TRƯỜNG ĐIỆN – ĐIỆN TỬ

KHOA ĐIỆN TỬ

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Nội dung chính

- Giới thiệu chung về tổ chức dữ liệu và giải thuật
- Cấu trúc dữ liệu
  - Cấu trúc mảng (Arrays)
  - Danh sách (Lists)
  - Ngăn xếp và hàng đợi (Stacks and Queues)
  - Cấu trúc cây (Trees)
  - Đồ thị (Graphs)
- Giải thuật
  - Đệ quy
  - Sắp xếp
  - Tìm kiếm

Đánh giá môn học

- Điểm quá trình : 30%
  - Điểm chuyên cần : Bài tập tại lớp + bài tập về nhà + bài kiểm tra (không báo trước)
  - Bài tập lớn
- Điểm cuối kỳ : 70 %

Tài liệu tham khảo

- Data Structures Using C, 2nd edition – Reema Thareja, Oxford University Express
- Cấu trúc dữ liệu và giải thuật – Đỗ Xuân Lôi, nxb Khoa học và Kỹ thuật
- Cấu trúc dữ liệu và thuật toán – Nguyễn Đức Nghĩa – nxb ĐHBK HN

Các khái niệm cơ bản về CTDL và giải thuật

- Giải thuật (algorithm):
  - Là một đặc tả chính xác và không nhập nhằng về một chuỗi các bước có thể được thực hiện một các tự động, để cuối cùng ta thu được các kết quả mong muốn.
  - Đặc tả (specification) : bản mô tả chi tiết và đầy đủ về một đối tượng hay một vấn đề

Giải thuật

- Một số yêu cầu của giải thuật
  - Đúng đắn,
  - Rõ ràng (không nhập nhằng),
  - Phải kết thúc sau một số hữu hạn bước thực hiện,
  - Có mô tả các đối tượng dữ liệu mà thuật toán sẽ thao tác như dữ liệu vào (nguồn), dữ liệu ra (đích) và các dữ liệu trung gian,
  - Thời gian thực hiện phải hợp lý.

Dữ liệu

- Dữ liệu (data):
  - Là các đối tượng mà thuật toán sẽ sử dụng để đạt được kết quả mong muốn. Nó cũng được dùng để biểu diễn cho các thông tin của bài boán như: các thông tin vào, thông tin ra (kết quả) và các các thông tin trung gian nếu cần.

Dữ liệu

- Dữ liệu gồm có hai mặt:
  - Mặt tĩnh (static):
    - xác định kiểu dữ liệu (data type).
    - cho biết cách tổ chức dữ liệu cũng như tập các giá trị mà một đối tượng dữ liệu có thể nhận, hay miền giá trị của nó.
    - Ví dụ như kiểu số nguyên, kiểu số thực,..
  - Mặt động (dynamic):
    - là trạng thái của dữ liệu như tồn tại hay không tồn tại, sẵn sàng hay không sẵn sàng.
    - Nếu dữ liệu đang tồn tại thì mặt động của nó còn thể hiện ở giá trị cụ thể của dữ liệu tại từng thời điểm.
    - Trạng thái hay giá trị của dữ liệu sẽ bị thay đổi khi xuất hiện những sự kiện, thao tác tác động lên nó.

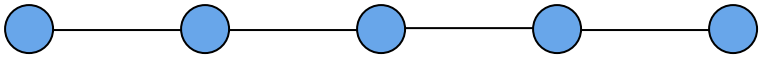
Cấu trúc dữ liệu

- Cấu trúc dữ liệu (data structure) :
  - Là kiểu dữ liệu có chứa nhiều thành phần dữ liệu
  - Dùng để biểu diễn cho các thông tin có cấu trúc của bài toán.
  - Cấu trúc dữ liệu thể hiện khía cạnh logic của dữ liệu.
- Các dữ liệu không có cấu trúc được gọi là các dữ liệu vô hướng hay các dữ liệu đơn giản.
  - VD: các kiểu dữ liệu số nguyên (integer), số thực (real), logic (boolean) là các kiểu dữ liệu đơn giản.

