Học viện
Công nghệ Bưu chính Viễn thông

KHỞI ĐỘNG

ĐỘ PHỨC TẠP THUẬT TOÁN

Nội dung

Giải thuật và cấu trúc dữ liệu

Phân tích và thiết kế giải thuật

- Thiết kế giải thuật
- Phân tích giải thuật

Một số lớp các giải thuật

Giải bài toán bằng máy tính

Hai yếu tố tạo nên một chương trình máy tính

- Cấu trúc dữ liệu
- Giải thuật

Cấu trúc dữ liệu + Giải thuật = Chương trình

Giải thuật

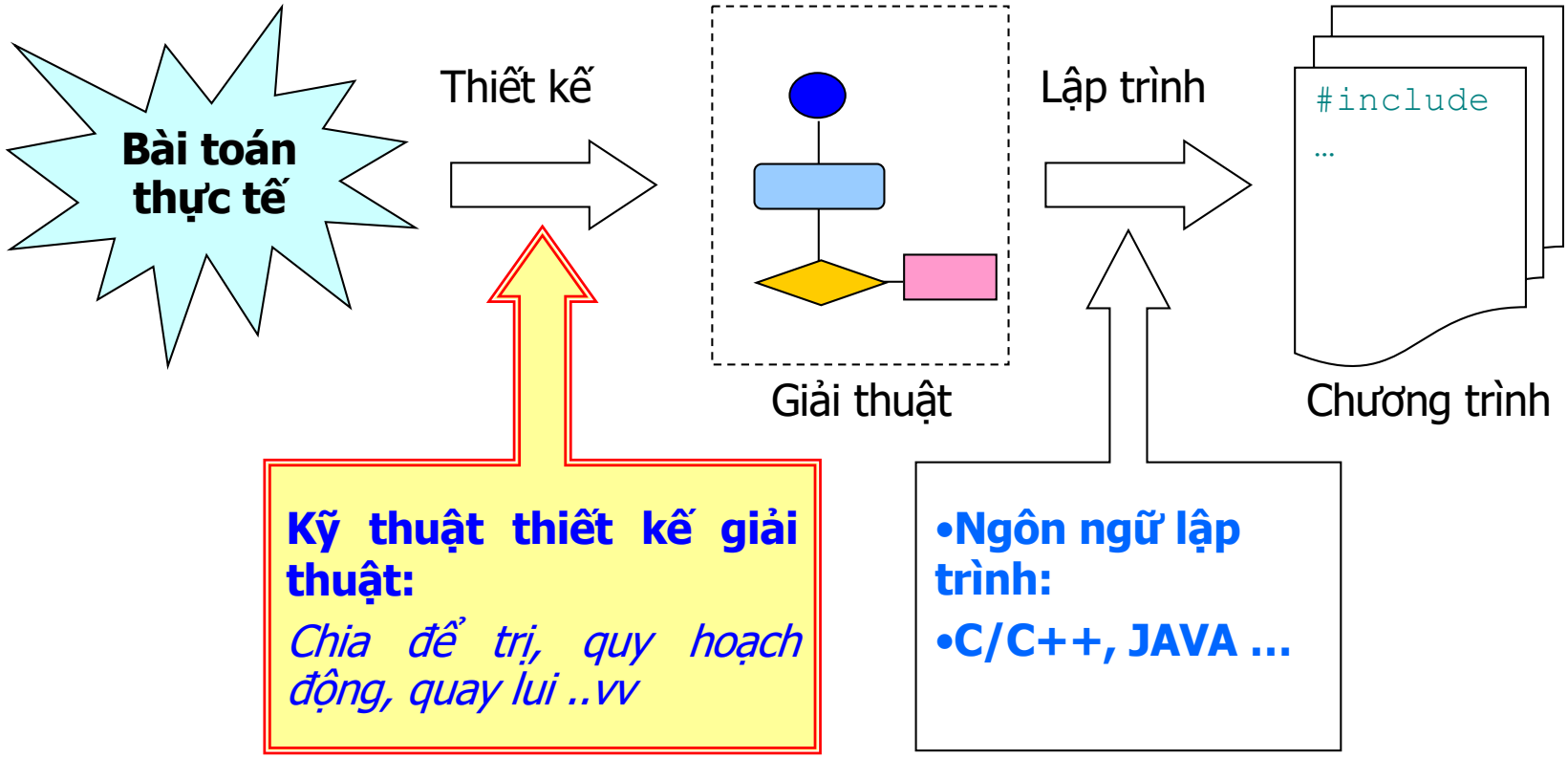
- **Định nghĩa**: là dãy các câu lệnh chặt chẽ và rõ ràng xác định một trình tự các thao tác trên một số đối tượng nào đó, sao cho sau một số hữu hạn bước thực hiện ta đạt được kết quả mong muốn
- Mỗi thuật toán có một **dữ liệu vào (Input)** và một **dữ liệu ra (Output)**;

