Nhập môn
Phân tích độ phức tạp thuật toán

Chủ đề 4: Tổng quan về tiếp cận lý thuyết để phân tích độ phức tạp thuật toán

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Tóm tắt nội dung chủ đề 4

Chủ đề này trình bày

- Tổng quan về phương pháp toán học để đánh giá và phân tích độ phức tạp của các thuật toán.
- Đánh giá một họ những thuật toán đơn giản nhờ phương pháp chung.

Nội dung chủ đề 4

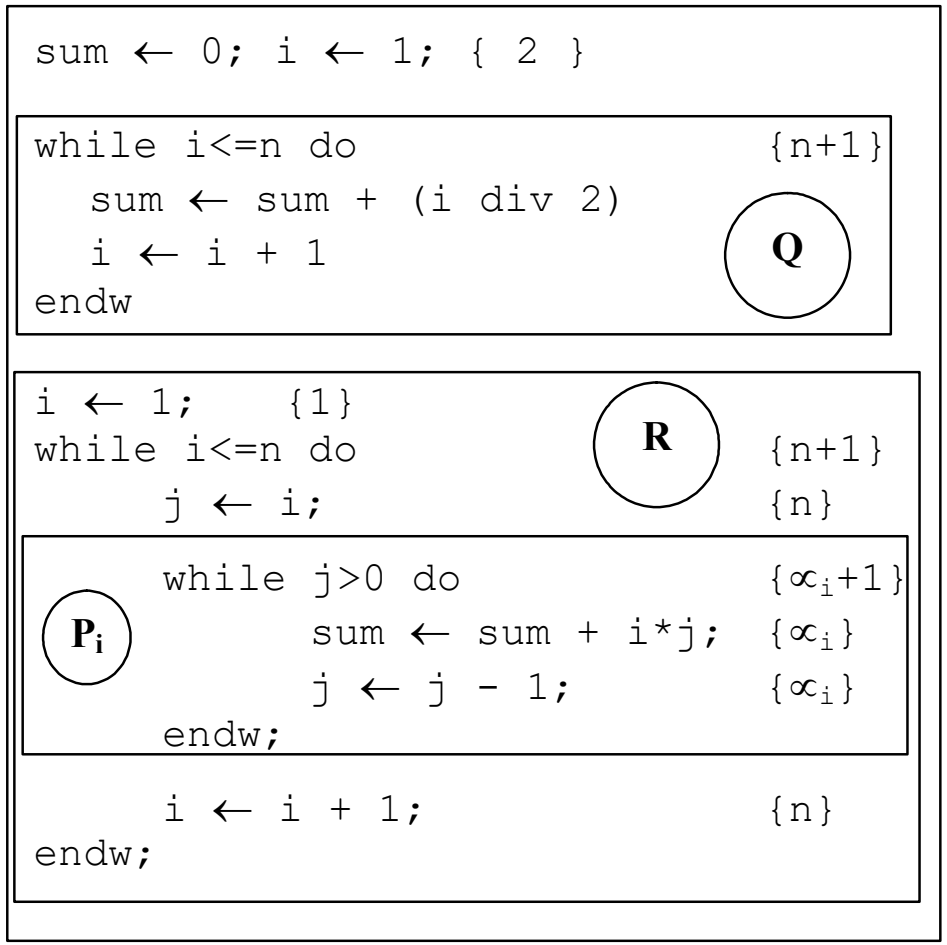
- Ý tưởng chung về đánh giá độ phức tạp về mặt lý thuyết
- Kỹ thuật phân đoạn thuật toán
- Kỹ thuật chặt miền giá trị
- Phân loại theo cấu trúc điều khiển
- Công thức truy hồi

Đánh giá độ phức tạp lý thuyết

- Có kết quả khá chặt chẽ, chính xác
- Không cần thực nghiệm, có trước khi thực nghiệm
- Công cụ toán học đa dạng, không dễ
- Khó khăn đối với thuật toán hoàn toàn mới, có bản chất phức tạp
- Phải lưu ý về các giả định, các điều kiện cho dữ liệu nhập

