|  |
| --- |
| **TRƯỜNG ĐẠI HỌC SƯ PHẠM THÀNH PHỐ HỒ CHÍ MINH**  **KHOA CÔNG NGHỆ THÔNG TIN**  **🙠🕮🙢**  KHOA CNTT  **THUẬT TOÁN QUY HOẠCH ĐỘNG**  **( DYNAMIC PROGRAMMING)**  **Sinh viên thực hiện: Phạm Hoài Nhân**  **Mã số sinh viên: 47.01.104.147**  **Mã lớp học phần: COMP140101**  **Thành phố Hồ Chí Minh, ngày 10 tháng 01 năm 2022** |

**MỤC LỤC**

NỘI DUNG 4

**I. KHÁI QUÁT THUẬT TOÁN QUY HOẠCH ĐỘNG4**

I.1. Khái niệm4

I.2. Các khái niệm trong bài toán quy hoạch động4

I.3. Yêu cầu đối với một bài toán quy hoạch động4

I.4. Các bước cài đặt bài toán quy hoạch động5

I.5. Các dạng bài toán quy hoạch động5

I.6. Ưu điểm và hạn chế của phương pháp quy hoạch động5

I.6.1 Ưu điểm5

I.6.2 Hạn chế5

**II. PHÂN TÍCH ĐỘ PHỨC TẠP THUẬT TOÁN QUY HOẠCH ĐỘNG7**

**III. ỨNG DỤNG THUẬT TOÁN CHO BÀI TOÁN CỤ THỂ 8**

III.1 Mô tả bài toán8

III.2. Cách giải8

III.2.1 Cơ sở quy hoạch động (bài toán nhỏ nhất)8

III.2.2 Công thức truy hồi8

III.2.3 Tính bảng phương án và truy vết9

III.2.4. Phân tích độ phức tạp thuật toán10

**IV. THIẾT KẾ BÀI TOÁN ỨNG DỤNG11**

IV.1. Mô hình xử lí bài toán11

IV.2. Các chức năng chính11

IV.3. Dữ liệu đầu vào, đầu ra12

**V. CÀI ĐẶT BÀI TOÁN ỨNG DỤNG ỨNG VỚI 10 BỘ TEST CASE13**

V.1. Tổng quan về chương trình13

V.2. Kết quả 10 bộ test case cho chương trình14

V.3. Kết quả đạt được15

V.4. Hạn chế16

V.5. Hướng phát triển16

TÀI LIỆU THAM KHẢO 17

**LỜI MỞ ĐẦU**

Đối với một số thuật toán do bản chất tự nhiên của chúng đã mang tính đệ quy nên việc cài đặt đệ quy (Recursion) là gọn và đẹp nhất, nhưng khuyết điểm của đệ quy là ở lời giải đệ quy cho một số bài toán có thể chạy rất chậm do sự bùng nổ tổ hợp khi kích thước bài toán gia tăng. Vì thế Quy hoạch động (Dynamic Programming) được sinh ra nhằm giải quyết vấn đề bằng cách giải quyết bài toán con và lưu trữ lời giải của chúng với mục đích sử dụng lại theo một cách nào đó để giải quyết bài toán tổng quát hơn.

Phương pháp quy hoạch động đã được áp dụng để giải hàng loạt bài toán thực tế trong các quá trình kĩ thuật công nghệ, tổ chức sản xuất, kế hoạch hoá kinh tế,… với ưu điểm là chương trình thực thi nhanh do không tốn thời gian giải lại bài toán con nhưng khuyết điểm là số lượng bài toán con cần giải và lưu trữ kết quả rất lớn và việc kết hợp lời giải của các bài toán con chưa chắc cho lời giả của bài toán ban đầu. Do đó có một số bài toán mà cách giải bằng quy hoạch động tỏ ra không thích hợp.

**NỘI DUNG**

**I.KHÁI QUÁT THUẬT TOÁN QUY HOẠCH ĐỘNG**

*I.1 Khái niệm*

Quy hoạch động (Dynamic Programming) là kĩ thuật lập trình dùng để giải quyết những bài toán có tính chất tối ưu bằng cách chia nhỏ bài toán thành những bài toán con và lưu trữ lời giải cho chúng với mục đích sử dụng lại theo quy tắc nào đó để giải quyết những bài toán tổng quát hơn.

Khác với phép phân giải đệ quy theo hướng “từ trên xuống” (top – down) tức là từ bài toán lớn phân rã thành các bài toán con và đi giải từng bài toán con đó, quy hoạch động theo hướng “từ dưới lên” tức là giải những bài toán cơ sở để từ đó từng bước giải quyết những bài toán lớn hơn cho tới khi giải được bài toán lớn nhất (bài toán ban đầu).

*I.2 Các khái niệm trong bài toán quy hoạch động*

Bài toán quy hoạch động là bài toán được giải theo phương pháp quy hoạch động.

Công thức phối hợp nghiệm của các bài toán con để ra được nghiệm của bài toán lớn hơn gọi là công thức truy hồi.