Đặc trưng của giải thuật

- Tính xác định
- Tính dừng (hữu hạn)
- Tính đúng đắn
- Tính phổ dụng
- Tính khả thi

Thiết kế giải thuật

Từ bài toán đến chương trình



Thiết kế giải thuật

- Với một vấn đề đặt ra, làm thế nào để đưa ra thuật toán giải quyết nó?
- Chiến lược thiết kế:
  - Tham lam (greedy method)
  - Chia-để-trị (divide-and-conquer)
  - Quy hoạch động (dynamic programming)
  - Quay lui (backtracking)
  - …

Phân tích Giải thuật

Khi một giải thuật được xây dựng, các yêu cầu đặt ra

- Yêu cầu về tính đúng đắn của giải thuật
- Tính đơn giản của giải thuật.
- Yêu cầu về không gian (bộ nhớ)
- Yêu cầu về thời gian

Phân tích thời gian chạy của giải thuật

Thời gian thực hiện giải thuật phụ thuộc vào các yếu tố sau:

- Dữ liệu vào
- Tốc độ thực hiện các phép toán của máy tính (phần cứng máy tính)
- Trình biên dịch

Độ phức tạp tính toán

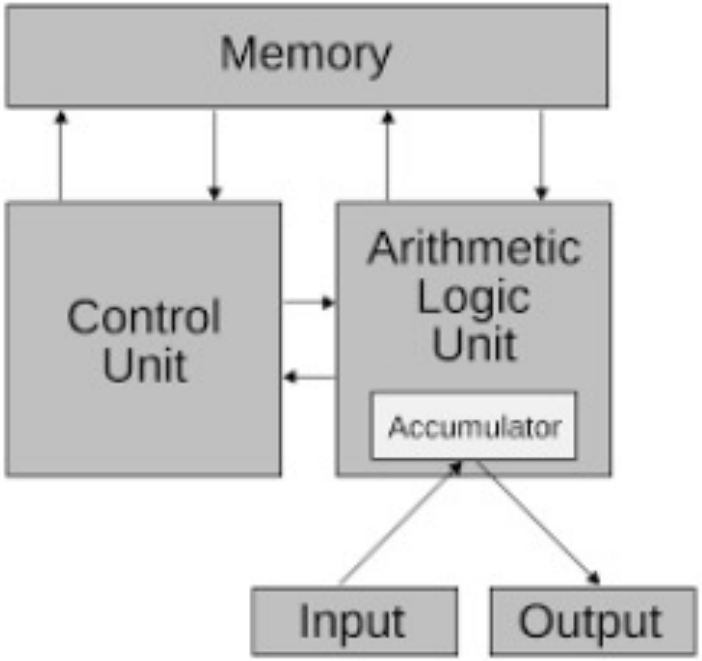
Một số yếu tố khác ảnh hưởng đến tốc độ tính toán