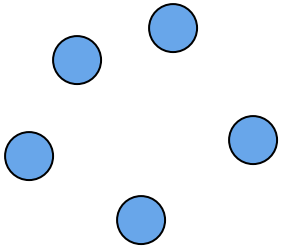
Cấu trúc dữ liệu

- Có hai loại cấu trúc dữ liệu chính:
  - Cấu trúc tuyến tính:
    - Các phần tử bên trong nó luôn được bố trí theo một trật tự tuyến tính hay trật tự trước sau.
    - Đây là loại cấu trúc dữ liệu đơn giản nhất. Ví dụ: mảng, danh sách.
  - Cấu trúc phi tuyến:
    - Các thành phần bên trong không được bố trí theo trật tự tuyến tính mà theo các cấu trúc khác.
    - Ví dụ: tập hợp (không có trật tự), cấu trúc cây (cấu trúc phân cấp), đồ thị (cấu trúc đa hướng).

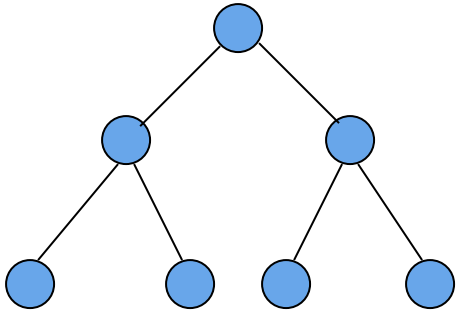
Hình minh họa: các loại CTDL



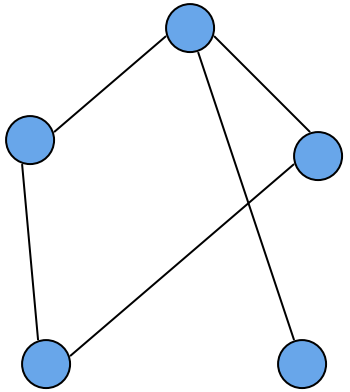
Danh sách



Tập hợp



Cây



Đồ thị

Cấu trúc lưu trữ (storage structure)

- Cấu trúc lưu trữ của một cấu trúc dữ liệu:
  - Thể hiện khía cạnh vật lý (cài đặt) của cấu trúc dữ liệu đó.
  - Là một trong số các cách tổ chức lưu trữ của máy tính

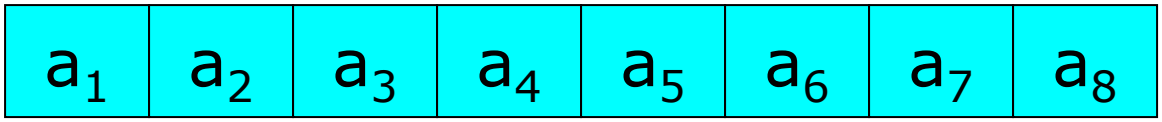
Cấu trúc lưu trữ

- Cấu trúc lưu trữ trong:
  - nằm ở bộ nhớ trong (bộ nhớ chính) của máy tính.
  - tương đối đơn giản, dễ tổ chức và tốc độ thao tác rất nhanh.
  - nhược điểm: không có tính lưu tồn (persistence), và kích thước khá hạn chế.
- Cấu trúc lưu trữ ngoài:
  - Nằm ở bộ nhớ ngoài (bộ nhớ phụ).
  - Thường có cấu trúc phức tạp và tốc độ thao tác chậm hơn rất nhiều so với CTLT trong,
  - CTLT này có tính lưu tồn và cho phép lưu trữ các dữ liệu có kích thước rất lớn.

