Phân tích và Thiết kế THUẬT TOÁN

Hà Đại Dương
duonghd@mta.edu.vn
Web: fit.mta.edu.vn/~duonghd

Bài 3 - Thiết kế thuật toán và Phương pháp trực tiếp

PHÂN TÍCH VÀ THIẾT KẾ THUẬ TOÁN

NỘI DUNG

I. Giới thiệu
II. Thiết kế thuật toán
   1. Modul hóa và phân tích từ trên xuống (top-down)
   2. Một số phương pháp thiết kế
   3. Tối ưu thuật toán
III. Phương pháp trực tiếp
   1. Lược đồ chung
   2. Một số bài toán áp dụng
IV. Bài tập

I. Giới thiệu

- Thiết kế thuật toán là vấn đề mang tính:
  - Kỹ thuật
  - Nghệ thuật
- Đòi hỏi người thực hiện phải có:
  - Kiến thức
  - Kinh nghiệm
  - Kỹ năng
- Thuật toán được thiết kế phải:
  - Đúng, đơn giản, dễ dùng
  - Phù hợp với bộ nhớ của máy tính và có thời gian thực hiện hợp lý

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Các bài toán cần giải quyết trên máy tính ngày càng đa dạng, phức tạp
- Các thuật toán đòi hỏi có quy mô lớn, tốn nhiều thời gian và công sức

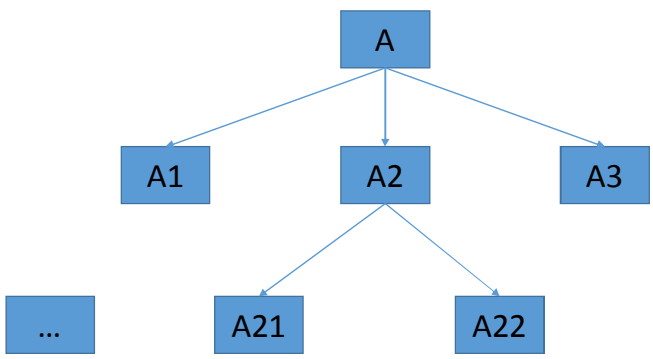
Bài toán cần giải quyết: A

Chia nhỏ bài toán thành các bài toán nhỏ: Bài toán cần giải quyết là modul chính ->Chia bài toán thành các modul nhỏ hơn

Đây là cách tiếp cận thông thường của con người với hầu hết các vấn đề đặt ra của cuộc sống.

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN



II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Phương pháp phân tích top-down để giải một bài toán:
  - Là quá trình phân tích bài toán được thực hiện từ trên xuống dưới;
  - Từ mức tổng quát là các ý tưởng giải quyết, các bước để giải quyết bài toán, cho đến mức chi tiết là các câu lệnh trong chương trình.
  - Quá trình phân rã bài toán được thực hiện theo từng mức khác nhau.

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Phương pháp phân tích top-down để giải một bài toán:
  - Mức có chỉ số thấp nhất (đầu tiên) được gọi là mức tổng quan. Ở mức tổng quan có thể xem xét tổng thể lời giải bài toán thông qua các nhiệm vụ chính.
  - Mức tiếp theo là mức các nhiệm vụ chính để thực hiện lời giải bài toán. Công việc chủ yếu ở mức này là mô tả cụ thể từng nhiệm vụ chính.
  - Quá trình phân tích tiếp tục phân rã lời giải bài toán thành từng nhiệm vụ cho tới khi nhận được các đơn thể (hàm hoặc thủ tục), trong đó mọi công việc được hình dung khá rõ ràng và xác định.

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự

  Cho xâu ký tự S. Hãy sửa xâu S sao cho:
  - Giữa hai âm tiết có đúng một dấu cách;
  - Sau các dấu đặc biệt như dấu chấm phảy ";", dấu phảy "’,", dấu chấm ".", dấu hai chấm “:” có đúng một kí tự trắng;
  - Trước các dấu đặc biệt không có kí tự trắng và
  - Đầu câu phải viết hoa.
  - Ví dụ, cho *S* = " học tập , phấn đấu .rèn luyện . ", cần sửa thành "Học tập, phấn đấu. Rèn luyện. "

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức tổng quan:** Hình dung toàn bộ những thao tác (nhiệm vụ chính) phải thực hiện trên *S*. Có nhiều cách phân chia bài toán, ta xét một trong nhiều cách phân chia nhiệm vụ như sau:
    1. Xóa dấu trống ở đầu và cuối. Ví dụ, S = "học tập , phấn đấu .rèn luyện . ".
    2. Xóa hết các kí tự trắng đứng liên tiếp: nghĩa là không để hơn một kí tự trắng đứng cạnh nhau. Ví dụ S = "học tập , phấn đấu .rèn luyện . ".

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức tổng quan:** Hình dung toàn bộ những thao tác (nhiệm vụ chính) phải thực hiện trên *S*. Có nhiều cách phân chia bài toán, ta xét một trong nhiều cách phân chia nhiệm vụ như sau:
    3. Xóa hết kí tự trắng (nếu có) đứng trước các dấu ": ; , .". Ví dụ S = "học tập, phấn đấu.rèn luyện. ".
    4. Thêm kí tự trắng (nếu cần) vào sau các dấu ": ; , .". Ví dụ S = "học tập, phấn đấu. rèn luyện. ".
    5. Sửa (nếu cần) kí tự đầu câu thành viết hoa. Ví dụ S = "Học tập, phấn đấu. Rèn luyện. ".

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 1:** Mô tả chi tiết các nhiệm vụ chính

1) Xóa kí tự trắng ở đầu và cuối xâu kí tự S

```
void Xoa_dau(string &s)
  {  while (s[0] là kí tự trắng)
     s[0..strlen(s)] ← s[1..strlen(s)];
}
void Xoa_cuoi(string &s)
  {  while (kí tự cuối cùng là trắng)
     kí tự cuối cùng ←kí tự rỗng;
}
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 1:**

2) Xóa kí tự trắng thừa ở bên trong

```
void Xoa_trong(string &s)
{    i = 1;
     while (i<strlen(s))
     {
          while (s[i]  và s[i+1] đều là kí tự trắng)
               s[i..strlen(s)]←s[i+1..strlen(s)];
               i++;
     }
}
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 1:**

3) Xóa kí tự trắng (nếu có) đứng trước các dấu đặc biệt

```
void Xoa_truoc_dau_DB(string &s)
{
     for (i=1; i<strlen(s); i++)
          if  (s[i] thuộc ". , ; :" và s[i-1] là kí tự trắng)
               s[i-1..strlen(s)] ← s[i..strlen(s)];
}
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 1:**

4) Thêm kí tự trăng (nếu cần) vào sau các dâu đặc biệt

```
void Them_sau_dau_DB(string &s)
{   string b;
    for (i=1;i<strlen(s)-1;i++)
        if (s[i] thuộc ".,;:" và s[i+1] không phải kí tự trắng)
        {
            Chép đoạn s[i+1..strlen(s)] vào b;
            Đặt s[i+1] là kí tự trắng;
            Đặt b vào s từ s[i+2];
        }
}
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 1:**

5) Viết hoa đầu câu

```
void Viet_hoa_dau_cau(string &s)
{   Viết hoa từ đầu tiên;
    Xét kí tự i =1 đến kí tự thứ strlen(s)-1:
        Nếu  s[i] là dấu chấm thì
            viết hoa kí tự s[i+2];
}
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 2:** Mô tả chi tiết các nhiệm vụ chính

1) Xóa kí tự trắng ở đầu và cuối xâu kí tự S

```
void Xoa_dau(string &s)
 {   while (s[0]= =32)
          strcpy(&s[0],&s[1]);
}
void Xoa_cuoi(string &s)
 {   while (s[strlen(s)-1]= =' ')
          strcpy(&s[strlen(s)-1],"");
}
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 2:** Mô tả chi tiết các nhiệm vụ chính

2) Xóa kí tự trắng thừa ở bên trong

```
void Xoa_trong(string &s)
{    i = 1;
     while (i<strlen(s))
     {
          while ((s[i]==32)&&(s[i+1]==32))
               strcpy(&s[i],&s[i+1]);
          i++;
     }
}
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 2:** Mô tả chi tiết các nhiệm vụ chính

3) Xóa kí tự trắng (nếu có) đứng trước các dấu đặc biệt