Tốc độ của máy tính

Độ phức tạp tính toán

Một số yếu tố khác ảnh hưởng đến tốc độ tính toán

Kiến trúc của máy tính



Độ phức tạp tính toán

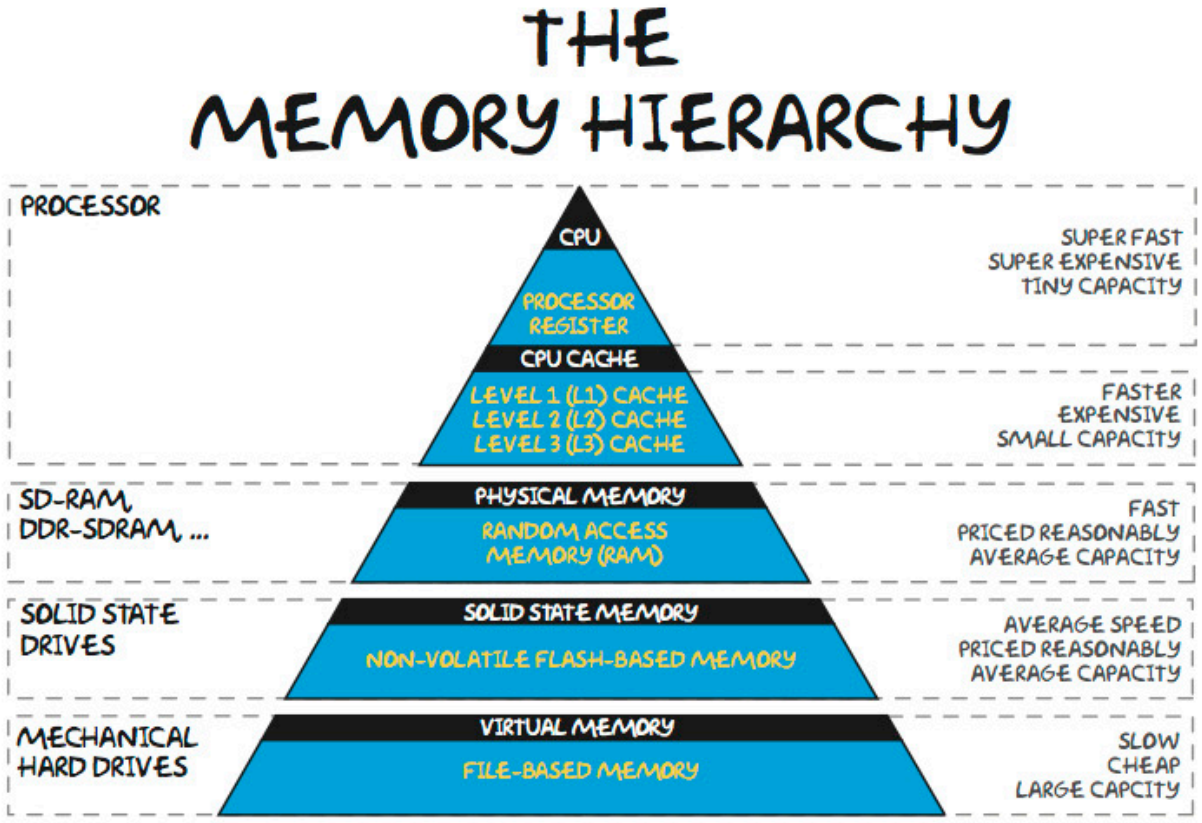
Một số yếu tố khác ảnh hưởng đến tốc độ tính toán

Trình biên dịch được sử dụng

Độ phức tạp tính toán

Một số yếu tố khác ảnh hưởng đến tốc độ tính toán

Phân cấp bộ nhớ



Độ phức tạp tính toán

Vấn đề

Việc tính toán thời gian chạy của chương trình là khó khăn

Đôi khi phụ thuộc vào những yếu tố ngay cả người lập trình không biết hoặc không kiểm soát được

Làm thế nào để đo được thời gian chạy mà không bị phụ thuộc những yếu tố này?

Độ phức tạp tính toán

- **Tính toán xấp xỉ**
  - Việc tính toán chính xác thời gian chạy của thuật toán là khó khăn vì phụ thuộc nhiều yếu tố
  - Cần tìm giải pháp dễ tính toán hơn, có thể không chính xác tuyệt đối nhưng lại dễ tính toán hơn mà vẫn phản ánh được tốc độ tính toán.
  - Tính toán tương đối dựa trên kích thước dữ liệu vào
    - Ví dụ: Thời gian chạy là n so với 1000n có vẻ chênh lệch nhưng nếu như so sánh với n^2 với input lớn thì 1000n vẫn cho thời gian tốt hơn n^2.

Ký hiệu Big-O

- Ký hiệu O (big-Oh):

  Hàm f(n) và g(n), ta nói:

  f(n) = O(g(n)), nếu tồn tại các hằng số dương c và n_o sao cho f(n) ≤ cg(n) khi n ≥ n_o.

- Ký hiệu này dùng để chỉ chặn trên của một hàm
- Ý nghĩa: Tốc độ tăng của hàm f(n) không lớn hơn hàm g(n)

Xác định độ phức tạp tính toán

T1(n) và T2(n) là thời gian thực hiện của hai giai đoạn chương trình P1 và P2 mà T1(n) = O(f(n)); T2(n) = O(g(n))

Quy tắc cộng:

- Thời gian thực hiện đoạn P1 rồi P2 tiếp theo sẽ là
- T1(n) + T2(n) = O(max(f(n),g(n))).

