
Chương 1

Phân tích và thiết kế giải thuật

Mục tiêu

- Mối quan hệ giữa giải thuật và cấu trúc dữ liệu
- Thiết kế giải thuật
- Phân tích giải thuật
- Đánh giá độ phức tạp giải thuật

Nội dung

1.1 Giải thuật và cấu trúc dữ liệu

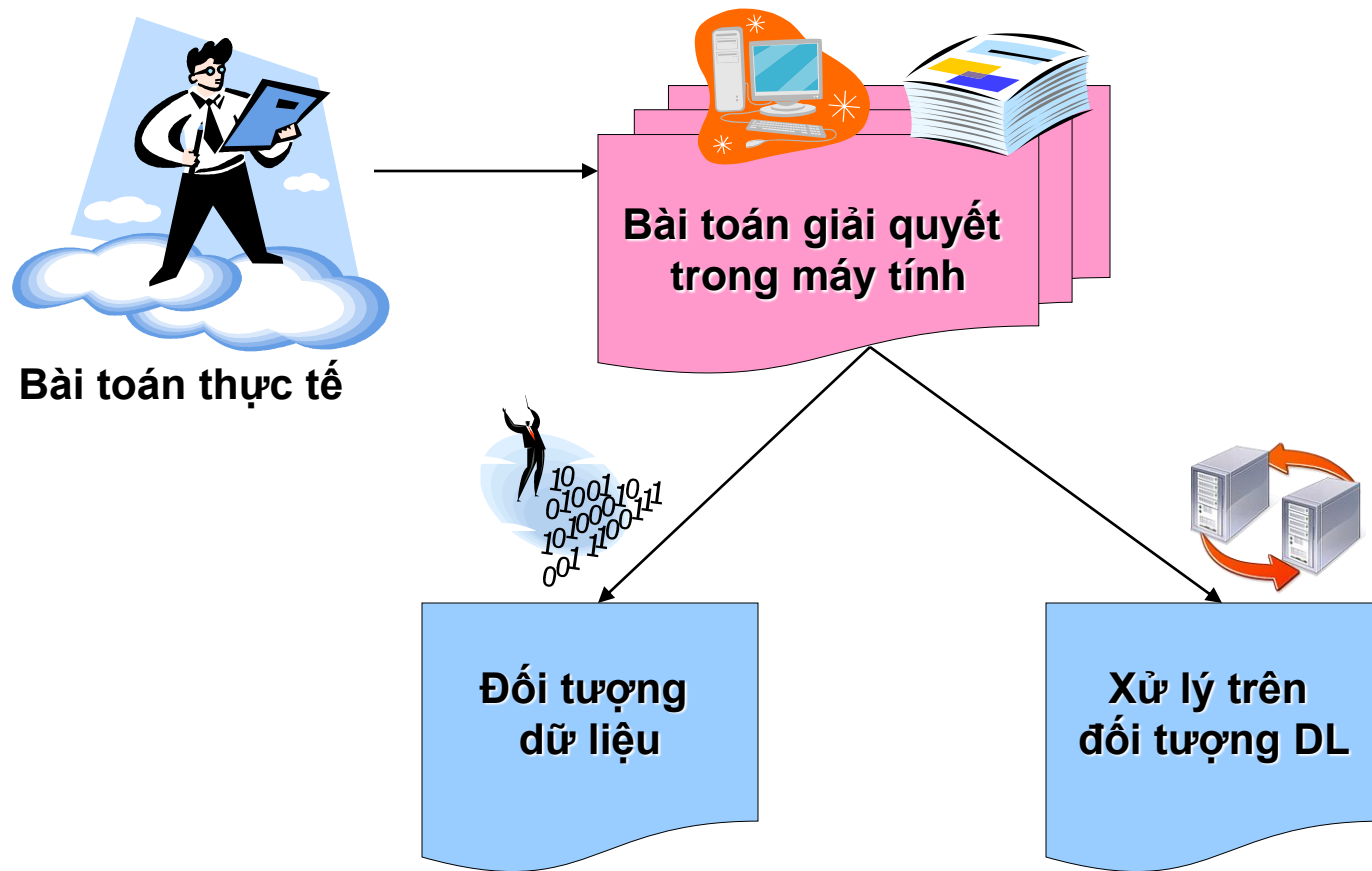
1.2 Cấu trúc dữ liệu và các vấn đề liên quan

1.3 Các phương pháp thiết kế giải thuật

1.4 Phân tích giải thuật

1.1 Giải thuật và cấu trúc dữ liệu

- Vai trò của cấu trúc dữ liệu trong một dự án tin học



1.1 Giải thuật và cấu trúc dữ liệu

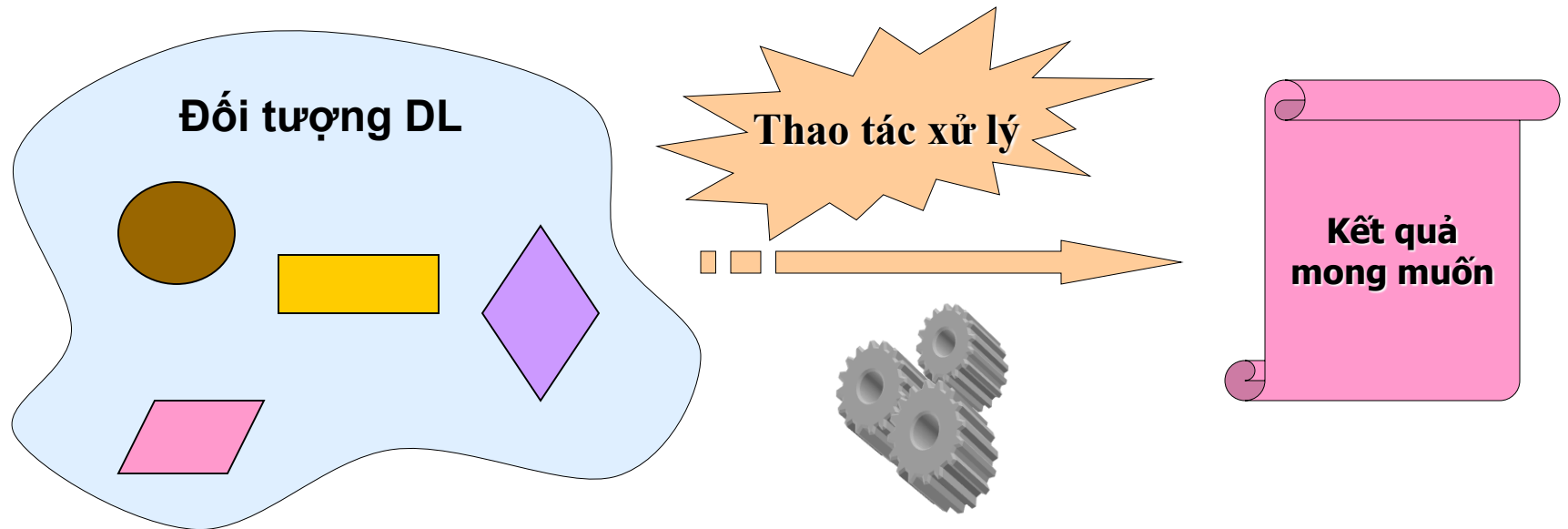
- Dữ liệu thực tế
 - Muôn hình vạn trạng, đa dạng, phong phú
 - Thường có chứa đựng quan hệ với nhau
- Cần phải tổ chức biểu diễn thành cấu trúc thích hợp nhất
 - Phản ánh chính xác dữ liệu thực tế
 - Dễ dàng xử lý trong máy tính