Tập các bài toán con nhỏ nhất có ngay lời giải gọi là cơ sở quy hoạch động.

Không gian lưu trữ lời giải những bài toán con để tìm cách phối hợp chúng gọi là bảng phương án của quy hoạch động.

*I.3 Yêu cầu đối với một bài toán quy hoạch động*

Bài toán muốn giải theo phương pháp quy hoạch động cần thoả mãn những yêu cầu sau:

Bài toán lớn phải chia nhỏ thành nhiều bài toán con mà sự phối hợp của những lời giải của bài toán con đó cho ra được lời giải của bài toán lớn.

Cần có một không gian lưu trữ đủ lớn để lưu trữ lời giải cho tất cả các bài toán con (vì bản chất quy hoạch động là đi giải tất cả các bài toán con cho đến khi giải được bài toán lớn nhất ban đầu).

Quá trình từ bài toán cơ sở tìm ra được lời giải bài toán ban đầu phải qua một hữu hạn các bước.

*I.4 Các bước cài đặt bài toán quy hoạch động*

Phân tích bài toán và lập công thức truy hồi (tương tự phương pháp đệ quy).

Giải các bài toán cơ sở và lưu vào bảng phương án.

Dùng công thức tuy hồi phối hợp những lời giải của bài toán con đã lưu trong bảng trong phương án để tìm ra lời giải cho bài toán lớn hơn và lưu chúng vào trong bảng phương án. Lặp lại cho đến khi tìm được lời giải của bài toán ban đầu.

Dựa vào bảng phương án, truy vết tìm ra nghiệm tối ưu.

*I.5 Các dạng bài toán quy hoạch động*

Bài toán tối ưu: Là bài toán tìm ra đáp án tốt nhất từ mục tiêu của bài toán, mỗi bài toán con sẽ được giải quyết dựa trên bài toán con đã giải trước đó cho đến khi tìm được đáp án của bài toán đã cho.

Bài toán tổ hợp: Là bài toán yêu cầu ta đi tìm số cách khác nhau để thực hiện một việc gì đó. Sự khác biệt cơ bản của dạng bài toán này với dạng bài toán tối ưu là ở chỗ chúng ta cần tính tổng thay vì tìm số lớn nhất hoặc nhỏ nhất.

*I.6 Ưu điểm và hạn chế của phương pháp quy hoạch động*

***I.6.1 Ưu điểm:***

Tiết kiệm được thời gian thực hiện vì không phải thực hiện việc tính toán lại các bài toán con giống nhau.

***I.6.2 Hạn chế:***

- Việc tìm công thức truy hồi hoặc phân rã bài toán nhiều khi đòi hỏi sự phân tích tổng hợp khó khăn, dễ sai sót, cần nhiều thời gian suy nghĩ. Và không phải lúc nào việc kết hợp lời giả các bài toán con cũng cho ra kết quả của bài toán lớn hơn.

- Khi bảng lưu trữ đòi hỏi mảng hai, ba chiều,… thì khó có thể xử lí dữ liệu với kích cỡ mỗi chiều lớn đến hàng trăm.

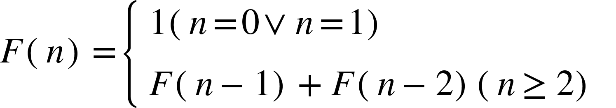
- Có những bài toán tối ưu không thể giải được bằng quy hoạch động.

=> Kết luận: Không phải lúc nào việc kết hợp các bài toán con cũng cho ta kết quả của bài toán lớn hơn. Hay nói cách khác là việc tìm kiếm công thức truy hồi rất khó khăn. Ngoài ra số lượng kết quả các bài toán con có thể rất lớn, khó có thể lưu trữ với dữ liệu và bộ máy cho phép.

**II. PHÂN TÍCH ĐỘ PHỨC TẠP THUẬT TOÁN QUY HOẠCH ĐỘNG**

Trong các bài toán quy hoạch động, độ phức tạp thời gian là số trạng thái/ bài toán con cần phải giải nhân với thời gian thực thi ở mỗi trạng thái/ bài toán con. Việc sử dụng bảng phương án đã giúp cho độ phức tạp bài toán được cải thiện đi rất nhiều so với phương pháp lập trình đệ quy như bài toán balo (Knapsack Problem), xâu con chung dài nhất (Longest Common Subquence), nhân hai ma trận (Matrix Product Parenthesization),…

Ví dụ: Việc áp dụng quy hoạch động vào việc tính giá trị phần từ của dãy Finonanci:



long Fibo (int n)

{

if (n==1 || n==0) return 1;

else

{

Fn1=Fibo(n-1);

Fn2=Fibo(n-2);

return Fn1 + Fn2;

}

}

long Fibo (int n)

{

long F[101];

F[0]=F[1]=1;

for (int i=1;i<=n;i++)

{

F[i]=F[i-1] + F[i-2];

}

return F[n];

}

Ta có thể thấy:

* Cột bên trái phân rã các bài toán thành các bài toán nhỏ hơn và đi giải từng bài toán nhỏ đấy bằng đệ quy. