây, hoặc  $f(n) = O(g(n))$ ,  $f(n) = \Omega(g(n))$ , hoặc  $f(n) = \Theta(g(n))$ . Hãy xác định quan hệ nào là đúng.

$f(n) = \log n^2; g(n) = \log n + 5$	$f(n) = \Theta(g(n))$
$f(n) = n; g(n) = \log n^2$	$f(n) = \Omega(g(n))$
$f(n) = \log \log n; g(n) = \log n$	$f(n) = O(g(n))$
$f(n) = n; g(n) = \log^2 n$	$f(n) = \Omega(g(n))$
$f(n) = n \log n + n; g(n) = \log n$	$f(n) = \Omega(g(n))$
$f(n) = 10; g(n) = \log 10$	$f(n) = \Theta(g(n))$
$f(n) = 2^n; g(n) = 10n^2$	$f(n) = \Omega(g(n))$
$f(n) = 2^n; g(n) = 3^n$	$f(n) = O(g(n))$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ

---

- $2n^2 = O(n^3)$ :  $2n^2 \leq cn^3 \Rightarrow 2 \leq cn \Rightarrow c = 1$  and  $n_0 = 2$

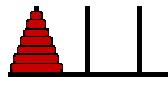
- $n^2 = O(n^2)$ :  $n^2 \leq cn^2 \Rightarrow c \geq 1 \Rightarrow c = 1$  and  $n_0 = 1$

- $1000n^2 + 1000n = O(n^2)$ :

$$1000n^2 + 1000n \leq cn^2 \Rightarrow c = 1001 \text{ and } n_0 = 1000$$

- $n = O(n^2)$ :  $n \leq cn^2 \Rightarrow cn \geq 1 \Rightarrow c = 1$  and  $n_0 = 1$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ

–  $5n^2 = \Omega(n)$

$\exists c, n_0$  sao cho:  $0 \leq cn \leq 5n^2 \Rightarrow cn \leq 5n^2 \Rightarrow c = 1$  và  $n_0 = 1$

–  $100n + 5 \neq \Omega(n^2)$

Giả sử:  $\exists c, n_0$  sao cho:  $0 \leq cn^2 \leq 100n + 5$ .

Ta có:  $100n + 5 \leq 100n + 5n (\forall n \geq 1) = 105n$

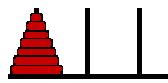
Suy ra:  $cn^2 \leq 105n \Rightarrow n(cn - 105) \leq 0$

Do  $n$  dương  $\Rightarrow cn - 105 \leq 0 \Rightarrow n \leq 105/c$

$\Rightarrow$  vô lý:  $n$  không thể nhỏ hơn hằng số

–  $n = \Omega(2n), n^3 = \Omega(n^2), n = \Omega(\log n)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ

**Chứng minh các khẳng định:**

❖  $n^2/2 - n/2 = \Theta(n^2)$

•  $\frac{1}{2}n^2 - \frac{1}{2}n \leq \frac{1}{2}n^2 \quad \forall n \geq 0 \quad \Rightarrow \quad c_2 = \frac{1}{2}$

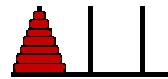
•  $\frac{1}{2}n^2 - \frac{1}{2}n \geq \frac{1}{2}n^2 - \frac{1}{2}n * \frac{1}{2}n (\forall n \geq 2) = \frac{1}{4}n^2$

$\Rightarrow c_1 = \frac{1}{4}$

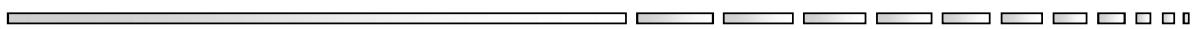
❖  $n \neq \Theta(n^2)$ :  $c_1 n^2 \leq n \leq c_2 n^2$

$\Rightarrow$  chỉ đúng với:  $n \leq 1/c_1$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ



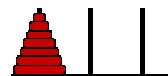
❖  $6n^3 \neq \Theta(n^2)$ :  $c_1 n^2 \leq 6n^3 \leq c_2 n^2$