```
void Xoa_truoc_dau_DB(string &s)
 {
   for  (i=1;i<strlen(s);i++)
        if ((In_Set(s[i],".,;:")) &&(s[i-1] = =32))
                strcpy(&s[i-1],&s[i]);
 }
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 2:** Mô tả chi tiết các nhiệm vụ chính

4) Thêm kí tự trắng (nếu cần) vào sau các dấu đặc biệt

```
void Them_sau_dau_DB(string &s)
{   string b;
    for (i=1;i<strlen(s)-1;i++)
        if ((In_Set(s[i],".,;:"))&&(s[i+1]!=32))
        {
            strcpy(&b[0],&s[i+1]);
            s[i+1]=32;
            strcpy(&s[i+2],&b[0]);
        }
}
```

II. Thiết kế thuật toán

1. MODUL HÓA VÀ PHÂN TÍCH TOP-DOWN

- Ví dụ - Bài toán: Chuẩn hóa xâu ký tự
  - **Mức chi tiết 2:** Mô tả chi tiết các nhiệm vụ chính

5) Viết hoa đầu câu

```
void Viet_hoa_dau_cau(string &s)
{    s[0]=toupper(s[0]);
     for (i=1;i<strlen(s)-1;i++)
     if  (s[i]=='.')
          s[i+2]=toupper(s[i+2]);
}
```

II. Thiết kế thuật toán

2. MỘT SỐ PHƯƠNG PHÁP THIẾT KẾ

- Phương pháp trực tiếp (Straight method)
  - Từ bài toán đã cho -> Phát hiện những đặc trưng của nó;
  - Xác định mối liên hệ giữa dữ liệu vào và yêu cầu đầu ra;
  - Sử dụng các công cụ biểu diễn để mô tả thuật toán từ đơn giản đến phức tạp.

- Chia để trị (Divide and Conquer)
  - Để giải quyết bài toán có thể phân chia thành bài toán có kích thước nhỏ hơn mà việc tìm lời giải được thực hiện với cùng 1 cách.
  - Kết hợp lài giải các bài toán con thành kết quả bài toán cần giải quyết
  - Ví dụ: tìm kiếm nhị phân, sắp xếp hòa nhập.

II. Thiết kế thuật toán

2. MỘT SỐ PHƯƠNG PHÁP THIẾT KẾ

- Phương pháp tham lam (Greedy Method)
  - Giải các bài toán tối ưu rời rạc
  - Dựa trên nguyên tắc: lời giải tối ưu của bài toán đạt được nhờ việc chọn tối ưu trong từng bước ở mỗi bài toán con.
  - Ví dụ: tối ưu số tờ tiền phải trả của máy ATM, bài toán người du lịch …
- Phương pháp quy hoạch động (Dynamic Programming)
  - Giải bài toán tối ưu.
  - Một dạng của cách tiếp cận chia để trị.
  - Dựa trên nguyên lý Bellman: nếu lời bài toán là tối ưu thì lời giải bài toán con là tối ưu.
  - Ví dụ: Bài toán cái túi …

II. Thiết kế thuật toán

2. MỘT SỐ PHƯƠNG PHÁP THIẾT KẾ

- Phương pháp quay lui (Back Tracking)
  - Thường dùng cho các bài toán không có thuật giải trực tiếp
  - Tư tưởng chính là chiến lược Thử-và-sai
  - Bài toán: Bài toán liệt kê hoán vị, 8-hậu…
- Phương pháp nhánh cận (Branch and Bound)
  - Dùng cho các bài toán không có thuật giải trực tiếp
  - Khi kích thước lớn pp quay lui có thể dẫn đến không đáp ứng thời gian
  - Bài toán: Cái túi, hoán vị …

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Việc xem xét thuật toán, chỉnh sửa để thuật toán hiệu quả hơn là cần thiết.
- Việc tối ưu thuật toán thường dựa trên "Điểm mất thời gian nhất của thuật toán".
- Tối ưu vòng lặp
- Tối ưu điều khiển chọn

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:
  - Là điểm cần quan tâm đầu tiên
  - Cố gắng giảm các vòng lặp lồng nhau
  - Tăng số lệnh thực hiện trong một vòng lặp để giảm số lượng các bước lặp
  - Tách các lệnh không phụ thuộc vào chỉ số lặp ra khỏi vòng lặp

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:
  - Ví dụ 1: tính e^x

Xét thuật toán tính giá trị của e^x theo công thức gần đúng :

$$\sum_{i\geq 0}\frac{x^i}{i!} = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots$$

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:
  - Ví dụ 1: tính e^x

Thuật toán 1 :{ Tính từng số hạng rồi cộng lại.}

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

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

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:
  - Ví dụ 1: tính e^x

T(n) = ???

Thuật toán 2 :

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

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:
  - Ví dụ 2: C = A.B

$T(n) = O(n^3)$

Thuật toán 1 :

```
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] += a[i][k]*b[k][j];
   }
```

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:
  - Ví dụ 2: C = A.B

Thuật toán 2 :

trong trường hợp n chẵn
cải tiến tính một lần 4 giá trị
c[i][j]
c[i+1][j]
c[i][j+1]
c[i+1][j+1]

II. Thiết kế thuật toán

Thuật toán 2 :

```
i = 1;
while (i < n)
{  ii = i+1;
   j = 1;
   while ( j < n )
   {   jj = j+1;
       c[i][j] = 0;
       c[i][jj] = 0;
       c[ii][j] = 0;
       c[ii][jj] = 0;
       k = 1;
       while (k < n)
       {     kk = k+1;
             c[i][j] = c[i][j] + a[i][k]*b[k][j] + a[i][kk]*b[kk][j];
             c[i][jj] = c[i][jj]+a[i][k]*b[k][jj] + a[i][kk]*b[kk][jj];
             c[ii][j] = c[ii][j]+a[ii][k]*b[k][j] + a[ii][kk]*b[kk][j];
             c[ii][jj] = c[ii][jj]+a[ii][k]*b[k][jj] +  a[ii][kk]*b[kk][jj];
             k = k+2;
       }
       j = j + 2;
   }
   i = i + 2;
}
```

Độ phức tạp : T(n) ???

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:
  - Ví dụ 3: Thuật toán sắp xếp chọn

Thể hiện 1

```
for (int i=0;i<n-1;i++)
{
        for (int j=i+1;j<n;j++)
        {
                if (a[i]>a[j])
                {
                        int tg=a[i];
                        a[i]=a[j];
                        a[j]=tg;
                }
        }
}
```

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:

Thể hiện 2

```
int i, j, tg;
for (i=0;i<n-1;i++)
{
        for (j=i+1;j<n;j++)
        {
                if (a[i]>a[j])
                {
                        tg=a[i];
                        a[i]=a[j];
                        a[j]=tg;
                }
        }
}
```

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu vòng lặp:

Thể hiện 3

Độ phức tạp
- Thể hiện 1?
- Thể hiện 2?
- Thể hiện 3?

```
int i, j, tg, imax;
for (i=0;i<n-1;i++)
{
        imax=i;
        for (j=i+1;j<n;j++)
        {
                if (a[imax]>a[j])
                {
                        imax=j;
                }
        }
        if (imax!=i)
        {
                tg=a[imax];
                a[imax]=a[i];
                a[i]=tg;
        }
}
```

II. Thiết kế thuật toán

3. TỐI ƯU THUẬT TOÁN

- Tối ưu rẽ nhánh
  - Dùng cấu trúc switch khi lựa chọn 1 trong nhiều khả năng
  - Dùng cấu trúc if … then … else
  - Với biểu thức logic kết hợp bằng phép &&

    A_1 && A_2 &&... && A_n

    viết theo thứ tự sai nhiều lên trước
  - Với biểu thức logic kết hợp bằng ||

    A_1 || A_2 ||... || A_n

    viết theo thứ tự đúng nhiều lên trước

III. TKTT Phương pháp trực tiếp

1. LƯỢC ĐỒ CHUNG

- Từ bài toán đã cho xem xét những tính chất đặc trưng của nó => Xác định mối liên hệ giữa đầu vào và đầu ra.

