Cấu trúc dữ liệu - CT177

Chương 1

Giới thiệu độ phức tạp của giải thuật

Bộ môn Công Nghệ Phần Mềm

Mở đầu

- Mô hình hóa bài toán thực tế
- Giải thuật là chuỗi hữu hạn các thao tác để giải một bài toán nào đó.
- Tính chất của giải thuật:
  - Hữu hạn
  - Xác định
  - *Hiệu quả*
- Cấu trúc dữ liệu + Giải thuật = Chương trình

Kiểu dữ liệu trừu tượng

- Trừu tượng hóa chương trình
- Trừu tượng hóa dữ liệu.
- Kiểu dữ liệu trừu tượng (ADT): mô hình toán học cùng với một tập hợp các phép toán trừu tượng được định nghĩa trên mô hình đó.
- Cài đặt kiểu dữ liệu trừu tượng:
  - Tổ chức lưu trữ: một cấu trúc dữ liệu
  - Viết chương trình con thực hiện các phép toán

Độ phức tạp của giải thuật

Cần phải phân tích, đánh giá giải thuật để:

- Lựa chọn một giải thuật tốt nhất trong các giải thuật để cài đặt chương trình giải quyết bài toán đặt ra.
- Cải tiến giải thuật hiện có để được một giải thuật tốt hơn.

Độ phức tạp của giải thuật

Ví dụ: một mảng có n phần tử. Hãy sắp xếp mảng theo thứ tự tăng dần.

Giải thuật sắp xếp thông dụng dùng 2 vòng lặp i:1->n và j:1->n để đổi chỗ. Lúc này độ phức tạp thuật toán là $O(n^2)$

Một số giải thuật sắp xếp như Quicksort, độ phức tạp chỉ là O(n*log(n)).

Quy ước: log(n) được hiểu là $\log_2 n$

Độ phức tạp của giải thuật

Với n=10, thì giải thuật thông thường có thể hiểu sẽ chạy xấp xỉ là 10*10=100 phép tính, nhưng giải thuật Quicksort thì chỉ dùng khoảng 10 phép tính. Với n nhỏ, 100 hay 1000 thì chương trình đều chạy có thời gian xấp xỉ bằng nhau. Nhưng với n rất lớn, ví dụ 100000 phần tử, thì thuật toán có độ phức tạp $O(n^2)$ với O(n*log(n)) là rất khác biệt.

Độ phức tạp của giải thuật

Thời gian thực hiện một chương trình là một hàm của kích thước dữ liệu vào, ký hiệu T(n) trong đó n là kích thước (độ lớn) của dữ liệu vào.

Ví dụ : Chương trình tính tổng của n số có thời gian thực hiện là T(n) = cn trong đó c là một hằng số.

Thời gian thực hiện chương trình là một hàm không âm, tức là $T(n) \geq 0 \ \forall \ n \geq 0$.

Độ phức tạp của giải thuật

Thời gian thực hiện chương trình không chỉ phụ thuộc vào kích thước mà còn phụ thuộc vào tính chất của dữ liệu vào.

Vì vậy thường ta coi T(n) là thời gian thực hiện chương trình trong trường hợp xấu nhất trên dữ liệu vào có kích thước n, tức là: T(n) là thời gian lớn nhất để thực hiện chương trình đối với mọi dữ liệu vào có cùng kích thước n.

Độ phức tạp của giải thuật

Trong những ứng dụng thực tiễn, chúng ta không cần biết chính xác hàm T(n) mà chỉ cần biết một ước lượng đủ tốt của nó.

Ước lượng thường dùng nhất là ước lượng cận trên O-lớn

Cho một hàm T(n), T(n) gọi là có độ phức tạp f(n) nếu tồn tại các hằng C, N_0 sao cho $T(n) \leq Cf(n)$ với mọi $n \geq N_0$ (tức là T(n) có tỷ suất tăng là f(n)) và kí hiệu T(n) là O(f(n))

Độ phức tạp của giải thuật

Chú ý: O(C.f(n))=O(f(n)) với C là hằng số. Đặc biệt O(C)=O(1)

Các hàm thể hiện độ phức tạp có các dạng thường gặp sau: 1, logn, n, nlogn, n^2, n^3, **2^n, n!, n^n**.

Ba hàm cuối cùng ta gọi là dạng hàm mũ, các hàm khác gọi là hàm đa thức. Một giải thuật có độ phức tạp là một hàm đa thức thì chấp nhận được, còn các giải thuật có độ phức tạp hàm mũ thì phải tìm cách cải tiến giải thuật.

Tính độ phức tạp của giải thuật

Quy tắc cộng: Nếu T1(n) và T2(n) là thời gian thực hiện của hai đoạn chương trình P1 và P2; và T1(n)=O(f(n)), T2(n)=O(g(n)) thì thời gian thực hiện của đoạn hai chương trình đó nối tiếp nhau là T(n)=O(max(f(n),g(n))).

Quy tắc nhân: Nếu T1(n) và T2(n) là thời gian thực hiện của hai đoạn chương trình P1và P2 và T1(n) = O(f(n)), T2(n) = O(g(n)) thì thời gian thực hiện của đoạn hai đoạn chương trình đó lồng nhau là T(n) = O(f(n).g(n)).

Tính độ phức tạp của giải thuật

Thời gian thực hiện của mỗi lệnh gán, nhập, xuất là O(1)

Thời gian thực hiện của một chuỗi tuần tự các lệnh được xác định bằng qui tắc cộng.

Thời gian thực hiện cấu trúc IF là thời gian lớn nhất thực hiện lệnh sau IF hoặc sau ELSE và thời gian kiểm tra điều kiện. Thường thời gian kiểm tra điều kiện là O(1).

Tính độ phức tạp của giải thuật

Thời gian thực hiện vòng lặp là tổng (trên tất cả các lần lặp) thời gian thực hiện thân vòng lặp. Nếu thời gian thực hiện thân vòng lặp không đổi thì thời gian thực hiện vòng lặp là tích của số lần lặp với thời gian thực hiện thân vòng lặp.

Tính độ phức tạp của giải thuật

Ví dụ 1: Tính độ phức tạp

```
/*1*/ Sum=0;
/*2*/ for(i=1;i<=n;i++) {
/*3*/      scanf("%d",&x);
/*4*/      Sum=Sum+x;
       }
```

- Dòng 3,4 là các thao tác nhập, gán nên độ phức tạp là O(1). Áp dụng quy tắc cộng, độ phức tạp cho cả 3 và 4 là O(1)
- Vòng lặp 2 thực hiện n lần mỗi lần O(1)-> theo quy tắc nhân độ phức tạp của 2 là O(n)
- Dòng 1 là phép gán-> O(1)
- Áp dụng quy tắc cộng cho 1 và 2 -> T(n) = O(n)

Tính độ phức tạp của giải thuật

Ví dụ 2: Tính độ phức tạp

```
/*1*/ Sum=0;
/*2*/ for(i=1;i<=n;i++) {
/*3*/      if (i%2==0)
/*4*/        Sum=Sum+i;
      }
```

- Dòng 3 rẽ nhánh với kiểm tra điều kiện mất O(1), công việc thực hiện khi điều kiện đúng mất O(1)-> theo quy tắc cộng độ phức tạp của 3 là O(1)
- Vòng lặp 2 thực hiện n lần mỗi lần O(1)-> theo quy tắc nhân độ phức tạp của 2 là O(n)
- Dòng 1 là phép gán-> O(1)
- Áp dụng quy tắc cộng cho 1 và 2 -> T(n) = O(n)

Tính độ phức tạp của giải thuật

Ví dụ 3: Tính độ phức tạp

```
/*1*/ Sum=0;
/*2*/ for(i=1;i<=n;i=i*2) {
/*3*/      scanf("%d",&x);
/*4*/      Sum=Sum+x;
       }
```

- Cách tính tương tự như ví dụ 1, tuy nhiên ở đây do vòng lặp 2 thực hiện ~ logn lần, do đó T(n) = O(logn)

Ví dụ 4: Tính độ phức tạp

```
/*1*/ Sum1=0;
/*2*/ k=1;
/*3*/ while (k<=n) {
/*4*/    for(j=1;j<=n;j++)
/*5*/       Sum1=Sum1+1;
/*6*/    k=k*2;
       }
```

Q&A?