$\Rightarrow$  chỉ đúng với:  $n \leq c_2 / 6$

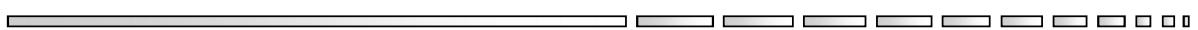
❖  $n \neq \Theta(\log n)$ :  $c_1 \log n \leq n \leq c_2 \log n$

$\Rightarrow c_2 \geq n/\log n$ ,  $\forall n \geq n_0$  – không thể xảy ra

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Chú ý

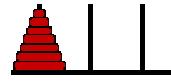


- Giá trị của  $n_0$  và  $c$  **không phải là duy nhất** trong chứng minh công thức tiệm cận
- Chứng minh rằng  $100n + 5 = O(n^2)$ 
  - $100n + 5 \leq 100n + n = 101n \leq 101n^2$  với mọi  $n \geq 5$   
 $n_0 = 5$  và  $c = 101$  là các hằng số cần tìm
  - $100n + 5 \leq 100n + 5n = 105n \leq 105n^2$  với mọi  $n \geq 1$   
 $n_0 = 1$  and  $c = 105$  cũng là các hằng số cần tìm
- Chỉ cần tìm các hằng  $c$  và  $n_0$  **nào đó** thoả mãn bất đẳng thức trong định nghĩa công thức tiệm cận

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

# Cận trên và cận dưới

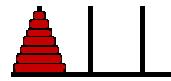
Upper Bound and Lower Bound



- **Định nghĩa (Upper Bound).** Cho bài toán  $P$ , ta nói cận trên cho thời gian tính của  $P$  là  $O(g(n))$  nếu để giải  $P$  tồn tại thuật toán giải với thời gian tính là  $O(g(n))$ .
- **Định nghĩa (Lower Bound).** Cho bài toán  $P$ , ta nói cận dưới cho thời gian tính của  $P$  là  $\Omega(g(n))$  nếu mọi thuật toán giải  $P$  đều có thời gian tính là  $\Omega(g(n))$ .
- **Định nghĩa.** Cho bài toán  $P$ , ta nói thời gian tính của  $P$  là  $\Theta(g(n))$  nếu  $P$  có cận trên là  $O(g(n))$  và cận dưới là  $\Omega(g(n))$ .

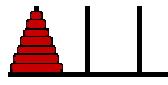
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

# Một số lớp thuật toán



- **Một số lớp thuật toán đặc biệt:**
  - $O(1)$ : hằng số (constant)
  - $O(\log n)$ : logarithmic
  - $O(n)$ : tuyến tính (linear)
  - $O(n \log n)$ : trên tuyến tính (superlinear)
  - $O(n^2)$ : bình phương (quadratic)
  - $O(n^3)$ : bậc ba (cubic)
  - $O(a^n)$ : hàm mũ (exponential) ( $a > 1$ )
  - $O(n^k)$ : đa thức (polynomial) ( $k \geq 1$ )

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

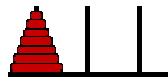


## NỘI DUNG

---

- 1.1. Ví dụ mở đầu
- 1.2. Thuật toán và độ phức tạp
- 1.3. Ký hiệu tiệm cận
-  **1.4. Giả ngôn ngữ**
- 1.5. Một số kĩ thuật phân tích thuật toán

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

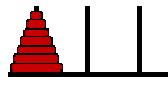


## Mô tả thuật toán: giả ngôn ngữ

---

- Để mô tả thuật toán có thể sử dụng một ngôn ngữ lập trình nào đó. Tuy nhiên điều đó có thể làm cho việc mô tả thuật toán trở nên phức tạp đồng thời rất khó nắm bắt.
- Vì thế, để mô tả thuật toán người ta thường sử dụng **giả ngôn ngữ (pseudo language)**, trong đó cho phép vừa mô tả thuật toán bằng ngôn ngữ đời thường vừa sử dụng những cấu trúc lệnh tương tự như của ngôn ngữ lập trình.
- Dưới đây ta liệt kê một số câu lệnh chính được sử dụng trong giáo trình để mô tả thuật toán.