Quy tắc nhân:

- Thời gian thực hiện P1 và P2 lồng nhau sẽ là :
- T1(n)T2(n) = O(f(n)*g(n))

Các quy tắc tổng quát

Các phép gán, đọc, viết, goto là các phép toán sơ cấp:

- Thời gian thực hiện là: O(1)

Lệnh lựa chọn: if-else có dạng

if (<điều kiện>)

lệnh 1

else

lệnh 2

Các quy tắc tổng quát

Câu lệnh switch được đánh giá tương tự như lệnh if-else.

Các lệnh lặp: for, while, do-while

- Cần đánh giá số tối đa các lần lặp, giả sử đó là L(n)
- Tiếp theo đánh giá thời gian chạy của mỗi lần lặp là $T_i(n)$, (i=1,2,..., L(n))
- Mỗi lần lặp, chi phí kiểm tra điều kiện lặp,là $T_0(n)$.
- Chí phí lệnh lặp là: $\sum_{i=1}^{L(n)} \left(T_0(n) + T_i(n)\right)$

Một số ví dụ

Case1: for (i=0; i<n; i++)

```
    for (j=0; j<n; j++)
        k++;
```

$O(n^2)$

Case 2: for (i=0; i<n; i++)

```
        k++;
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            k++;
```

$O(n^2)$

Case 3: for (int i=0; i<n-1; i++)

```
        for (int j=0; j<i; j++)
            int k+=1;
```

$O(n^2)$

Một số ví dụ

```
int MaxSubSum1(const int a[], int n) {
        int maxSum=0;

        for (int i=0; i<n; i++)
                for (int j=i; j<n; j++) {
                        int thisSum=0;

                        for (int k=i; k<=j; k++)
                                thisSum+=a[k];

                        if (thisSum>maxSum)
                                maxSum=thisSum;
                }
        return maxSum;
}
```

$O(n^3)$

Một số ví dụ

```
int MaxSubSum2(const int  a[], int n) {
        int maxSum=0;

        for (int i=0; i<n; i++) {
                thisSum=0;
                for (int j=i; j<n; j++) {
                        thisSum+=a[j];

                        if (thisSum>maxSum)
                                maxSum=thisSum;
                }
        }
        return maxSum;
}
```

$O(n^2)$

Một số ví dụ

```
int MaxSubSum3(const int a[], int n) {
        int maxSum=0, thisSum=0;

        for (int j=0; j<n; j++) {
                thisSum+=a[j];

                if (thisSum>maxSum)
                        maxSum=thisSum;
                else if (thisSum<0)
                        thisSum=0;
        }
        return maxSum;
}
```

O(n)

