CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT
Data Structure and algorithms

TPHCM - 2022

CHƯƠNG 1: TỔNG QUAN

Nội dung

- Cấu trúc dữ liệu
- Thuật toán
- Độ phức tạp của thuật toán

Cấu trúc dữ liệu

- Sự tổ chức hợp lý của các thành phần dữ liệu,
- Tập các thao tác để truy cập các thành phần dữ liệu.
- Ví dụ:
  - Mảng (array)
  - Danh sách liên kết (linked list)
  - Ngăn xếp (stack)
  - Hàng đợi (queue)
  - Cây (tree)
  - …

Nội dung

- Cấu trúc dữ liệu
- ✓ Thuật toán
- Độ phức tạp của thuật toán

Thuật toán

- Tập các bước *có thể tính toán được* để đạt được kết quả mong muốn
- Ví dụ: Tính tổng các số nguyên lẻ từ 1→n
  - B1: S=0
  - B2: i=1
  - B3: Nếu i=n+1 thì sang B7, ngược lại sang B4
  - B4: S=S+i
  - B5: i=i+2
  - B6: Quay lại B3
  - B7: Tổng cần tìm là S

Mối quan hệ của CTDL và thuật toán

CTDL + Thuật toán = Chương trình

Ví dụ

- Một chương trình quản lý điểm thi của sinh viên cần lưu trữ các điểm số của 3 sinh viên. Giả sử mỗi sinh viên có 4 điểm số ứng với 4 môn học khác nhau, dữ liệu có dạng bảng như sau:

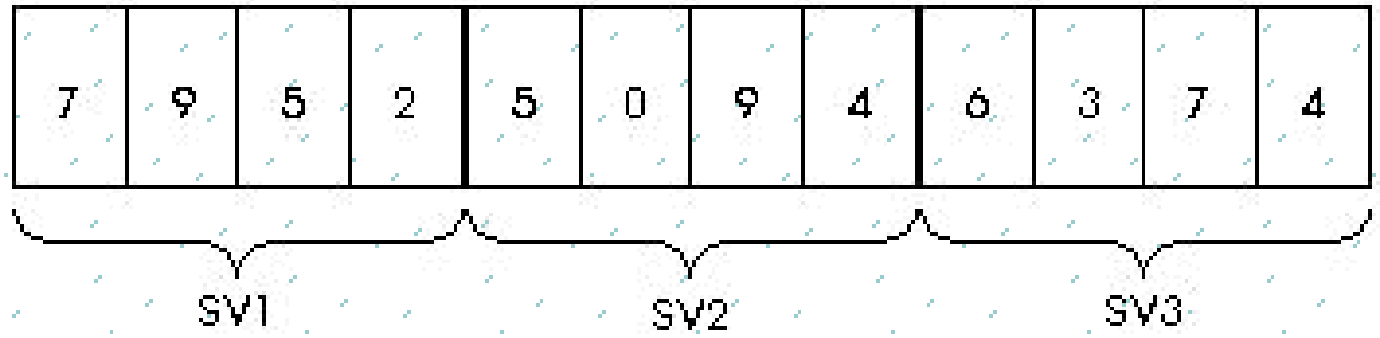
Sinh viên	Môn 1	Môn 2	Môn3	Môn4
SV 1	7	9	5	2
SV 2	5	0	9	4
SV 3	6	3	7	4

Ví dụ

- Chỉ xét thao tác xử lý là xuất điểm số các môn của từng sinh viên.
- Phương án 1 : Sử dụng mảng một chiều:

```
int result [12] = {7, 9, 5, 2, 5, 0, 9, 4, 6, 3, 7, 4};
```

  - các phần tử sẽ được lưu trữ như sau:



  - Truy xuất điểm số môn j của sinh viên i phải sử dụng một công thức xác định chỉ số tương ứng trong mảng result:

    result[(i*số cột) + j]

Ví dụ


```
void XuatDiem()//Xuất điểm số của tất cả sv
{
  const int so_mon = 4;
  int  sv, mon;
  for (int i=0; i<12; i++)
  {
     sv = i/so_mon;
     mon = i % so_mon;
     printf("Điểm môn  %d của sv %d là: %d”,
     mon, sv, result[i];
  }
}
```

Ví dụ

- Chỉ xét thao tác xử lý là xuất điểm số các môn của từng sinh viên.
- Phương án 2 : Sử dụng mảng hai chiều:

```
int result[3][4] ={{ 7, 9, 5, 2}, { 5, 0, 9, 4}, { 6, 3, 7, 4 }};
```

  - các phần tử sẽ được lưu trữ như sau:

	Cột 0	Cột 1	Cột 2	Cột 3
Dòng 0	result[0][0] =7	result[0][1] =9	result[0][2] =5	result[0][3] =2
Dòng 1	result[1][0] =5	result[1][1] =0	result[1][2] =9	result[1][3] =4
Dòng 2	result[2][0] =6	result[2][1] =3	result[2][2] =7	result[2][3] =4

  - Truy xuất điểm số môn j của sinh viên i cũng chính là phần tử nằm ở vị trí (dòng i, cột j) trong mảng: result[i][j]

Ví dụ


```
void XuatDiem()//Xuất điểm số của tất cả sv
{
  const int so_mon = 4, so_sv = 3;
  for ( int i=0; i<so_sv; i++)
      for ( int j=0; j<so_mon; j++)
          printf("Điểm môn %d của sv %d
  là:%d",j,i,result[i][j];
}
```

Nội dung

- Cấu trúc dữ liệu
- Thuật toán
- ✓ Độ phức tạp của thuật toán (algorithm complexity)

Độ phức tạp của thuật toán

- Phân tích thuật toán
  - Tính đúng
  - Tính đơn giản
  - Không gian
  - Thời gian chạy của thuật toán

Độ phức tạp của thuật toán

- Thời gian chạy của thuật toán
  - Đánh giá như thế nào
    - Thực nghiệm
    - Xấp xỉ

Độ phức tạp của thuật toán

- Thực nghiệm
  - Chịu sự hạn chế của ngôn ngữ lập trình
  - Ảnh hưởng bởi trình độ của người cài đặt
  - Chọn được các bộ dữ liệu thử đặc trưng cho tất cả tập các dữ liệu vào của thuật toán: khó khăn và tốn nhiều chi phí
  - Phụ thuộc nhiều vào phần cứng

Độ phức tạp của thuật toán

- Xấp xỉ tiệm cận
  - Cách thông dụng nhất để đánh giá một thuật toán là ký hiệu tiệm cận gọi là Big-O
  - *Định nghĩa toán học của Big-O:*

    Cho f và g là hai hàm từ tập các số nguyên hoặc số thực đến số thực. Ta nói f(x) là O(g(x)) nếu tồn tại hằng số C và k sao cho: $|f(x)| \leq C\,|g(x)|$ *với mọi* $x > k$
  - Ví dụ, hàm $f(x) = x^2 + 3x + 2$ là $O(x^2)$

    Thật vậy, khi $x > 2$ thì $x < x^2$ và $2 < 2x^2$

    Do đó $x^2 + 3x + 2 < 6x^2$

    Nghĩa là ta chọn được $C = 6$ và $k = 2$

Độ phức tạp của thuật toán

- Một số kết quả Big-O quan trọng:
  - Hàm đa thức:
    - $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$
    - Khi đó f(x) là $O(x^n)$
  - Hàm giai thừa:
    - $f(n) = n!$ là $O(n^n)$
  - Logarit của hàm giai thừa:
    - $f(n) = \log n!$ là $O(n \log n)$
  - Hàm điều hòa
    - $H(n) = 1 + 1/2 + 1/3 + .. + 1/n$ là $O(\log n)$

Độ phức tạp của thuật toán

- Một số lớp thuật toán

n \ Hàm	n	lg n	Nlgn	n^2	n^3	2^n
1	1	0	0	1	1	2
2	2	1	2	4	8	4
4	n	2	8	16	64	16
8	8	3	24	64	512	256
16	16	4	64	256	4096	65536
32	32	5	160	1024	32768	2,147,483,648

Độ phức tạp của thuật toán

- Một số lớp thuật toán

Độ phức tạp	Thuật ngữ/tên phân lớp
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
$O(n^a)$	Độ phức tạp đa thức
$O(a^n)$, a > 1	Độ phức tạp hàm mũ
O(n!)	Độ phức tạp giai thừa

Độ phức tạp của thuật toán

- Ví dụ, xét hàm sau:
  - Lệnh *printf* ngoài vòng lặp có độ phức tạp hằng O(1) , vì không phụ thuộc vào N
  - Số lệnh *printf* trong vòng lặp bằng với kích thước mảng, do đó vòng lặp có độ phức tạp O(N)
  - Tổng cộng: Hàm f có độ phức tạp: O(1) + O(N)

→ Độ phức tạp: O(N)

```
void f(int a[], int n)
{
    printf("n = %d", n) ;
    for (int i = 0; i < n; i++ )
        printf("%d  ",a[i]);
}
```

ÔN TẬP

- Các thao tác trên cấu trúc dữ liệu mảng một chiều:
  - Nhập/xuất mảng (hoặc phát sinh ngẫu nhiên - random)
  - Tìm vị trí của giá trị X trong mảng
  - Sắp xếp mảng có thứ tự
  - Ghi dữ liệu của mảng ra file text.