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Mô tả thuật toán: phỏng ngôn ngữ

### • Khai báo biến

**integer** x,y;

**real** u, v;

**boolean** a, b;

**char** c, d;

**datatype** x;

### • Câu lệnh gán

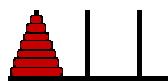
x = expression;

hoặc

x ← expression;

**Ví dụ:** x ← 1+4; y=a\*y+2;

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Mô tả thuật toán: giả ngôn ngữ

### • Cấu trúc điều khiển

**if** condition **then**

    dãy câu lệnh

**else**

    dãy câu lệnh

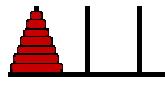
**endif;**

**while** condition **do**

    dãy câu lệnh

**endwhile;**

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Mô tả thuật toán: phỏng ngôn ngữ

**repeat**

dãy câu lệnh;

**until** condition;

**for** i=n1 **to** n2 [step d]

dãy câu lệnh;

**endfor;**

- **Vào-Ra**

**read(X);** /\* X là biến đơn hoặc mảng \*/

**print(data)** hoặc **print**(thông báo)

**Câu lệnh case:**

**Case**

*cond1: stat1;*

*cond2: stat2;*

.

.

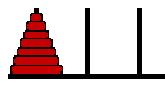
.

*condn: stat n;*

**endcase;**

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

## Mô tả thuật toán: giả ngôn ngữ



- **Hàm và thủ tục (Function and procedure)**

**Function** name(các tham số)

**begin**

mô tả biến;

các câu lệnh trong thân của hàm;

**return** (giá trị)

**end;**

**Procedure** name(các tham số)

**begin**

mô tả biến;

các câu lệnh trong thân của hàm;

**end;**

**Truyền tham số:**

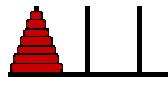
- **Tham trị**

- **Tham biến**

- **Biến cục bộ**

- **Biến toàn cục**

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Mô tả thuật toán: phỏng ngôn ngữ

- **Ví dụ:** Thuật toán tìm phần tử lớn nhất trong mảng A(1:n)

**Function** max(A(1:n))

**begin**

**datatype** x; /\* để giữ giá trị lớn nhất tìm được \*/

**integer** i;

    x=A[1];

**for** i=2 **to** n **do**

**if** x < A[i] **then**

            x=A[i];

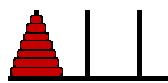
**endif**

**endfor** ;

**return** (x);

**end** max;

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Mô tả thuật toán: giả ngôn ngữ

- **Ví dụ:** Thuật toán hoán đổi nội dung hai biến

**Procedure** swap(x, y)

**begin**

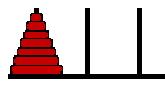
    temp=x;

    x = y;

    y = temp;

**end** swap;

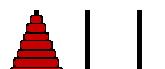
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Mô tả thuật toán: giả ngôn ngữ

- **Ví dụ.** Tìm số nguyên tố lớn hơn số nguyên dương  $n$ .
- Trước hết ta xây dựng hàm kiểm tra xem một số nguyên dương  $m$  có phải là số nguyên tố hay không (hàm Is\_prime).
- Sử dụng hàm này ta xây dựng thuật toán giải bài toán đặt ra.
- Nếu  $m=a*b$  với  $1 < a, b < m$ , thì một trong hai thừa số  $a, b$  sẽ không vượt quá  $\sqrt{m}$ . Do đó ước số nguyên tố của số nguyên dương  $m$  bao giờ cũng không vượt quá  $\sqrt{m}$ . Từ đó suy ra  $m$  sẽ là số nguyên tố nếu như nó không có ước số nào trong các số nguyên dương từ 2 đến  $\lfloor \sqrt{m} \rfloor$ .