Phân đoạn thuật toán

- Phân thành các đoạn rời nhau hay lồng nhau và đóng khung
- Lượng giá các thao tác trong khung, không quan tâm ngoài khung
- Tổng kết lại các thao tác nhờ các tổng số (có thể rời nhau hay lồng nhau)
- Xem minh họa trên bảng cho thuật toán bên cạnh



Thêm một ví dụ minh họa việc phân đoạn thuật toán

- Xem tính toán trên bảng
- Lưu ý về vòng lặp của P_i
- Lưu ý về giá trị của α_i

```
S ← 0;  i ← 1;                    {2}
while i<=n do                     {n+1}        Q
    j  ← i;                       {n}
    while j <= n - i do           {1 + αi}
      S ← S+i*j;                  { αi }       Pi
      j ← j+1;                    { αi }
    endw;
    i ← i+1;                      {n}
endw;
```

Kỹ thuật chặt miền giá trị

- *Mở rộng miền giá trị* của các biến đếm để có chận trên của một thuật toán.
- *Thu gọn miền giá trị* của các biến đếm để có chận dưới của một thuật toán.
- Chẳng hạn, xem hai vòng lặp lồng nhau:

```
For i = 1, 2, …, n
   For j = i, i+1, …, n
```

Vòng lặp **for j** lặp chính xác là $n - i + 1$ có giá trị trong {1, 2, …, n} tùy theo i chạy từ 1 đến n.

- Nới rộng: cho j chạy từ 1, làm trội **For j** thành n lần. Như vậy tối đa là n x n = n^2 lần lặp.
- Thu hẹp: giới hạn $i \leq n/2$ thì vòng lặp **For j** lặp lớn hơn $n/2$ lần. Suy ra lặp tối thiểu là $n/2$ x $n/2$ = $n^2/4$.
- Kết luận tổng số lần lặp là $\Theta(n^2)$.

Áp dụng chặt miền giá trị cho thuật toán phần trước

- Xem tính toán trên bảng
- Làm trội vòng lặp trong
- Làm non vòng lặp ngoài
- Thuật toán có độ phức tạp $\Theta(n^2)$.

```
S ← 0;  i ← 1;                    {2}
while i<=n do                     {n+1}        Q
    j  ← i;                       {n}
    while j <= n - i do           {1 + αi}
      S ← S+i*j;                  { αi }       Pi
      j ← j+1;                    { αi }
    endw;
    i ← i+1;                      {n}
endw;
```

Thêm ví dụ về chặt miền giá trị

```
i ← 1, S ← 0, m ← sqrt(n) // m = Căn bậc hai của n
while i <= m do              // Vòng lặp ngoài
    j ← m – 2*i
    while j >= i do          // Vòng lặp trong
        S ← S + j + i * j    // Cập nhật giá trị tính toán
        j ← j-1
    end while
    i ← i + 1
end while
```

Phân loại theo cấu trúc điều khiển

- Lặp không rẽ nhánh
  - Số lần lặp xác định theo kích thước bài toán
    - Xác định tường minh, hiển nhiên
    - Xác định tiềm ẩn theo logic thuật toán (phải tính ra)
  - Số lần lặp thay đổi ngẫu nhiên theo phân bố dữ liệu
- Lặp có rẽ nhánh
  - Số lần lặp: cũng xét như trên
  - Tần suất các nhánh rẽ hoàn toàn xác định theo kích thước bài toán: *tường minh* hoặc *tiềm ẩn*
  - Tần suất các nhánh rẽ thay đổi ngẫu nhiên theo phân bố dữ liệu

Xem minh họa trực tiếp trên bảng…

Công thức truy hồi

Nhiều trường hợp thuật toán có thể dùng công thức truy hồi để khảo sát độ phức tạp

- Bản thân thuật toán mang tính đệ quy
- Từ bản chất thủ thuật tính toán của thuật toán, có thể thiết lập công thức truy hồi để tính số thao tác
- Dùng các kỹ thuật toán học để tính toán dãy truy hồi → độ phức tạp

Xem minh họa trực tiếp trên bảng…

CÂU HỎI

&

THẢO LUẬN