Một số ví dụ

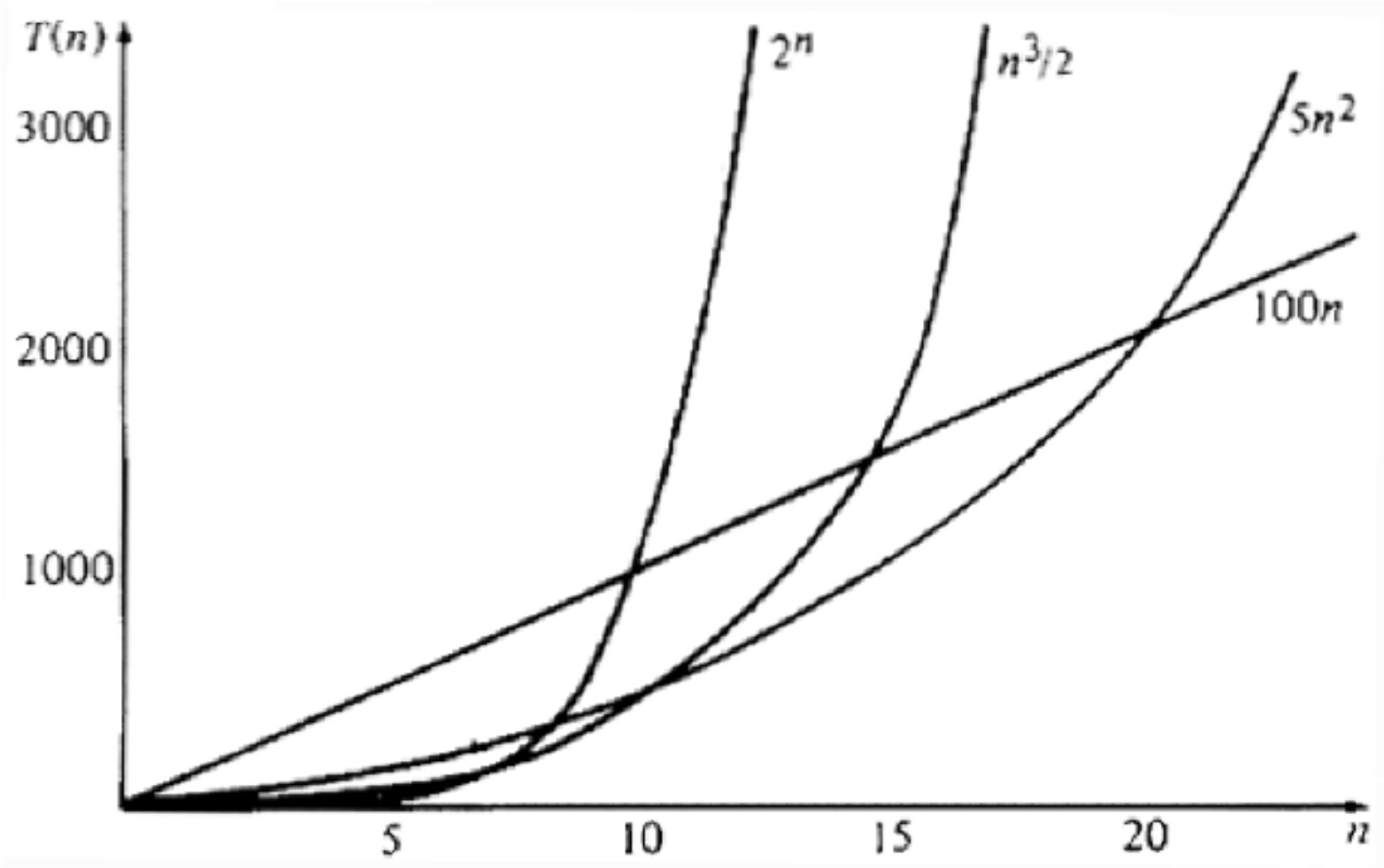
```
Sum=0
for (j=0;j<N;j++)
    for (k=0;k<N*N;k++)
        Sum++;
```

$O(N^3)$

Phân lớp giải thuật

- Độ phức tạp
  - O(1) độ phức tạp hằng số
  - O(logn) độ phức tạp logarit
  - O(n) độ phức tạp tuyến tính
  - O(nlogn) độ phức tạp nlogn
  - $O(n^b)$ độ phức tạp đa thức
  - $O(b^n)$ độ phức tạp mũ
  - O(n!) độ phức tạp giai thừa

Phân lớp giải thuật



Phân lớp giải thuật

Thời gian ước tính với máy tính có tốc độ 1Ghz

	n	$n \log n$	n^2	2^n
$n = 20$	1 sec	1 sec	1 sec	1 sec
$n = 50$	1 sec	1 sec	1 sec	13 day
$n = 10^2$	1 sec	1 sec	1 sec	$4 \cdot 10^{13}$ year
$n = 10^6$	1 sec	1 sec	17 min	
$n = 10^9$	1 sec	30 sec	30 year	
max n	10^9	$10^{7.5}$	$10^{4.5}$	30

Đánh giá độ phức tạp trong ba trường hợp

Độ phức tạp tính toán của giải thuật trong các trường hợp

- Xấu nhất
- Tốt nhất
- Trung bình

Đánh giá độ phức tạp trong ba trường hợp

Ví dụ: Thuật toán tìm kiếm tuần tự

```
int sequenceSearch(int x, int a[], int n){
        for (int i=0;i<n;i++){
                if (x==a[i]) return i;
          }
        return -1;
}
```

Đánh giá độ phức tạp trong ba trường hợp

- Tốt nhất: phần tử đầu tiên là phần tử cần tìm, số lượng phép so sánh là 2 → T(n) ~ O(2) = O(1)
- Xấu nhất: so sánh đến phần tử cuối cùng, số lượng phép so sánh là 2n → T(n) ~ O(n)
- Trung bình: so sánh đến phần tử thứ i, cần 2*i phép so sánh, vậy trung bình cần

(2+4+6+…+2n)/n=2(1+2+…+n)/n=n+1

→ T(n) ~ O(n)

Kinh nghiệm lập trình trực tuyến

Dựa trên đánh giá độ phức tạp tính toán:

- Nếu ước lượng số dòng lệnh không quá 10^7 thì thuật toán sẽ chạy tốt trong khoảng thời gian 1 giây
- Nếu ước lượng số dòng lệnh lớn hơn 10^8 thì nên suy nghĩ và lựa chọn giải pháp khác