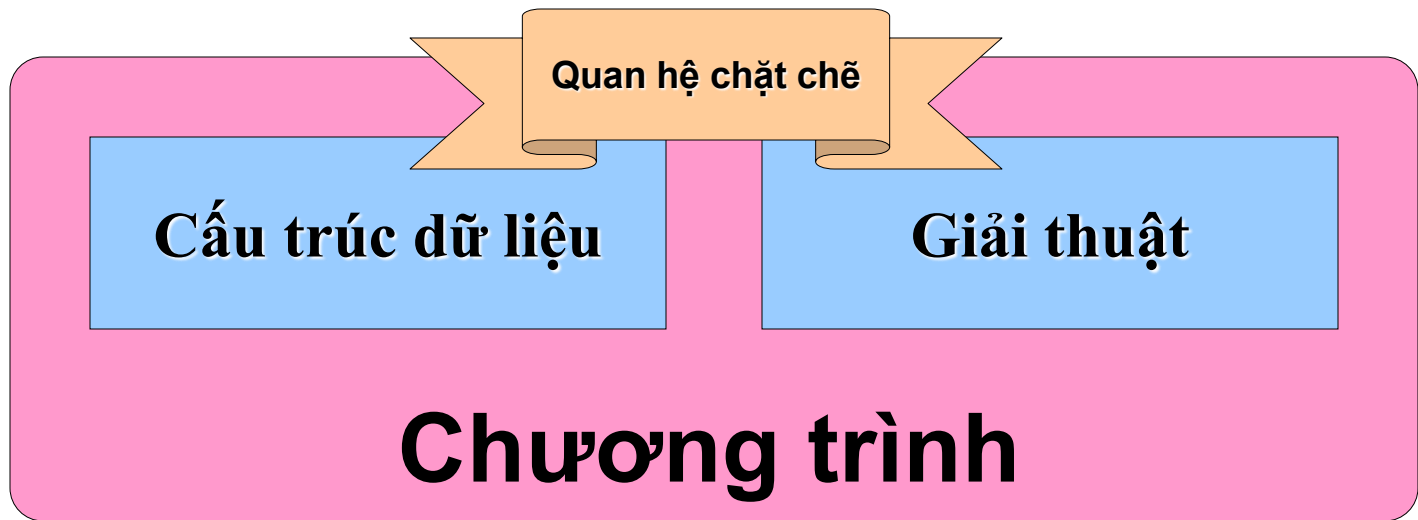
**Xây dựng
CTDL**

1.1 Giải thuật và cấu trúc dữ liệu

Dựa trên Y/C cụ thể, xác định các trình tự giải quyết vấn đề trên máy tính để đưa kết quả mong muốn



1.1 Giải thuật và cấu trúc dữ liệu



1.1 Giải thuật và cấu trúc dữ liệu

- Có mối quan hệ mật thiết
 - Giải thuật phản ánh phép xử lý, còn đối tượng xử lý của giải thuật là dữ liệu.
 - Với CTDL đã chọn sẽ có những giải thuật tương ứng phù hợp.
 - Khi CTDL thay đổi thì GT cũng thay đổi tránh xử lý gượng ép, thiếu tự nhiên trên cấu trúc ko thích hợp.
 - CTDL tốt giúp giải thuật xử lý phát huy tốt đa khả năng.

1.1 Giải thuật và cấu trúc dữ liệu

Ví dụ: Quản lý điểm học sinh: gồm có 4 điểm, và 3 học sinh
Thao tác duy nhất là xuất điểm số từng môn học của học sinh!

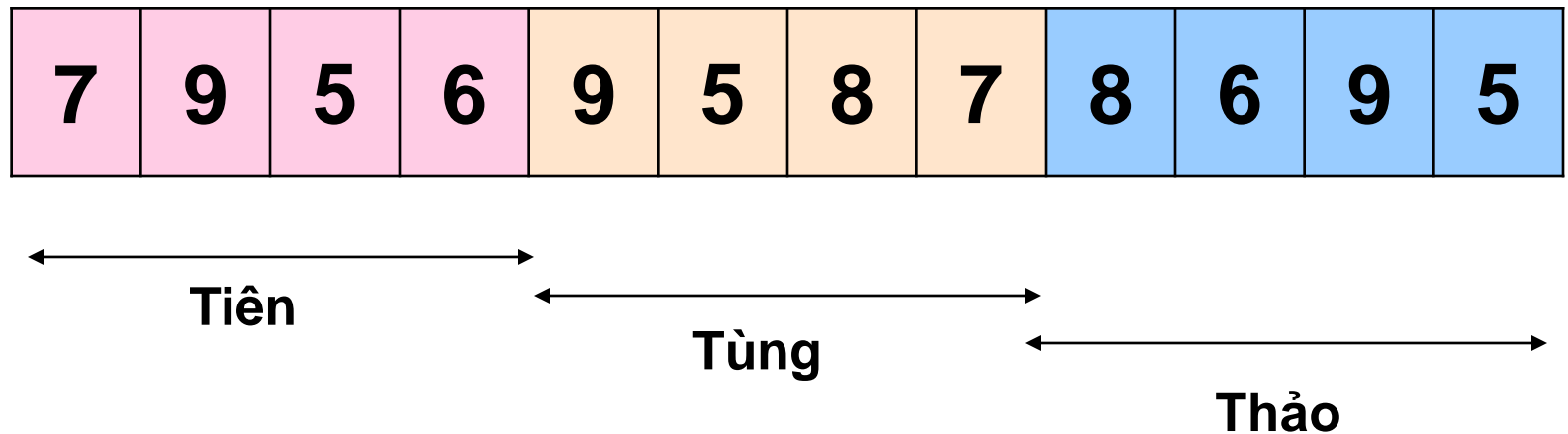
Học sinh	Toán	Lý	Hoá	Văn
Tiên	7	9	5	6
Tùng	9	5	8	7
Thảo	8	6	9	5

