CƠ BẢN VỀ QUY HOẠCH ĐỘNG

Nguyễn Tô Sơn

Hiện nay, bộ môn Tin học đã được đưa vào trường phổ thông. Giờ đây, các em học sinh lớp 11 đã được cung cấp những kiến thức căn bản để có thể lập trình. Trong bài viết này, tôi mong muốn giúp đỡ các em học một kiến thức nâng cao hơn so với chương trình sách giáo khoa về thuật toán Quy hoạch động.

Ta làm một phép so sánh giữa thuật toán Quy hoạch động và Chia để trị. Điểm giống nhau giữa chúng là thực hiện *chia bài toán thành các bài toán con nhỏ hơn có cùng một loại vấn đề,* giải từng bài toán con đó một cách lần lượt, sau đó kết hợp các lời giải con để thu được lời giải của bài toán nguyên thủy.

Điểm khác biệt giữa chúng: *Chia để trị chia bài toán thành các bài toán con độc lập, sau đó các bài toán con này được giải một cách đệ quy, và cuối cùng tổng hợp các lời giải của bài toán con để được lời giải của bài toán đặt ra;* còn đặc điểm của *Quy hoạch động là nó chia bài toán thành các bài toán con không độc lập, có nghĩa là các bài toán con cùng có chung các bài toán con nhỏ hơn (còn gọi là các bài toán con gối lên nhau).* Điều đó có nghĩa rằng, Quy hoạch động giải đi giải lại một bài toán nhưng nó sẽ sử dụng một phương pháp hiệu quả hơn. Trong khi đó, nếu sử dụng Chia để trị sẽ lãng phí rất nhiều thời gian giải lại các bài toán con đó.

Như đã nói ở trên, để tránh tình trạng giải đi giải lại một bài toán, Quy hoạch động thực hiện phương pháp lưu trữ kết quả của các bài toán con đó trên một bảng. Khi cần tìm kết quả của một bài toán con nào ta chỉ cần lấy trực tiếp từ bảng ra.

Tuy nhiên, cái khó nhất để giải một bài toán Quy hoạch động (QHĐ) là ở chỗ, ta biết rằng nó là một bài toán QHĐ, nghĩa là cần phải xác định được cấu trúc con tối ưu (các bài toán con gối lên nhau) của bài toán đặt ra. Để giúp các em tiếp cận dễ dàng với thuật toán này, tôi đề xuất việc giải quyết thuật toán này theo 5 bước như sau:

Bước 1: Xác định cấu trúc con tối ưu của bài toán.

Bước 2: Xây dựng công thức truy hồi thể hiện mối quan hệ giữa các bài toán con. Trong các bài toán QHĐ thì nó có một tên gọi mới là công thức QHĐ.

Bước 3: Xác định trình tự tính toán như thế nào cho hợp lý? Nghĩa là: làm thế nào để trước khi giải bài toán con nào đó ta đã giải cái bài toán con nhỏ hơn nó rồi. Bước này là bước xác định đâu là bài toán con, đâu là bài toán lớn hơn.

Bước 4: Giải quyết trực tiếp các bài toán con cơ bản như thế nào?

Bước 5: Dựa trên giá trị tối ưu của các bài toán con xác định được từ công thức QHĐ tính toán, truy vết tìm ra nghiệm của bài toán ban đầu.

Ở đây, bước 1 và bước 2 có mối quan hệ chặt chẽ với nhau.

Để các em có thể hình dung cụ thể từng bước thực hiện như thế nào, tôi sẽ viết chi tiết nội dung giải hai bài toán QHĐ cơ bản:

- Tìm mảng con liên tục có tổng lớn nhất.
- Giải bài toán nhân ma trận

1. Tìm mảng con liên tục có tổng lớn nhất

Chúng ta sẽ tuần tự thực hiện 5 bước trên:

Bước 1: Xác định cấu trúc con tối ưu của bài toán

Bằng cách tiếp cận tự nhiên, chúng ta xác định được: mảng con liên tục có tổng lớn nhất cần tìm sẽ phải kết thúc tại một trong n vị trí: hoặc 1, hoặc 2,…, hoặc n. Vậy nếu ta gọi *L(i) là tổng của mảng con liên tục có tổng lớn nhất lấy từ miền a_1 đến a_i và phần tử cuối cùng là a_i* thì tổng của mảng con liên tục có tổng lớn nhất là: max{L(i) với $1 \le i \le n$}.

Ta đã đạt được việc xác định được cấu trúc con tối ưu cho bài toán, điều này được chứng tỏ bằng bước 2 ngay sau đây:

Bước 2: Xây dựng công thức QHĐ

Do mảng con liên tục lớn nhất từ a_1 tới a_i hoặc là chứa a_{i-1} hoặc là không chứa a_{i-1}. Trường hợp không chứa a_{i-1} thì mảng con liên tục đó chỉ chứa một phần tử là a_i mà thôi (vì yêu cầu mảng phải liên tục). Vậy:

```
L(i) = max{a[i], L(i-1) + a[i]}
```

Bước 3: Xác định trình tự tính toán thế nào cho hợp lý?