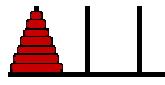
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Mô tả thuật toán: giả ngôn ngữ

- **Thuật toán kiểm tra một số nguyên dương có phải là nguyên tố hay không.**
  - **Đầu vào:** Số nguyên dương  $m$ .
  - **Đầu ra:** true nếu  $m$  là số nguyên tố, false nếu ngược lại.
- ```
function Is_prime(m);
begin
    i = 2;
    while (i*i <= m) and (m mod i ≠ 0) do i = i + 1;
    Is_prime = i > sqrt(m);
end Is_Prime;
```

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



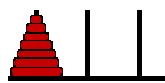
## Bình luận

- Thuật toán này chỉ có thể sử dụng đối với những số có dưới 25 chữ số.
- Nếu áp dụng đối với những số có nhiều chữ số hơn, chẳng hạn với

$n = 7483845764874895450060464578798470496948760902442664203481$

thuật toán không thể dùng được. Số  $n$  có 62 chữ số này quả thực là số nguyên tố, và vòng lặp trên sẽ phải thực hiện quãng  $10^{31}$  lần lặp. Nếu giả thiết là một phép chia có thể thực hiện trong thời gian 1 nanosecond thì thuật toán sẽ chạy trong  $10^{13}$  năm!

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Mô tả thuật toán: giả ngôn ngữ

- *Thuật toán tìm số nguyên tố lớn hơn số nguyên dương  $n$ .*

- Thuật toán sẽ sử dụng Is\_prime như chương trình con.

- **Đầu vào:** Số nguyên dương  $n$ .

- **Đầu ra:**  $m$  - số nguyên tố lớn hơn  $n$ .

**procedure** Lagre\_Prime( $n$ );

**begin**

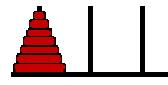
$m = n+1;$

**while** not Is\_prime( $m$ ) **do**  $m=m+1;$

**end;**

- Do tập các số nguyên tố là vô hạn, nên thuật toán Lagre\_Prime là hữu hạn.

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## NỘI DUNG

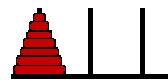
---

- 1.1. Ví dụ mở đầu
- 1.2. Thuật toán và độ phức tạp
- 1.3. Ký hiệu tiệm cận
- 1.4. Giả ngôn ngữ



## 1.5. Một số kỹ thuật phân tích thuật toán

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Các kỹ thuật cơ bản phân tích độ phức tạp của thuật toán

---

- **Cấu trúc tuần tự.** Giả sử P và Q là hai lệnh trong chương trình, cả hai lệnh đều lặp lại nhau nhưng cùng có thời gian thực ngắn hơn con. Giả Time(P), Time(Q) là thời gian tính cả P và Q tương ứng. Khi đó ta có

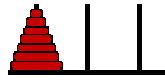
**Quy tắc tuần tự:** Thời gian tính tổng hai lệnh “P; Q”, nghĩa là P thực hiện trước, tiếp theo Q, sẽ là

$$\text{Time}(P; Q) = \text{Time}(P) + \text{Time}(Q),$$

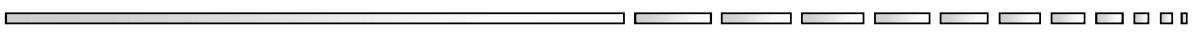
hoặc trong ký hiệu Theta:

$$\text{Time}(P; Q) = \Theta(\max(\text{Time}(P), \text{Time}(Q))).$$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Vòng lặp for

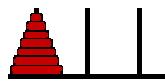


**for i =1 to m do P(i);**

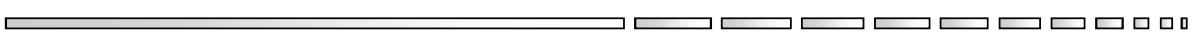
- Giả sử thời gian thực hiện  $P(i)$  là  $t(i)$
- Khi đó thời gian thực hiện vòng lặp for sẽ là

$$\sum_{i=1}^m t(i)$$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ: Tính số Fibonacci



```
function Fibiter(n)
begin
    i=0; j=1;
    for k=2 to n do
        begin
            j=j+i;
            i=i-j;
        end;
        Fibiter=j;
    end;
```

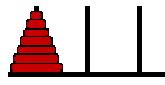
- Nếu coi các phép toán số học đòi hỏi thời gian là hằng số  $c$ , và không tính đến chi phí tổ chức vòng lặp for thì thời gian tính của hàm trên là  $\Theta(n)$ .
- Do (công thức Muavre)