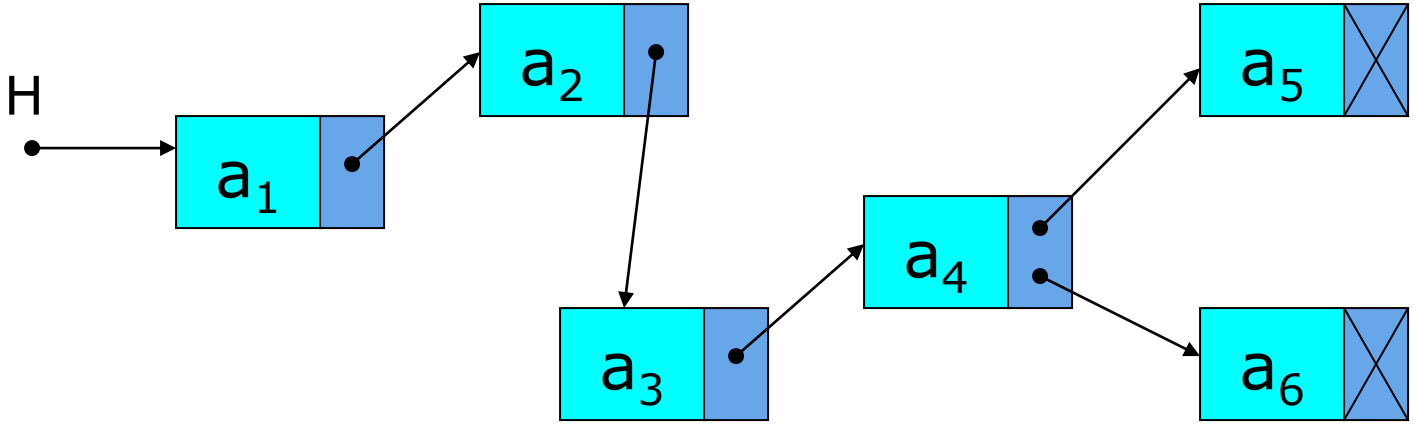
Cấu trúc lưu trữ trong

- Gồm hai loại:
  - Cấu trúc lưu trữ tĩnh:
    - Kích thước dữ liệu luôn cố định.
    - Còn được gọi là CTLT tuần tự.
  - Cấu trúc lưu trữ động:
    - Kích thước dữ liệu có thể thay đổi trong khi chạy chương trình.
    - Còn được gọi là cấu trúc con trỏ hay móc nối.

Hình minh họa: các loại CTLT trong



Cấu trúc tĩnh



Cấu trúc động

Một số đặc điểm của các CTLT trong

- CTLT tĩnh:
  - Các ngăn nhớ đứng liền kề nhau thành một dãy liên tục trong bộ nhớ
  - Số lượng và kích thước mỗi ngăn là cố định
  - Có thể truy nhập trực tiếp vào từng ngăn nhờ chỉ số, nên tốc độ truy nhập vào các ngăn là đồng đều
- CTLT động:
  - Chiếm các ngăn nhớ thường không liên tục
  - Số lượng và kích thước các ngăn có thể thay đổi
  - Việc truy nhập trực tiếp vào từng ngăn rất hạn chế, mà thường sử dụng cách truy nhập tuần tự, bắt đầu từ một phần từ đầu, rồi truy nhập lần lượt qua các con trỏ móc nối (liên kết)

Ngôn ngữ diễn đạt giải thuật

- Nguyên tắc khi sử dụng ngôn ngữ:
  - Tính độc lập của giải thuật : ngôn ngữ được chọn phải làm sáng tỏ tinh thần của giải thuật, giúp người đọc dễ dàng hiểu được logic của giải thuật.

    → Các ngôn ngữ thích hợp là ngôn ngữ tự nhiên và ngôn ngữ hình thức (như các lưu đồ thuật toán, các ký hiệu toán học).
  - Tính có thể cài đặt được của giải thuật : ngôn ngữ được chọn phải thể hiện được khả năng có thể lập trình được của giải thuật, và giúp người đọc dễ dàng chuyển từ mô tả giải thuật thành chương trình

    → Các ngôn ngữ lập trình là công cụ tốt nhất vì nó cho ta thấy rõ cài đặt của giải thuật và hoạt động của giải thuật khi chúng ta chạy chương trình trên máy tính

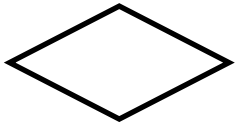
Các loại ngôn ngữ diễn đạt giải thuật

- Ngôn ngữ tự nhiên
- Lưu đồ giải thuật:
  - Sử dụng các hình vẽ, biểu tượng để biểu diễn cho các thao tác của giải thuật
- Ngôn ngữ lập trình: C/C++, java…

Các thành phần cơ bản của lưu đồ giải thuật

Chỉ đến khối lệnh tiếp theo

Khối lệnh (có thể lệnh đơn hay lệnh phức)



Lệnh rẽ nhánh (điều kiện rẽ nhánh)

Điểm bắt đầu giải thuật

Điểm kết thúc giải thuật

Thiết kế và đánh giá giải thuật

- Thiết kế giải thuật:
  - Thiết kế cấu trúc chương trình mà cài đặt giải thuật.
  - Tìm cách biến đổi từ đặc tả giải thuật (mô tả giải thuật làm cái gì, các bước thực hiện những gì) thành một chương trình được viết bằng một ngôn ngữ lập trình cụ thể (giải thuật được cài đặt như thế nào) mà có thể chạy tốt trên máy tính (minh hoạ hoạt động cụ thể của giải thuật).

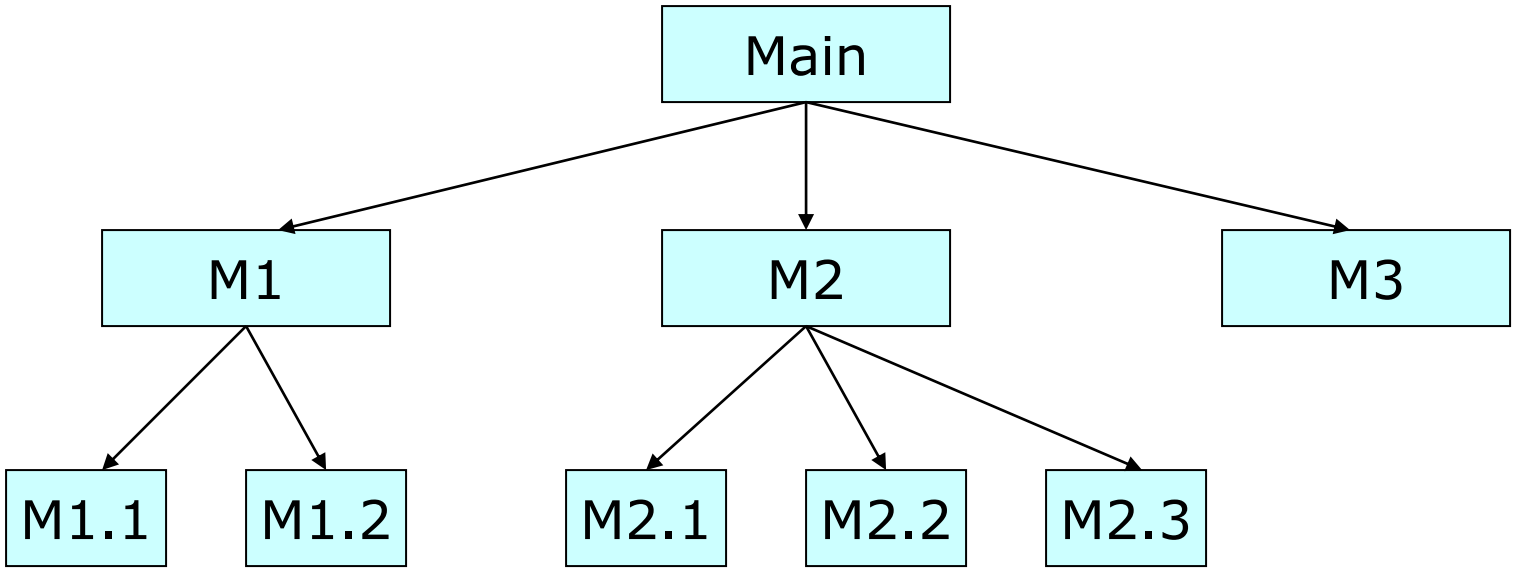
Các giai đoạn thiết kế chính

- Thiết kế sơ bộ:
  - cần tìm hiểu cặn kẽ các thành phần của giải thuật: giải thuật gồm có bao nhiêu thành phần cơ bản, mỗi thành phần đó làm gì, giữa các thành phần đó có mối liên quan gì.
  - Mỗi thành phần cơ bản được goi là một modulcủa giải thuật.
  - Phương pháp thiết kế: phương pháp thiết kế từ trên xuống
- Thiết kế chi tiết:
  - bắt đầu cài đặt các modul bằng một ngôn ngữ lập trình cụ thể.
  - tiến hành ghép nối các modul để tạo thành một chương trình hoàn chỉnh thực hiện giải thuật ban đầu.
  - Phương pháp thiết kế: phương pháp tinh chỉnh từng bước

Phương pháp TK từ trên xuống

- Còn được gọi khác là phương pháp modul hoá, dựa trên nguyên tắc chia để trị.
- Chia giải thuật ban đầu thành các giải thuật con (modul), mỗi giải thuật con sẽ thực hiện một phần chức năng của giải thuật ban đầu.
- Quá trình phân chia này được lặp lại cho các modul con cho đến khi các modul là đủ nhỏ để có thể giải trực tiếp
- Kết quả phân chia này sẽ tạo ra một sơ đồ phân cấp chức năng

Sơ đồ phân cấp chức năng



Phương pháp tinh chỉnh từng bước

- Chứa các quy tắc cho phép thực hiện việc chuyển đổi từ đặc tả giải thuật bằng ngôn ngữ tự nhiên hay lưu đồ sang một đặc tả giải thuật bằng một ngôn ngữ lập trình cụ thể.
- Quá trình chuyển đổi này gồm nhiều bước, trong đó mỗi bước là một đặc tả giải thuật.
  - Bước đầu tiên, đặc tả giải thuật bằng ngôn ngữ tự nhiên hay lưu đồ giải thuật.
  - Các bước sau, thay thế dần dần các thành phần được biểu diễn bằng ngôn ngữ tự nhiên của giải thuật bằng các thành phần tương tự được biểu diễn bằng ngôn ngữ lập trình đã chọn.
  - Lặp lại quá trình trên cho đến khi tạo ra một chương trình hoàn chỉnh có thể chạy được, thực hiện giải thuật yêu cầu

Quy tắc diễn đạt giả lệnh

- Tên CT: Viết bằng chữ in hoa
  - Ví dụ: Program NHAN_MA_TRAN
  - {}, #, // : viết chú thích
- Ký tự :
  - 26 chữ cái la tinh in hoa hoặc thường
  - 10 chữ số thập phân
  - Các phép toán số học: +,-,*,/…
  - Các phép toán quan hệ: <,>,=…
  - Giá trị logic: true, false
  - Dấu phép toán logic : and, or, not
- Tên biến : dãy chữ cái hoặc chữ số

Các câu lệnh

- Câu lệnh điều kiện: if, if else
- Câu lệnh lặp: for, while, do while
- Câu lệnh vào /ra : input/output
- Câu lệnh bắt đầu/kết thúc chương trình : {}, begin/end
- Câu lệnh trả về giá trị: return

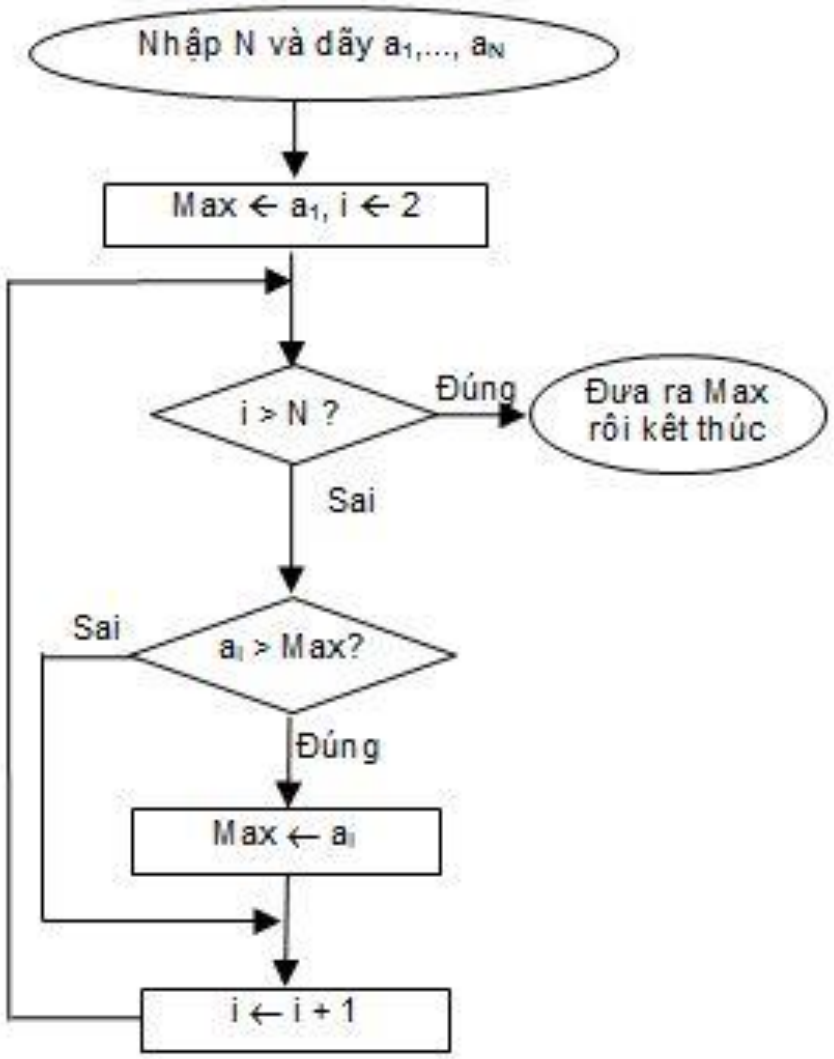
Chương trình con

- Chương trình con cho hàm: Function
- Chương trình con cho thủ tục: Procedure
- Lời gọi chương trình con
  - Call <Tên_thủ_tục>

Ví dụ 1

- Tìm số lớn nhất trong dãy số nguyên.

```
Program TIMMAX
Input: s
Output: x
ArrayMax(s){
   x=s[0]
   for(i=1; i<s.length; i++){
   if (s[i] > x ) x = s[i]; }
   return x
}
```



Ví dụ 2

- Sắp xếp một dãy số a1, a2, a3 … an thành một dãy số tăng dần
- Xác định rõ dữ liệu và yêu cầu: cho biết cái gì (input), đòi hỏi cái gì (output)
  - Input: 33, 77, 11, 55, 99, 22, 44, 88, 66
  - Output: 11, 22, 33, 44, 55, 66, 77, 88, 99
- Để có được kết quả thì phải làm gì:
  - Số bé nhất trong n số đặt vị trí đầu tiên
  - Số bé nhất trong n-1 số còn lại đặt ở vị trí thứ 2 …
- Thực hiện các công việc trên bằng cách nào?

Giải thuật sắp xếp

Procedure SELECTION_SORT(A,n)

//A là vector gồm n phần tử

1. for i:=1 to (n-1) do begin
2. Chọn số nhỏ nhất A[k] trong dãy các số A[i]
3. Hoán vị giữa A[k] và A[i]
4. Return end;

Thiết kế sơ bộ giải thuật

Giải thuật sắp xếp

```
Procedure SELECTION_SORT(A,n)
1. for i:=1 to (n-1) do
2. k:= i;
3. for j:= i+1 to n do
4. if A[j] < A[k] then k:=j;
5. temp: = A[k]; A[k] = A[j]; A[j] = temp;
6. return            end;
```

Tinh chỉnh từng bước giải thuật

Code C

```
//Sắp xếp đoạn mảng A gồm n phần tử
void SelectionSort(int A[], int n)
{
    for (int i = 1; i < n; i++)
    {
        int m = i;
        for (int j = i + 1; j <= n; j++)
            if (A[j] < A[m]) m = j;
            if (i != m) swap(A[i], A[m]);
    }
}
```

Đánh giá giải thuật

- Dựa trên hai yếu tố
  - Không gian nhớ cho cấu trúc lưu trữ
  - Thời gian thực hiện
- Thế nào là giải thuật tốt, tốt nhất, không tốt?

Đánh giá theo thời gian thực thi

- Các yếu tố ảnh hưởng đến thời gian thực thi
  - Cấu trúc máy tính
  - Hệ điều hành
  - Ngôn ngữ lập trình
  - Kích thước dữ liệu và số lệnh thực hiện.

Thời gian thực hiện giải thuật

- Thời gian thực hiện giải thuật của hàm số của kích thước dữ liệu n là: T(n)
  - n : là kích thước của bộ dữ liệu. Việc xác định n tùy thuộc vào bài toán cụ thể.
- Ví dụ : sắp xếp một dãy n số, thì kích thước dữ liệu là n.
- Làm thế nào để xác định T(n)?

@ Ví dụ 3

- Giải thuật tính giá trị trung bình của n số

```
Program TB
1.  Read (n); //đọc vào n
2.  S = 0;
3.  i = 1;
4.  While i <= n do {
5.  Read (X); // đọc số thứ i
6.  S  = S + X;
7.   i = i +1 ; }
8.  M = S/n;
9.   Write (M);
10. Return;
```

- Các lệnh 1, 2, 3, 8,9 thực hiện 1 lần
- Các lệnh 5, 6, 7 thực hiện n lần
- Lệnh 4 thực hiện n+1 lần
- Tổng số lần thực hiện lệnh là 4n + 5
- T(n) = 4n + 6
- T(n) tăng tuyến tính theo n
- T(n) có độ lớn bậc n

Độ phức tạp thuật toán

- Thời gian T(n) của một giải thuật được gọi là có độ lớn bậc f(n) ký hiệu bởi: T(n) = O(f(n))

Nếu tồn tại các số dương C và n_0 thỏa mãn: $T(n) <= Cf(n)$; $n >= n_0$

- Độ phức tạp về thời gian của giải thuật này là O(f(n)): Ký pháp chữ O lớn – "Big O" Notation
- f(n) : là hàm đơn giản biểu diễn độ phức tạp của một giải thuật

Tính chất của O

- Tính chất 1: nếu T(n) = O(f(n)) và f(n) = O(g(n)) thì T(n) = O(g(n)).
- Tính chất 2: nếu T(n) = T1(n) + T2(n) và T1(n) = O(f1(n)), T2(n) = O(f2(n)) thì T(n) = max(f1(n), f2(n)).
- Tính chất 3: nếu T(n) = T1(n).T2(n) và T1(n) = O(f1(n)), T2(n) = O(f2(n)) thì T(n) = f1(n). f2(n).

Tính chất của O

- Các hệ quả:
  - Hệ quả 1: Độ phức tạp của một lệnh rẽ nhánh bằng độ phức tạp của nhánh có độ phức tạp cao nhất.
  - Hệ quả 2: Độ phức tạp của một lệnh tuần tự bằng tổng độ phức tạp của các lệnh thành phần.
  - Hệ quả 3: Độ phức tạp của một lệnh lặp bằng tổng độ phức tạp của tất cả các lần lặp (trong trường hợp độ phức tạp của mỗi lần lặp là như nhau thì giá trị này sẽ bằng số lần lặp nhân với độ phức tạp của một vòng lặp).

Độ phức tạp thuật toán

- Giải thuật tính giá trị trung bình của n số

Program TB

1. Read (n); //đọc n giá trị khác nhau
2. S = 0;
3. i = 1;
4. While i <= n do {//begin
5. Read (X); // đọc số thứ i
6. S = S + X;
7. i = i +1 ;} //end
8. M = S/n;
9. Write (M);
10. Return;

- T(n) <= Cf(n) ; n >= n0
- Ta có :

T(n) = 4n + 5 <= 5n; n >= 5

Chọn C = 5, n_0 = 5 -> f(n) = n

Độ phức tạp là O(n)

T(n) = O(f(n)) = O(n)

Độ phức tạp thuật toán

```
Procedure SELECTION_SORT(A,n);
1. for i:=1 to (n-1) do
2. k:= i;
3. for j:= i+1 to n do
4. if A[j] < A[k] then k:=j;
5. temp: = A[k]; A[k] = A[i]; A[j] = temp;
6. return                    end;
```