Trường hợp đơn giản này chỉ cần tính toán theo trình tự tự nhiên.

```
for i:= 2 to n do
 L[i]:= max(a[i], L[i-1] + a[i]);
```

Bước 4: Xác định bài toán con cơ bản

```
L[1]:= a[1];
```

Bước 5: Truy vết, tìm nghiệm

```
max:= -∞;
for i:= 1 to n do
 if L[i] > max then
 begin
      max:= L[i];
      k:= i;
 end;

i:= k;
while max > 0 do
begin
      writeln(i, a[i]);
      max:= max – a[i];
      i:= i – 1;
end;
```

2. Giải bài toán nhân ma trận

Cho dãy ma trận $A_1,A_2,\ldots,A_n$ với kích thước các ma trận cho bởi dãy $d_0,d_1,\ldots,d_n$. Trong đó ma trận A_i có kích thước $d_{i-1}\times d_i$. Hãy xác định trình tự nhân tối ưu cho dãy ma trận trên.

Bước 1: Xác định cấu trúc con tối ưu của bài toán

Gọi F(i,j) là số phép nhân tối ưu để tính tích các ma trận từ A_i đến A_j ($A_i.A_{i+1}\ldots A_j$)

Bước 2: Xây dựng công thức QHĐ

Ta có thể tính $A_i.A_{i+1}\ldots A_j$ bằng cách chọn một vị trí k nào đó để đặt dấu ngoặc theo trình tự:

$$A_i.A_{i+1}\ldots A_j = (A_i\ldots A_k).(A_{k+1}\ldots A_j)$$

Ma trận kết quả của phép nhân (A_i...A_k) có kích thước $d_{i-1}\times d_k$, ma trận kết quả của phép nhân (A_{k+1}…A_j) có kích thước $d_k\times d_j$. Với cách đặt đó ta sẽ mất F(i,k) phép nhân để có kết quả trong dấu ngoặc thứ nhất, mất thêm F(k+1,j) phép nhân để có kết quả trong dấu ngoặc thứ hai, và cuối cùng mất $d_{i-1}d_k.d_j$ để nhân 2 ma trận kết quả đó. Từ đó tổng số phép nhân của cách đặt đó là:

$$F(i,k) + F(k+1,j) + d_{i-1}.d_k.d_j$$

Ta chọn vị trí k cho số phép nhân cần thực hiện là ít nhất:

Bước 3: Xác định trình tự tính toán thế nào cho hợp lý?

Ta có nhận xét: Để tính được F(i,j) ta phải tính F(i,k) và F(k+1,j) trước. Vậy ta cần tìm trình tự tính toán thế nào cho hợp lý.

Tôi đề ra một giải pháp bằng cách tiếp cận đơn giản: ***j-i lớn hơn k-i và j-k.*** Vậy ta tính theo trình tự: ***tính các phần tử F(i,j) với j-i từ nhỏ đến lớn*** (không phải là tính các gá trị F(i,j) từ nhỏ đến lớn như vẫn thường làm). Với cách đó, khi tính đến F(i,j) thì ta đã có F(i,k) và F(k+1,j).

```
for m:= 2 to n - 1 do { m = j – i }
 for i:= 1 to n - m do
 begin
      j:= i + m;
      F[i,j]:= +∞;
      for k:= i to j – 1 do
           F[i,j]:= min(F[i,j],
           F[i,k] + F[k+1,j] + d[i-1]*d[k]*d[j]);
 end;
```

Bước 4: Xác định bài toán con cơ bản

```
for i:= 1 to n do F[i,i]:= 0;
for i:= 1 to n - 1 do
 F[i,i+1]:= d[i-1]*d[i]*d[i+1];
```

Bước 5: Truy vết, tìm nghiệm

Để tìm lời giải tối ưu, ta sử dụng H(i,j) ghi nhận cách đặt dấu ngoặc tách đầu tiên cho giá trị F(i,j). Vậy ta tinh chế lại chương trình ở bước 3 và bước 4 thành:

```
for i:= 1 to n do F[i,i]:= 0;
for i:= 1 to n - 1 do
```

```
begin
      F[i,i+1]:= d[i-1]*d[i]*d[i+1];
      H[i,i+1]:= i + 1;
end;
for m:= 2 to n - 1 do { m = j – i }
 for i:= 1 to n - m do
 begin
      j:= i + m;
      F[i,j]:= +∞;
      for k:= i to j – 1 do
           if F[i,j] >
           F[i,k] + F[k+1,j] + d[i-1]*d[k]*d[j]) then
      begin
           F[i,j]:=
           F[i,k] + F[k+1,j] + d[i-1]*d[k]*d[j]);
           H[i,j]:= k;
      end;
 end;
```

Sau đó, quá trình truy vết tìm nghiệm lại là một hàm Chia để trị như sau:

```
procedure TruyVet(i,j): string;
begin
 if i < j then
 begin
      k:= H[i,j];
      X:= TruyVet(i,k);
      Y:= TruyVet(k+1,j);
      return (X*Y);
 end
      else { i = j }
           return A[i];
end;
```

(Các chương trình trong bài viết được viết trên ngôn ngữ tựa Pascal để bạn đọc có thể tự cài đặt trên một ngôn ngữ cụ thể)

Liên hệ: Nguyễn Tô Sơn – ĐHSP Hà Nội: tosonnguyen@gmail.com

TÀI LIỆU THAM KHẢO

[1]. Thuật toán và độ phức tạp tính toán, Vũ Đình Hòa - Đỗ Trung Kiên, NXB Đại học Sư phạm.

[2]. Một số bài toán Quy hoạch động điển hình, Nguyễn Thanh Tùng, Ebook.

[3]. Bài tập Quy hoạch động, Trần Đỗ Hùng - Đỗ Đức Đông - Lê Sĩ Quang, NXB Giáo dục.

[4]. Giải thuật & Lập trình, Lê Minh Hoàng, Ebook.