Tổ chức dữ liệu và xử lý



1.1 Giải thuật và cấu trúc dữ liệu

- Phương án A: dùng mảng 1 chiều
 - `int result[12] = { 7, 9, 5, 6, 9, 5, 8, 7, 8, 6, 9, 5 };`



1.1 Giải thuật và cấu trúc dữ liệu

- Truy xuất điểm môn j của $hs\ i$ là phần tử tại dòng i và cột j
 - Bảng điểm(i, j) \rightarrow result[(($i-1$)*số cột)+ j]
- Ngược lại
 - Result[i] \rightarrow Bảng điểm(dòng((i /số cột)+1), cột($i\%$ số cột))

1.1 Giải thuật và cấu trúc dữ liệu

- Thao tác xử lý như sau:

```
void          XuatDiem()
{
    const    int    so_mon = 4;
    int      sv, mon;
    for( int i=0; i < 12; i++)
    {
        sv = i/so_mon;
        mon = i % so_mon;
        cout<<"Diem mon"<<mon<<"cua sinh vien"<<mon<<
        "la:"<<result[i];
    }
}
```

1.1 Giải thuật và cấu trúc dữ liệu

■ Phương án B:

- Sử dụng mảng 2 chiều
- `int result[3][4] = { {7, 9, 5, 6},
 {9, 5, 8, 7},
 {8, 6, 9, 5} };`

	Cột 0	Cột 1	Cột 2	Cột 3
Dòng 0	7	9	5	6
Dòng 1	9	5	8	7
Dòng 2	8	6	9	5

1.1 Giải thuật và cấu trúc dữ liệu

- Truy xuất điểm số môn j của học sinh i là phần tử dòng i và cột j trong bảng
 - Bảng điểm(dòng i , cột j) \rightarrow `result[i][j]`

1.1 Giải thuật và cấu trúc dữ liệu

```
void      XuatDiem()  
  
{  
  
    int so_mon = 4, so_sv=3;  
  
    for(int i=0; i < so_sv; i++)  
  
        for(int j=0; j<so_mon; j++)  
  
            cout<<"Diem mon"<<j<<"cua sv"<<i<<  
            "la: ", result[i][j];  
  
}
```

1.1 Giải thuật và cấu trúc dữ liệu

- Nhận xét: về 2 phương án
 - Phương án B cung cấp cấu trúc dữ liệu phù hợp với thực tế hơn!
 - Do đó giải thuật xây dựng trên CTDL B cũng đơn giản và tự nhiên hơn.

1.2 CTDL và các vấn đề liên quan

Các tiêu chuẩn đánh giá CTDL

- Phản ánh đúng thực tế:
 - Thể hiện được đầy đủ thông tin nhập/xuất của bài toán.
- VD:
 - Chọn kiểu dữ liệu nguyên (int) lưu trữ tiền thưởng bán hàng (TienThuong)
 - $TienThuong = \text{trị giá hàng} * 5\%$



1.2 CTDL và các vấn đề liên quan

Các tiêu chuẩn đánh giá CTDL

- Phù hợp với thao tác xử lý:
 - Tăng tính hiệu quả của chương trình → hiệu quả của dự án tin học.
- VD:
 - Dữ liệu được cập nhật thêm, xoá liên tục nên dùng cấu trúc danh sách liên kết!
 - Dữ liệu có kích thước cố định, không thêm xoá thì dùng mảng (có thể tĩnh)

1.2 CTDL và các vấn đề liên quan

Các tiêu chuẩn đánh giá CTDL

- Tiết kiệm tài nguyên hệ thống:
 - Sử dụng tài nguyên vừa đủ để thực hiện chức năng & nhiệm vụ.
- VD:
 - Lưu tháng hiện hành sử dụng kiểu char (1byte) là được.
 - Lưu danh sách sinh viên nên sử dụng danh sách liên kết hơn là cấp phát bộ nhớ trước.

1.2 CTDL và các vấn đề liên quan

Các tiêu chuẩn đánh giá CTDL

- Sự thích hợp giữa CTDL & NNLT:
 - Dễ dàng cài đặt trên ngôn ngữ được chọn
- VD:

1.2 CTDL và các vấn đề liên quan

Thuật toán

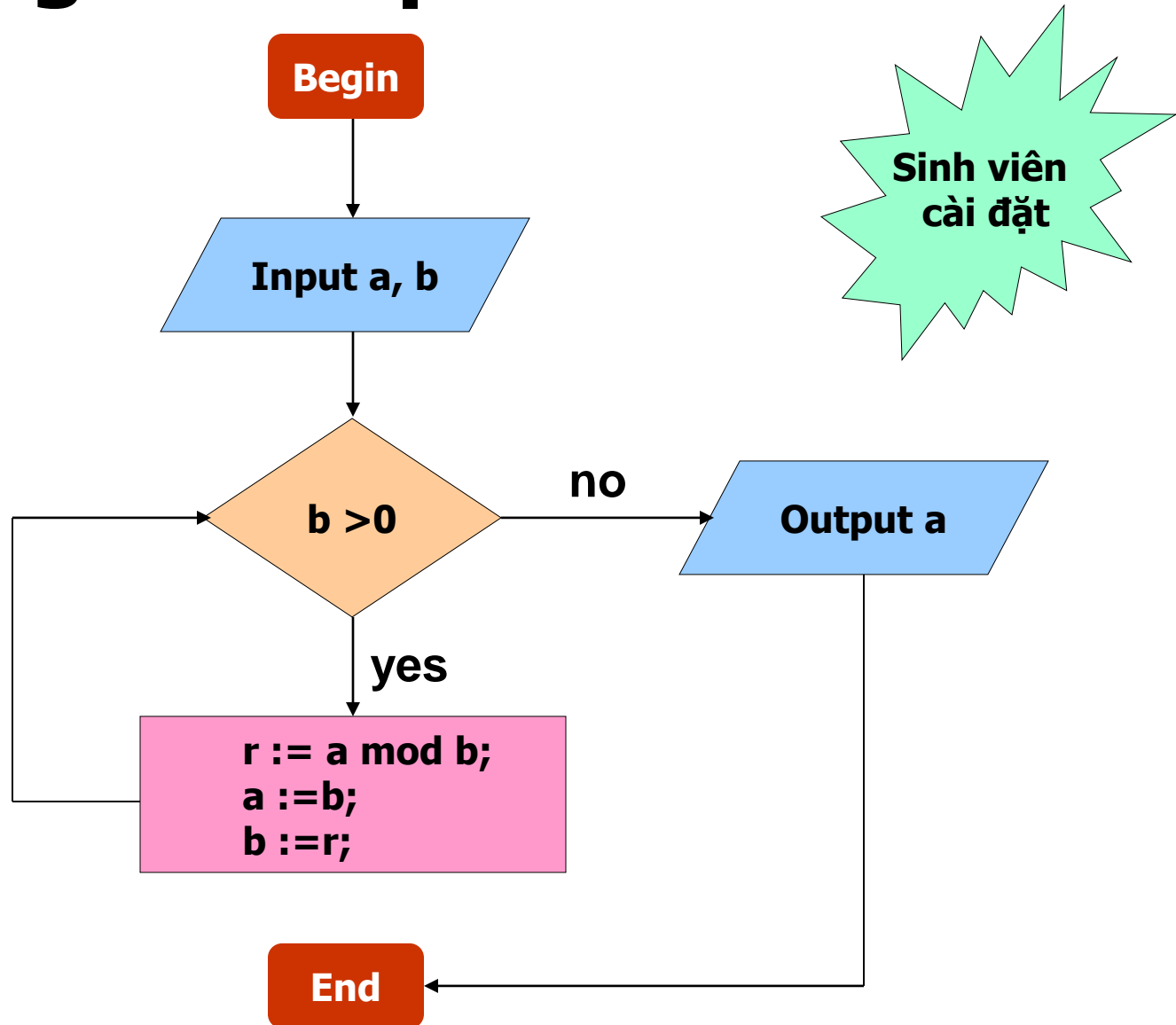
- Là một tập hợp các bước các thao tác, các công việc có thể biểu diễn được trên máy tính và được sắp xếp theo trật tự nhất định để đạt được mục đích mong muốn.
- Bao gồm:
 - Một bộ dữ liệu vào
 - Sau một số hữu hạn bước thực hiện thao tác chỉ ra.
 - Kết quả đạt được theo mục tiêu đã định.

1.2 CTDL và các vấn đề liên quan

Ví dụ về giải thuật

- Input: hai số nguyên a và b khác 0
- Output: ước số chung lớn nhất của a và b
- Các bước thực hiện như sau (Euclidean)
- B1: Nhập a và b : số tự nhiên
- B2: nếu $b \neq 0$ sang B3, ngược lại qua B4
- B3: đặt $r = a \% b$; $a = b$; $b = r$; quay lại B2
- B4: kết quả USCLN là giá trị a . Kết thúc thuật toán

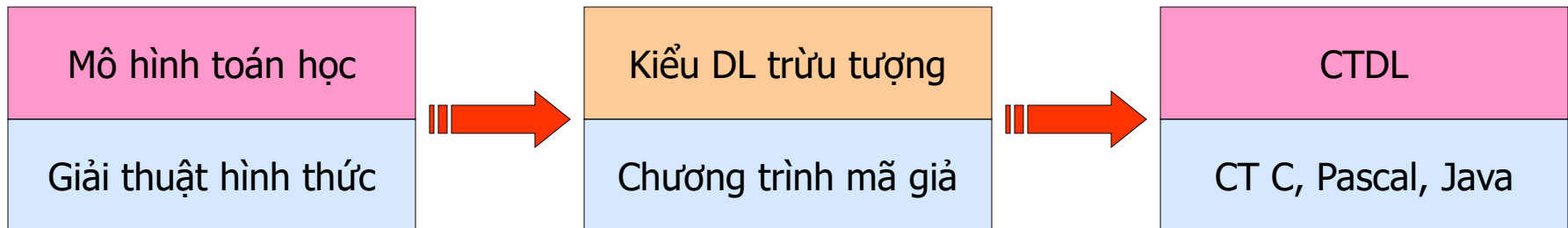
VD về giải thuật



1.2 CTDL và các vấn đề liên quan

Các bước tiếp cận bài toán

1. Mô hình hoá bài toán
2. Tìm giải thuật trên mô hình đó
3. Hình thức hoá giải thuật thông qua các thủ tục hay mã giả
4. Cài đặt giải thuật trên NNLT cụ thể



1.3 Phương pháp thiết kế giải thuật

- Modul hóa và việc giải quyết bài toán
 - Chiến thuật giải quyết bài toán: “Chia để trị”
 - Sử dụng cách thiết kế từ trên xuống (top – down)
- Phương pháp tinh chỉnh từng bước.
 - Là phương pháp thiết kế giải thuật gắn liền với lập trình
 - PP này phản ánh tinh thần của quá trình modul hóa bài toán và thiết kế top - down

1.3 Phương pháp thiết kế giải thuật

- Phương pháp tinh chỉnh từng bước.



1.3 Phương pháp thiết kế giải thuật

■ Ngôn ngữ tự nhiên

- **Bước 1: duyệt tuần tự** từ phần tử đầu tiên;
- **Bước 2:** so sánh các phần tử trong danh sách với khóa tìm kiếm có hai khả năng
 - Nếu bằng nhau \Rightarrow Tìm thấy \Rightarrow Dừng
 - Nếu khác nhau chuyển Sang bước 3
- **Bước 3:** xét phần tử kế tiếp trong mảng
 - Nếu hết mảng, không tìm thấy. \Rightarrow Dừng
 - Nếu chưa hết mảng quay lại bước 2

1.3 Phương pháp thiết kế giải thuật

■ Giải ngôn ngữ:

- **Bước 1:** $i = 0$;
- **Bước 2:** So sánh $a[i]$ với x , có hai khả năng
 - $a[i] = x$: Tìm thấy. \Rightarrow Dừng
 - $a[i] \neq x$: Sang bước 3
- **Bước 3:** $i = i + 1$ // xét phần tử kế tiếp trong mảng
 - Nếu $i > N$: Hết mảng, không tìm thấy. \Rightarrow Dừng
 - Nếu $i \leq N$: Quay lại bước 2

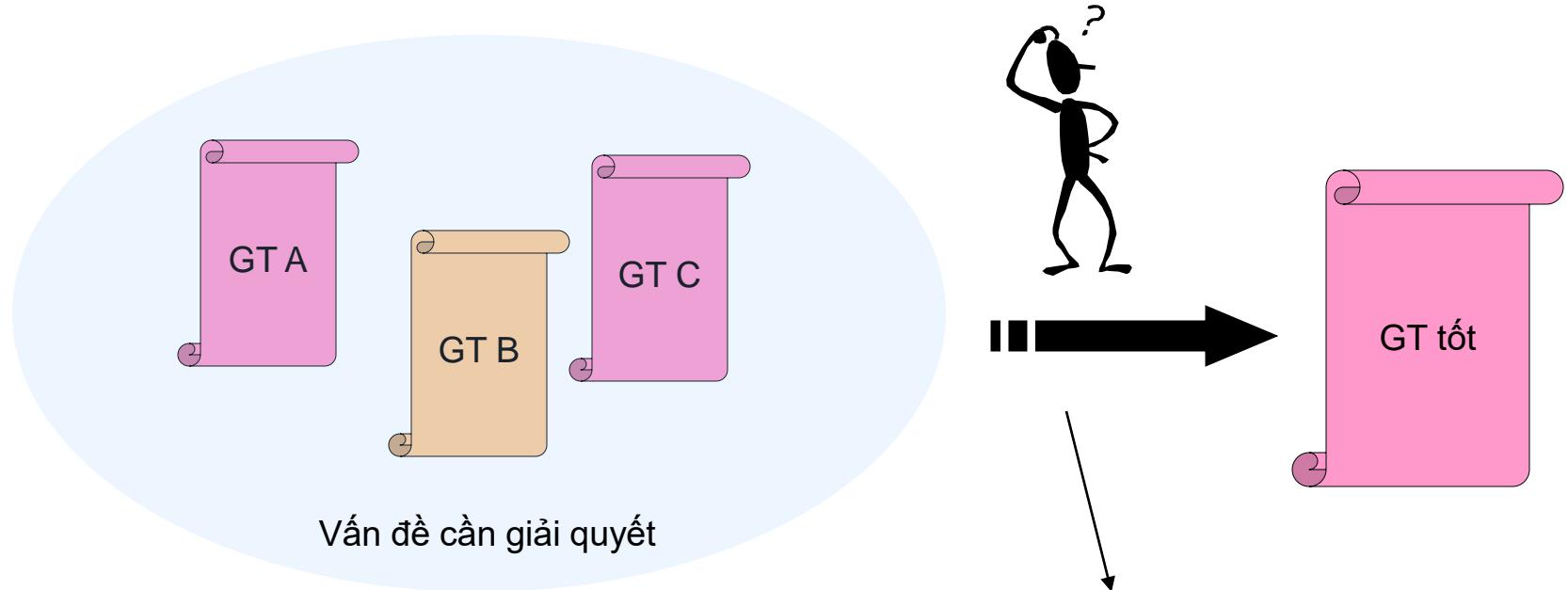
1.3 Phương pháp thiết kế giải thuật

■ Ngôn ngữ lập trình (C++)

```
int      Search(int a[], int n, int x)
{
    int i =0;
    while (i<n) && ( a[i] != x)
        i++;
    if (i >= n)
        return -1; // tìm không thấy
    else
        return i;  // tìm thấy tại vị trí i
}
```

1.4 Phân tích giải thuật

■ Sự cần thiết phân tích giải thuật



1. Giải thuật đúng
2. Giải thuật đơn giản
3. Giải thuật thực hiện nhanh

1.4 Phân tích giải thuật

- Thời gian thực hiện giải thuật phụ thuộc vào:
 - Giải thuật
 - Tập chỉ thị của máy tính
 - Cấu hình của máy tính (tốc độ)
 - Kỹ năng của người lập trình
- Tính phức tạp của thời gian được tiếp cận theo sự đo lường cơ bản của việc thực thi.
- Thời gian thực hiện một chương trình là một hàm theo kích thước dữ liệu vào: $T(n)$, n là kích thước của dữ liệu vào.

1.4 Phân tích giải thuật

Thời gian thực hiện

- Đơn vị của $T(n)$: theo số lệnh được thực hiện.
- $T(n) = C_n$ thì CT cần C_n chỉ thị thực thi
- Thời gian thực hiện **xấu nhất**: do tính chất dữ liệu cũng ảnh hưởng
 - VD chương trình sắp xếp sẽ cho thời gian khác nhau với các bộ DL có thứ tự khác nhau!
- ☞ $T(n)$ thường được xem là TG chương trình thực hiện xấu nhất trên DL kích thước n .

Tỉ suất tăng

- Hàm ko âm $T(n)$ có tỉ suất tăng $f(n)$ nếu tồn tại các hằng số C và N_0 sao cho:
 - $T(n) < Cf(n)$, với mọi $n \geq N_0$
 - “cho hàm ko âm $T(n)$ bất kỳ, ta luôn tìm được tỉ suất tăng $f(n)$ của nó”

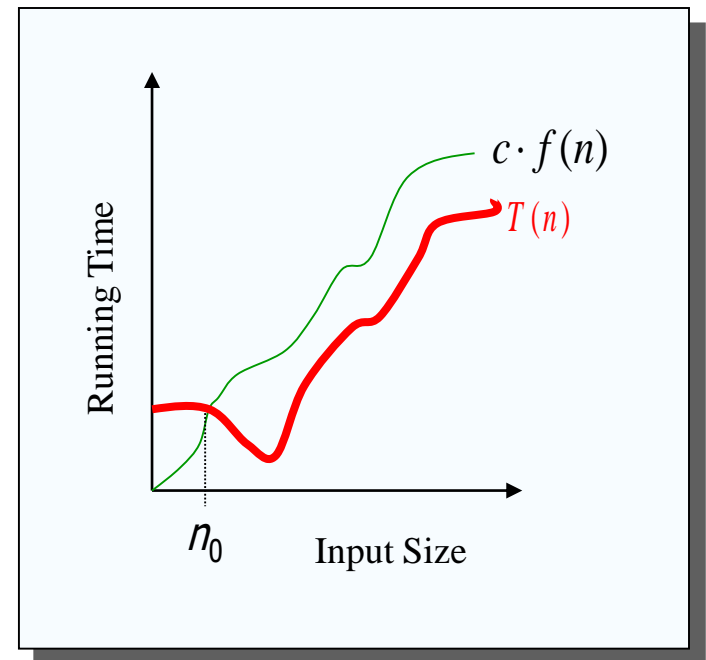
■ Giả sử $T(0) = 1$, $T(1) = 4$,
tổng quát $T(n) = (n+1)^2$.

Đặt $N_0 = 1$, và $C = 4$, thì

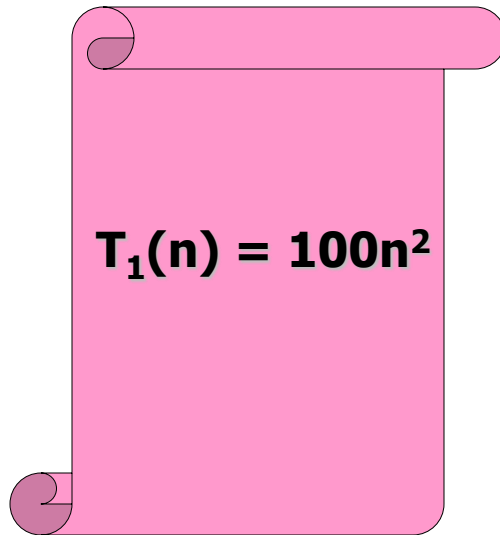
$\forall n \geq 1$, chứng minh được rằng

$T(n) = (n+1)^2 \leq 4n^2$, $\forall n \geq 1$,

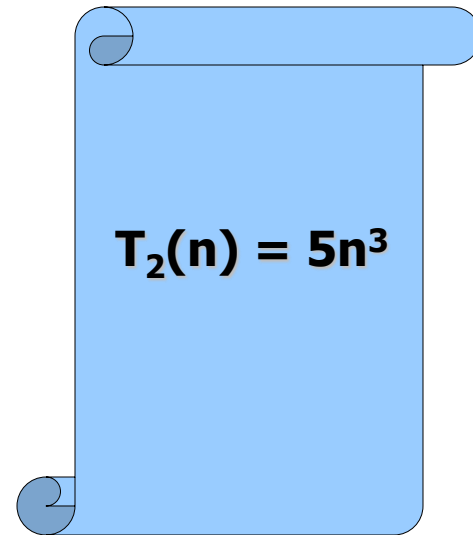
tỉ suất tăng khi đó n^2



1.4 Phân tích giải thuật

A pink scroll with a rolled-up top and bottom edge, containing the text $T_1(n) = 100n^2$.
$$T_1(n) = 100n^2$$



A blue scroll with a rolled-up top and bottom edge, containing the text $T_2(n) = 5n^3$.
$$T_2(n) = 5n^3$$

- Khi n đủ lớn: $n > 20$, thì $T_1(n) < T_2(n)$
- Cách hợp lý nhất là xét tỷ suất tăng của hàm TG thực hiện CT thay vì chính bản thân thời gian thực hiện.

1.4 Phân tích giải thuật

Độ phức tạp giải thuật

- Cho hàm $T(n)$, $f(n)$ được gọi là độ phức tạp tính toán của $T(n)$ nếu tồn tại các hằng C , n_0 đủ lớn sao cho khi $n \geq n_0$ thì $T(n) \leq Cf(n)$
- Nói cách khác độ phức tạp tính toán giải thuật là hàm chặn trên của hàm thời gian
 - VD: $T(n) = (n+1)^2$ có tỷ suất tăng là n^2 nên $T(n) = (n+1)^2$ là $O(n^2)$.
 - Lưu ý: Với C là hằng số.
 $O(C) = O(1)$.

1.4 Phân tích giải thuật

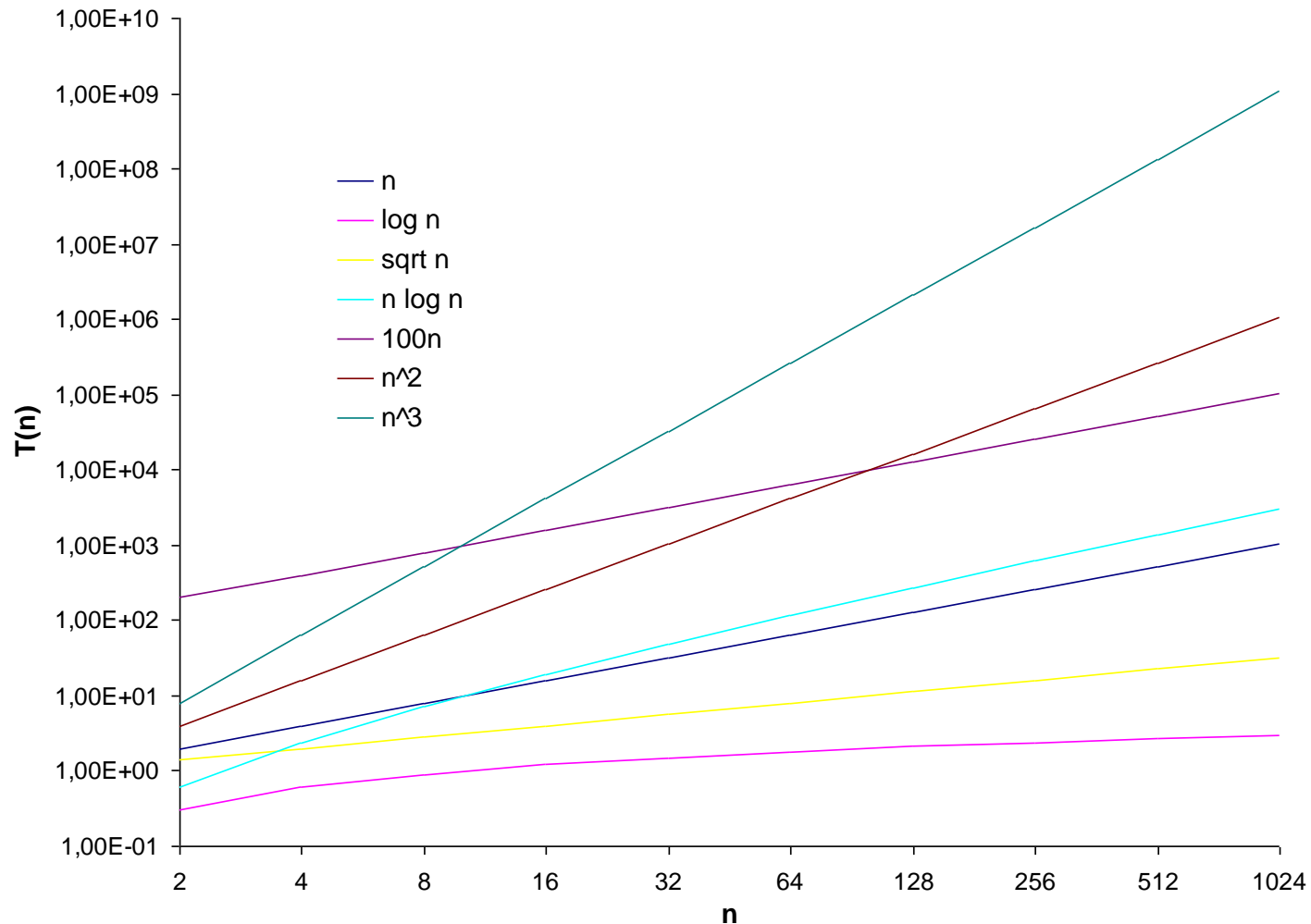
Độ phức tạp giải thuật

- Các độ phức tạp thường gặp:
 - $\log_2 n$, n , $n \log_2 n$, n^2 , n^3 , 2^n , $n!$, n^n

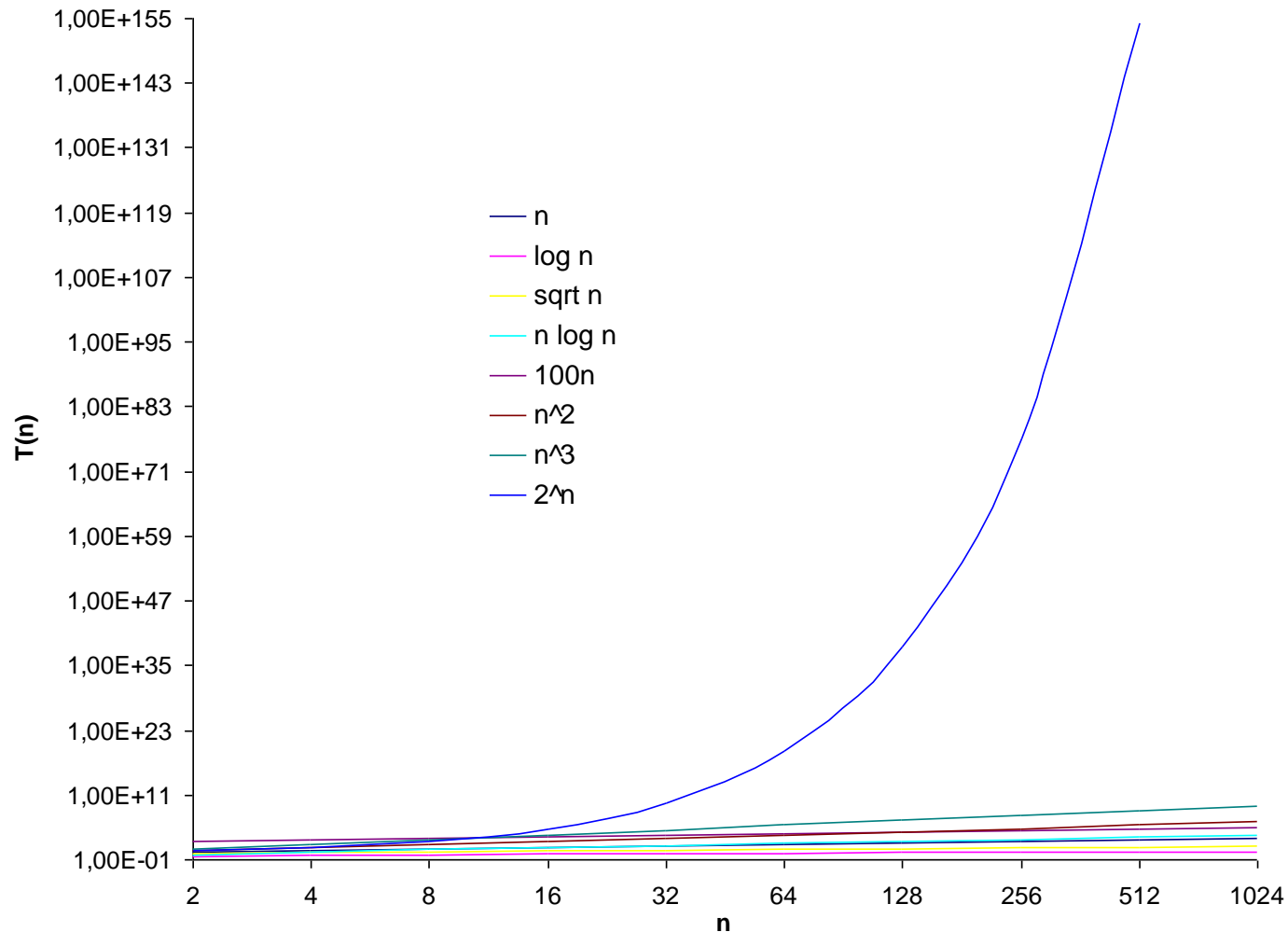
n	$\log_2 n$	2^n
16	64	256
32	160	2.147.483.648

- Thông thường thuật giải có độ phức tạp đa thức thì có thể cài đặt
- Còn phức tạp ở mức hàm mũ thì phải cải tiến giải thuật!