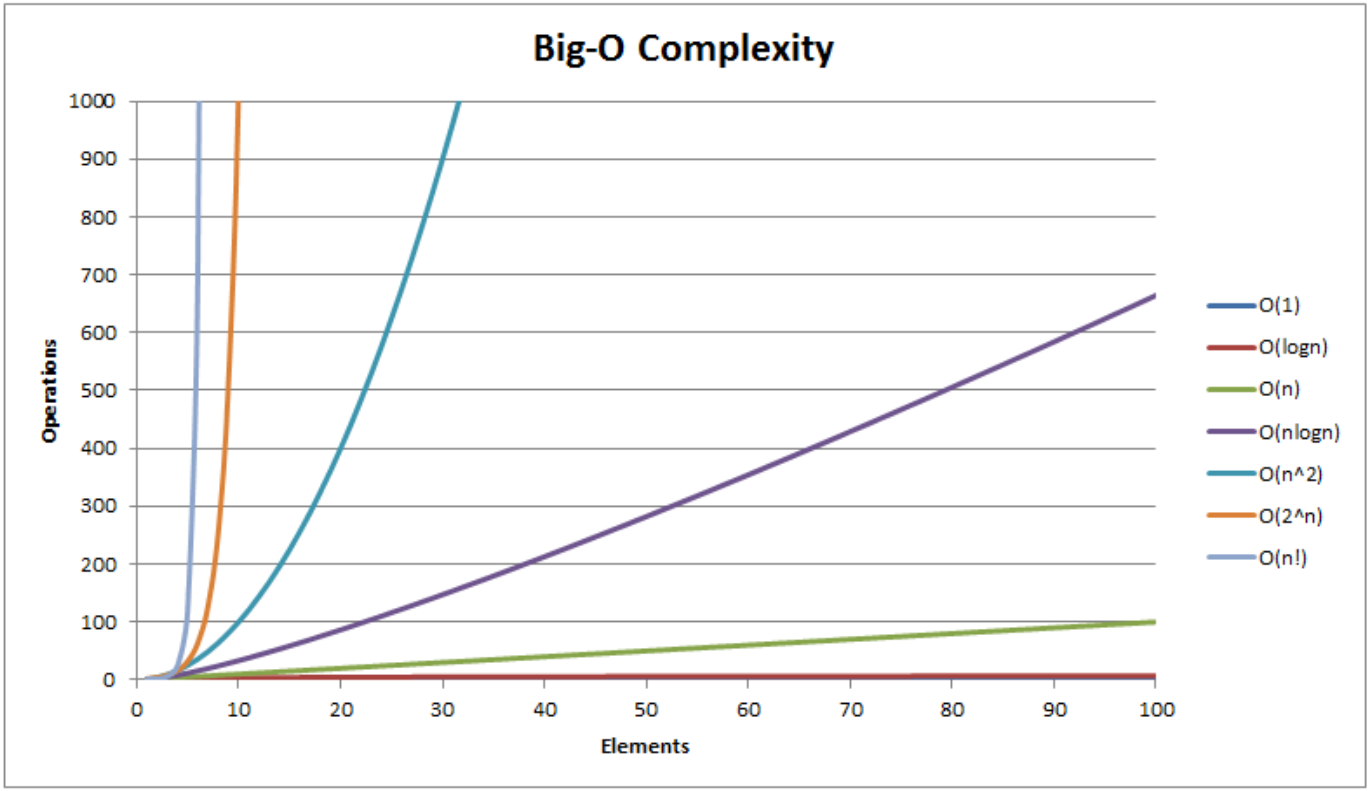
- i = 1 bước 4 thực hiện n-1 lần
- i = 2 bước 4 thực hiện n-2 lần
- ….
- i = n-1 bước 4 thực hiện 1 lần
- Tổng số lần thực hiện là:

1+2+…+(n-1)=n*(n-1)/2= ½*n^2–½*n

- T(n) = O(n^2)

Các loại độ phức tạp thường gặp

- Độ phức tạp hằng số: O(1)
- Độ phức tạp tuyến tính: O(n)
- Độ phức tạp đa thức: O(nk)
- Độ phức tạp logarit: O(log n)
- Độ phức tạp nlogn: O(nlog n)
- Độ phức tạp lũy thừa: O(an)
- Độ phức tạp giai thừa: O(n!)



Bài tập

Bài 1: Biểu diễn độ phức tạp thuật toán theo Big 0 tốt nhất theo các hàm thời gian sau đây

1. $T(n) = n^3 + 100n\log_2 n + 500;$
2. $T(n) = 2^n + n^{99} + 7$
3. $T(n) = n*(3+n) - 7n;$
4. $T(n) = ((n+1)*\log_2(n+1)-(n+1)+1)/n$
5. $T(n) = (n+1)*\log_2(n+1)-(n+1)+1/n$

Bài tập

Bài 2: Với mỗi đoạn giải thuật dưới đây, hãy dùng Big_O để biểu diễn thời gian thực hiện

1.
```
For i = 1 to n do
        For j = 1 to n do
                A[i,j] = B[i,j] + C[i,j];
```

2.
```
S = 0;
For i = 1 to n do
        Read (X)
        S = S + X;
```

3.
```
For i = 1 to n do
        For j  = 1 to n do
                C[i,j] = 0;
                For k = 1 to n do
                C[i,j] = C[i,j] + A[i,k]*B[k,j]
```

4.
```
j = n
        Repeat
        j = j/2;
        Until j <=1;
```