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

suy ra số bit biểu diễn  $f_k$  là  $\Theta(k)$ . Do đó thời gian tính của một lần lặp là  $\Theta(k)$ . Vậy thời gian tính của Fibiter là:

$$\sum_{k=1}^n c.k = c \sum_{k=1}^n k = c \frac{n(n+1)}{2} = \Theta(n^2)$$

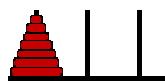
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Phân tích vòng lặp While và Repeat

- Cần xác định một hàm của các biến trong vòng lặp sao cho hàm này có giá trị giảm dần trong quá trình lặp. Khi đó
  - Để chứng minh tính kết thúc của vòng lặp ta chỉ cần chỉ ra giá trị của hàm là số nguyên dương.
  - Còn để xác định số lần lặp ta cần phải khảo sát xem giá trị của hàm giảm như thế nào.
- Việc phân tích vòng lặp Repeat được tiến hành tương tự như phân tích vòng lặp While.

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



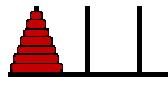
## Ví dụ: Tìm kiếm nhị phân (Binary Search)

**Input:** Mảng T[1..n] và số x  
**Output:** Giá trị i:  $1 \leq i \leq n$  sao cho  $T[i] = x$ .

**Giả thiết là x có mặt trong mảng T**

```
function Binary-Search(T[1..n], x);
begin
    i:=1; j:=n;
    while i < j do
        k:= (i+j) div 2;
        case
            x < T[k]: j:=k-1;
            x = T[k]: i:=k; j:=k; Binary-Search=k; exit; {Found!}
            x > T[k]: i:=k+1;
        end case
    endwhile
end;
```

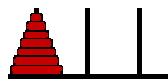
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Phân tích Binary-Search

- Gọi  $d = j - i + 1$ .  $d$  – số phần tử của mảng cần tiếp tục khảo sát
- Ký hiệu  $i, j, i^*, j^*$  theo thứ tự là giá trị của  $i, j$  trước và sau khi thực hiện lần lặp. Khi đó
  - Nếu  $x < T[k]$ , thì  $i^* = i, j^* = (i+j) \text{ div } 2 - 1$ , vì thế  $d^* = j^* - i^* + 1 = (i+j) \text{ div } 2 - 1 - i + 1 \leq (i+j)/2 - i < (j - i + 1)/2 = d/2$ .
  - Nếu  $x > T[k]$ , thì  $j^* = j, i^* = (i+j) \text{ div } 2 + 1$ , vì thế  $d^* = j^* - i^* + 1 = j - (i+j) \text{ div } 2 \leq j - (i+j-1)/2 - i = (j - i + 1)/2 = d/2$ .
  - Nếu  $x = T[k]$ , thì  $d^* = 1$  còn  $d \geq 2$

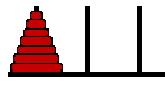
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Binary-Search (tiếp)

- Vậy luôn có:  $d^* \leq d/2$
- Do vòng lặp kết thúc khi  $d \leq 1$ , nên từ đó suy ra thuật toán phải kết thúc.
- Gọi  $d_p$  là giá trị của  $j - i + 1$  ở lần lặp thứ  $p \geq 1$  và  $d_0 = n$ . Khi đó
$$d_p \leq d_{p-1}/2, p \geq 1.$$
- Thuật toán kết thúc tại bước  $p$  nếu  $d_p \leq 1$ , điều đó xảy ra khi  $p = \lceil \log n \rceil$ .
- Vậy thời gian tính của thuật toán là  $O(\log n)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Các mô tả khác của thuật toán Binary Search

```
Function mid=bsearch1(L,p,q,key)

while q>p
    mid=floor((p+q)/2);
    if key<=L(mid)
        q=mid;
    else
        p=mid+1;
    end
end

if key==L(p)
    mid=p;
else
    mid=0;
end
```

Tìm thấy rồi thì dừng?!