1. Xác định đầu vào, đầu ra (những gì? Như thế nào?)
2. Xác định mối liên hệ đầu vào, đầu ra: Thường là các công thức, các khái niệm hoặc quy luật
3. Biểu diễn thuật toán: từng bước chi tiết cho đến chương trình trên ngôn ngữ lập trình.

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 1** - Cho số nguyên *a* có không quá 100 chữ số và số nguyên *b*, $1 \leq b \leq 9$. Tính $c = a \times b$.
- **Minh họa**

b = 6 với a = 6789

Bước 1:

b =				**b =**	**6**
a =		**6**	**7**	**8**	**9**
c =					**4**
nhớ				**5**	**0**

6× 9 = 54 + 0 = 54

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 1** - Cho số nguyên *a* có không quá 100 chữ số và số nguyên *b*, $1 \le b \le 9$. Tính $c = a \times b$.
- **Minh họa**

b = 6 với a = 6789

Bước 2:

				b =	6
a =		6	7	8	9
c =				3	4
nhớ			5	5	0

6× 8 = 48 + 5 = 53

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 1** - Cho số nguyên *a* có không quá 100 chữ số và số nguyên *b*, $1 \le b \le 9$. Tính $c = a \times b$.
- **Minh họa**

b = 6 với a = 6789

Bước 2:

				b =	6
a =		6	7	8	9
c =			7	3	4
nhớ		4	5	5	0

6× 7 = 42 + 5 = 47

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 1** - Cho số nguyên *a* có không quá 100 chữ số và số nguyên *b*, $1 \le b \le 9$. Tính $c = a \times b$.
- **Minh họa**

b = 6 với a = 6789

Bước 4:

				b =	6
a =		6	7	8	9
c =		0	7	3	4
nhớ	4	4	5	5	0

6× 6 = 36 + 4 = 40

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 1** - Cho số nguyên *a* có không quá 100 chữ số và số nguyên *b*, $1 \le b \le 9$. Tính $c = a \times b$.
- **Minh họa**

b = 6 với a = 6789

Bước 5:

				b =	6
a =		6	7	8	9
c =	4	0	7	3	4
nhớ	4	4	5	5	0

4 + 0 = 4

Kết quả:

c = 40734

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 1** - Cho số nguyên *a* có không quá 100 chữ số và số nguyên *b*, $1 \le b \le 9$. Tính $c = a \times b$.

Thuật toán:

input: a = (a[n], ..., a[1]) ∈ N, n ≤ 100; b ∈{0, 1, ..., 9}

output: c = a × b

1) nhớ = 0;
2) Nhân tuần tự từ a[1] đến a[n], mỗi lần cộng với nhớ cho kết quả tg. Mỗi lần nhân, phần dư (tg mod 10) đưa vào c, thương tg/10 đưa vào nhớ cho lần sau.
3) Cuối cùng, nếu nhớ > 0 thêm nhớ vào chữ số thứ n+1 cho c
4) output c = (c[n+1] c[n], ...c[1]);

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 1** - Cho số nguyên *a* có không quá 100 chữ số và số nguyên *b*, $1 \le b \le 9$. Tính $c = a \times b$.

Thuật toán:

input: a = (a[n], ..., a[1]) ∈ N, n ≤ 100; b ∈{0, 1, ..., 9}

output: c = a × b

```
1) nhớ = 0;
2) for (i = 1; i ≤ n; i++)
   a) tg = b*a[i] + nhớ;
   b) c[i] = tg % 10;
   c) nhớ = tg/10;
3) if (nhớ > 0)
            {n++; c[n] = nhớ;}
4) output c = (c[n]...c[1]);
```

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 1** - Cho số nguyên *a* có không quá 100 chữ số và số nguyên *b*, $1 \le b \le 9$. Tính $c = a \times b$.

Thuật toán:

```
input: a = (a[n], ...., a[1]) ∈ N, n ≤ 100; b ∈{0, 1, ..., 9}
output:              c = a × b
1)    nhớ = 0;
2)    for (i = 1; i ≤ n; i++)
      a)    tg = b*a[i] + nhớ;
      b)    c[i] = tg % 10;
      c)    nhớ = tg/10;
3)    if (nhớ > 0)
                  {n++; c[n] = nhớ;}
4)    output c = (c[n]...c[1]);
```

Độ phức tạp của thuật toán:

Các bước 1), 3), 4) đều có độ phức tạp O(1), bước 2) có độ phức tạp O(*n*).

Tổng hợp (theo qui tắc cộng):

độ phức tạp của thuật toán là O(*n*).

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 2** - Giải phương trình bậc 2: $a.x^2+b.x+c = 0$
- Đầu vào: a, b, c - Số (thực hoặc nguyên), ràng buộc $a \neq 0$
- Đầu ra: Kết luận về nghiệm, x_1, x_2
- Quan hệ giữa đầu vào, đầu ra:
  - Tính delta = b2 – 4ac
  - Nếu delta<0 -> Vô nghiệm
  - Nếu delta=0 -> Có nghiệm kép
  - Nếu delta>0 -> Có 2 nghiệm phân biệt
- Độ phức tạp: n ???, T(n) ???

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 3a** - Tìm kiếm

Cho dãy số A[1..n], tìm xem giá trị x có trong dãy A hay không

- Đầu vào: A[1..n], x
- Đầu ra: Chỉ số k của mảng mà A[k]=x, k = -1 nếu x∉A[1..n]
- Quan hệ giữa đầu vào và đầu ra: Duyệt lần lượt từng phần tử (i) của dãy A và thực hiện phép so sánh phần tử đang xét A[i] và x, nếu phép so sánh bằng cho kết quả đúng -> i là chỉ số cần tìm (k=i).
- Độ phức tạp T(n)=O(n)

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 3b** - Tìm kiếm

Cho dãy số A[1..n] đã sắp xếp theo thứ tự tăng dần, tìm xem giá trị x có trong dãy A hay không

- Đầu vào: A[1..n], x
- Đầu ra: Chỉ số k của mảng mà A[k]=x, k = -1 nếu x∉A[1..n]
- Quan hệ giữa đầu vào và đầu ra: Dựa trên thông tin dãy A đã sắp xếp -> Tìm kiếm nhị phân
- Độ phức tạp $T(n)=O(\log_2 n)$

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 4** - Nhân hai ma trận vuông cấp n

A, B là hai ma trận vuông cấp n, tính ma trận C là tích của A.B

- Đầu vào: A, B
- Đầu ra: C
- Quan hệ giữa đầu vào và đầu ra: phép toán trên ma trận

$$C[i,j] = \sum_{k}^{n} A[i,k] \times B[k,j] \qquad \text{(i,j = 1..n)}$$

- Độ phức tạp T(n)=O(n^3)

III. TKTT Phương pháp trực tiếp

2. Bài toán áp dụng

- **Ví dụ 5** - Tính định thức của cấp n

A là ma trận vuông cấp n, tính định thức (det(A)) của A.

- Đầu vào: A, B
- Đầu ra: C
- Quan hệ giữa đầu vào và đầu ra: Định thức và cách tính

(khai triển theo dòng i)

$$D = \sum_{k=1}^{n} (-1)^{i+k} a[i,k] \times A[i,k]$$

- Độ phức tạp T(n)=O(n!)

NỘI DUNG BÀI HỌC

I. Giới thiệu

II. Thiết kế thuật toán
1. Modul hóa và thiết kế từ trên xuống (top-down)
2. Một số phương pháp thiết kế
3. Tối ưu thuật toán

III. Phương pháp trực tiếp
1. Lược đồ chung
2. Một số bài toán áp dụng

IV. Bài tập

Mô tả thông tin đầu vào, đầu ra và quan hệ (quy luật trên quan hệ đó:
1. Giải hệ 2 phương trình bậc nhất 2 ẩn (x, y)
2. Giải hệ n phương trình n ẩn
3. Chuyển điểm từ dạng số -> Dạng chữ (0.0-10.0)
   - 1 chữ số sau dấu thập phân
   - 2 chữ số sau dấu thập phân.
4. Chuyển số tiền dạng số chẵn -> dạng chữ
5. Chuyển số tiền dạng số có phần lẻ -> dạng chữ