Hàm tăng trưởng



Hàm tăng trưởng



1.4 Phân tích giải thuật

Quy tắc tính độ phức tạp

- Thời gian thực hiện lệnh gán, đọc/ghi dữ liệu là $O(1)$.
- Thời gian thực hiện một chuỗi tuần tự các lệnh được xác định bằng **quy tắc cộng**.
- Thời gian thực hiện cấu trúc IF là thời gian lớn nhất thực hiện sau THEN hoặc ELSE và thời gian điều kiện. Thường thời gian điều kiện là $O(1)$.

1.4 Phân tích giải thuật

Quy tắc tính độ phức tạp

- Thời gian thực hiện vòng lặp là tổng thời gian thực hiện thân vòng lặp. Nếu thời gian thực hiện thân vòng lặp ko đổi thì tg thực hiện vòng lặp là tích số lần lặp với thời gian thực hiện thân vòng lặp
- Thời gian thực hiện một chuỗi các lệnh lồng nhau được xác định bằng **quy tắc nhân**.

1.4 Phân tích giải thuật

Quy tắc cộng

- Cho giải thuật P_1 có thời gian thực hiện giải thuật $T_1 = O(f(n))$
- Cho giải thuật P_2 có thời gian thực hiện giải thuật $T_2 = O(g(n))$
- Quy tắc cộng
Nếu P_1 tiếp theo P_2
Hoặc P_2 tiếp theo P_1 } $T_1 + T_2 = O(\max(f(n), g(n)))$
- Ví dụ
Nếu $T_1 = O(n)$, $T_2 = O(n^2) \implies T_1 + T_2 = O(n^2)$

1.4 Phân tích giải thuật

Quy tắc nhân

- Cho giải thuật P_1 có thời gian thực hiện giải thuật $T_1 = O(f(n))$
- Cho giải thuật P_2 có thời gian thực hiện giải thuật $T_2 = O(g(n))$
- Quy tắc nhân
Nếu P_1 lồng trong P_2
Hoặc P_2 lồng trong P_1 } $T_1 * T_2 = O(f(n)*g(n))$
- Ví dụ
Nếu $T_1 = O(n)$, $T_2 = O(n^2) \implies T_1 + T_2 = O(n^3)$

1.4 Phân tích giải thuật

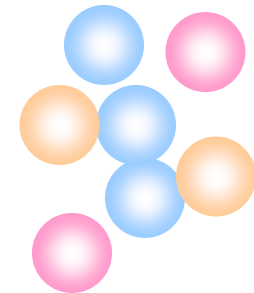
Quy tắc tính độ phức tạp

- Để tính thời gian thực toàn bộ giải thuật ta cần tìm ra phép toán mà thời gian thực hiện của nó không ít hơn thời gian thực hiện của các phép toán khác (đó là phép toán tích cực). Tính thời gian thực hiện phép toán đó, đó chính là thời gian thực hiện giải thuật

VD tính độ phức tạp

■ VD giải thuật sắp xếp nổi bọt

- (1) **for** ($i = 0; i < n-1; i++$)
- (2) **for** ($j=n-1; j > i; j--$)
- (3) **if** ($a[j-1] > a[j]$)
- (4) { **temp = a[j-1];**
- (5) $a[j-1] = a[j];$
- (6) $a[j] = temp;$
- (7) }



**Phép toán
tích cực**

1.4 Phân tích giải thuật

Quy tắc tính độ phức tạp

- Có những trường hợp, thời gian thực hiện giải thuật không chỉ phụ thuộc vào kích thước của dữ liệu vào mà còn phụ thuộc vào tình trạng của dữ liệu vào, khi đó ta phải tính thời gian thực hiện giải thuật trong 3 trường hợp: tốt nhất, xấu nhất và trung bình

VD tính độ phức tạp

■ VD hàm tìm kiếm tuần tự

- (1) `i=0;`
- (2) `found = false;`
- (3) `while (i<n && !found)`
- (4) `if (a[i] == x)`
- (5) `found = true;`
- (6) `else`
- (7) `i++;`
- (8) `return found;`



Tổng kết

Với CTDL đã chọn sẽ có những giải thuật tương ứng phù hợp.

Mối
quan hệ
giữa giải
thuật và
CTDL



Khi CTDL thay đổi thì giải thuật cũng thay đổi theo

Thiết kế
giải
thuật



Phương pháp tinh chỉnh từng bước

Giả ngôn ngữ (Pseudo Language)

Độ phức
tạp tính
toán của
giải
thuật



Khái niệm

Các hàm độ phức tạp thường gặp

Quy tắc tính độ phức tạp

Bài tập

1. Xác định độ phức tạp tính toán của giải thuật (O)

a.

```
sum = 0;
for (i=1; i<=n; i++)
{
    cin>>x;
    sum = sum + x;
}
```


Bài tập

1. Xác định độ phức tạp tính toán của giải thuật (O)

b.

```
for (i=1; i<=n; i++)  
    for (j=1; j<=n; j++)  
    {  
        C[i, j] = 0;  
        for (k=1; k<=n; k++)  
            C[i, j] = C[i, j] + A[i, k] * B[k, j];  
    }
```

Bài tập

2. Hãy nêu một giải thuật mà độ phức tạp tính toán là $O(1)$.

Tài liệu tham khảo

- [1]. Giáo trình Cấu trúc dữ liệu và giải thuật – Lê Văn Vinh, NXB Đại học quốc gia TP HCM, 2013
- [2]. Cấu trúc dữ liệu & thuật toán, Đỗ Xuân Lôi, NXB Đại học quốc gia Hà Nội, 2010.
- [3]. Trần Thông Quế, *Cấu trúc dữ liệu và thuật toán (phân tích và cài đặt trên C/C++)*, NXB Thông tin và truyền thông, 2018
- [4]. Robert Sedgewick, *Cẩm nang thuật toán*, NXB Khoa học kỹ thuật, 2004 .
- [5]. PGS.TS Hoàng Nghĩa Tý, *Cấu trúc dữ liệu và thuật toán*, NXB xây dựng, 2014