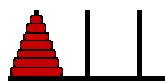
```
Function mid=bsearch2(L,p,q,key)

while q>p
    mid=floor((p+q)/2);
    if key==L(mid)
        p=mid; break
    elseif key<L(mid)
        q=mid-1;
    else
        p=mid+1;
    end
end

% Chú ý: p có thể có giá trị sai ở đây
if key==L(p)
    mid=p;
else
    mid=0;
end
```

Hãy thử với:  
L = 1, 2, 3  
key = 1, 2, 3

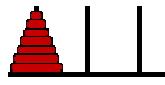
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Hàm trên C

```
boolean binary_search_1(int* a, int n, int x) {
/* Test xem x có mặt trong mảng a[] kích thước n. */
    int i;
    while (n > 0) {
        i = n / 2;
        if (a[i] == x)
            return true;
        if (a[i] < x) { /* Tiếp tục tìm ở nửa trái */
            a = &a[i + 1];
            n = n - i - 1; }
        else /* Tiếp tục tìm ở nửa phải */
            n = i;
    }
    return false;
}
```

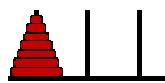
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Câu lệnh đặc trưng

- **Định nghĩa.** Câu lệnh đặc trưng là câu lệnh được thực hiện thường xuyên ít nhất là cũng như bất kỳ câu lệnh nào trong thuật toán.
- Nếu giả thiết thời gian thực hiện mỗi câu lệnh là bị chặn bởi hằng số thì thời gian tính của thuật toán sẽ cùng cõi với số lần thực hiện câu lệnh đặc trưng
- => Để đánh giá thời gian tính có thể đếm số lần thực hiện câu lệnh đặc trưng

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



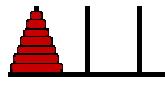
## Ví dụ: FibIter

```
function Fibiter(n)
begin
    i:=0; j:=1;
    for k:=1 to n do
        begin
            j:= j+i;
            i:= j-i;
        end;
        Fibiter:= j;
    end;
```

Câu lệnh đặc  
trưng

- Số lần thực hiện câu lệnh đặc trưng là n.
- Vậy thời gian tính của thuật toán là  $O(n)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



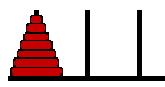
## Ví dụ: Thuật toán 1 ví dụ mở đầu

```
int maxSum = 0;
for (int i=0; i<n; i++) {
    for (int j=i; j<n; j++) {
        int sum = 0;
        for (int k=i; k<=j; k++)
            sum += a[k];
        if sum > maxSum
            maxSum = sum;
    }
}
```

Chọn câu lệnh đặc trưng là  $\text{sum} += \text{a}[k]$ .

=> Đánh giá thời gian tính của thuật toán là  $O(n^3)$

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



## Ví dụ: Sắp xếp

- **Nhận xét:** Khi thuật toán đòi hỏi thực hiện nhiều vòng lặp lồng nhau, thì có thể lấy câu lệnh nằm trong vòng lặp trong cùng làm câu lệnh đặc trưng.

- Tuy vậy, cũng cần hết sức thận trọng khi sử dụng cách lựa chọn này.  
**Ví dụ. Sắp xếp kiểu nhốt chim vào chuồng (Pigeonhole Sorting).**

Sắp xếp  $n$  số nguyên dương có giá trị nằm trong khoảng  $1..s$ .

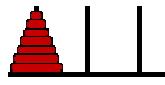
**Đầu vào:**  $T[1..n]$ : mảng chứa dãy cần sắp xếp.

**Đầu ra:**  $T[1..n]$ : mảng chứa dãy được sắp xếp không giảm.

**Thuật toán bao gồm 2 thao tác:**

- **Đếm chim:** Xây dựng mảng  $U[1..s]$ , trong đó phần tử  $U[k]$  đếm số lượng số có giá trị là  $k$  trong mảng  $T$ .
- **Nhốt chim:** Lần lượt nhốt  $U[1]$  con chim loại 1 vào  $U[1]$  lồng đầu tiên,  $U[2]$  con chim loại 2 vào  $U[2]$  lồng tiếp theo, ...

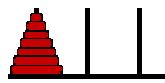
Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



# Sắp xếp kiểu nhốt chim

```
procedure Pigeonhole-Sorting;  
begin  
    (* đếm chim *)  
    for i:=1 to n do inc(U[T[i]]);  
    (* Nhốt chim *)  
    i:=0;  
    for k:=1 to s do  
        while U[k]<>0 do  
            begin  
                i:=i+1;  
                T[i]:=k;  
                U[k]:=U[k]-1;  
            end;  
    end;
```

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



# Sắp xếp kiểu nhốt chim

- Nếu chọn câu lệnh bất kỳ trong vòng lặp while làm câu lệnh đặc trưng, thì rõ ràng với mỗi  $k$ , câu lệnh này phải thực hiện  $U[k]$  lần. Do đó số lần thực hiện câu lệnh đặc trưng là

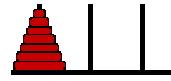
$$\sum_{k=1}^s U[k] = n$$

(do có tất cả  $n$  số cần sắp xếp). Từ đó ta có thời gian tính là  $\Theta(n)$ .

- Ví dụ sau đây cho thấy đánh giá đó chưa chính xác.  
Giả sử  $T[i] = i^2$ ,  $i = 1, \dots, n$ . Ta có

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT

# Sắp xếp kiểu nhốt chim

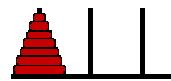


$$U[k] = \begin{cases} 1, & \text{nếu } k \text{ là số chính phương} \\ 0, & \text{nếu } k \neq 1 \end{cases}$$

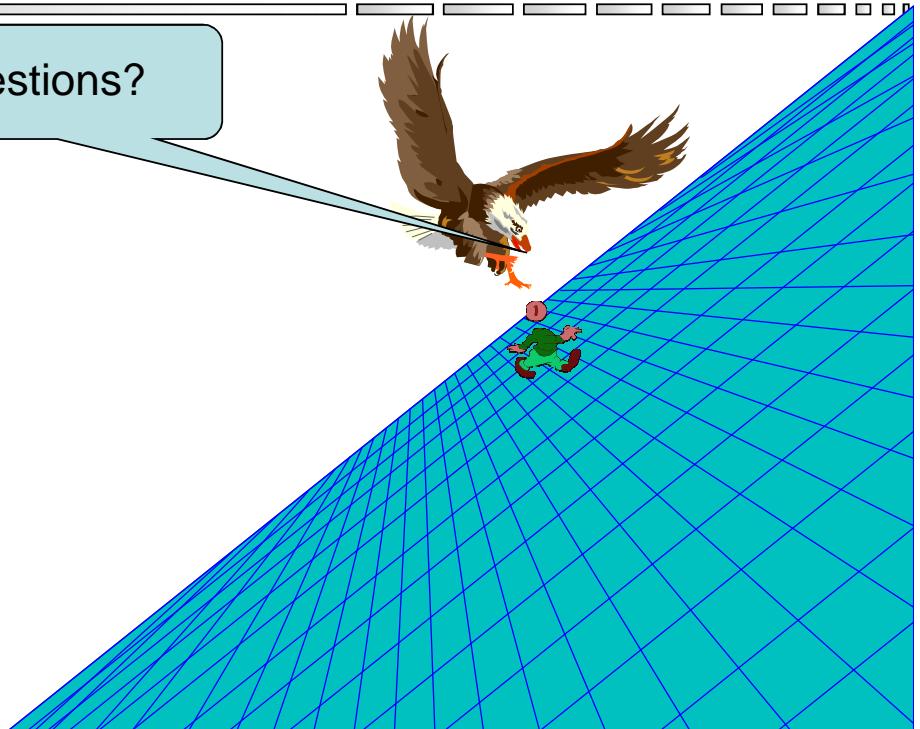
Khi đó  $s = n^2$ . Rõ ràng thời gian tính của thuật toán không phải là  $O(n)$ , mà phải là  $O(n^2)$ .

- Lỗi ở đây là do ta xác định câu lệnh đặc trưng chưa đúng, các câu lệnh trong thân vòng lặp *while* không thể dùng làm câu lệnh đặc trưng trong trường hợp này, do có rất nhiều vòng lặp rỗng (khi  $U[k] = 0$ ).
- Nếu ta chọn câu lệnh kiểm tra  $U[k] \neq 0$  làm câu lệnh đặc trưng thì rõ ràng số lần thực hiện nó sẽ là  $n + s$ . Vậy thuật toán có thời gian tính  $O(n+s) = O(n^2)$ .

Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT



Questions?



Cấu trúc dữ liệu và thuật toán - N.Đ. Nghĩa. Bộ môn KHMT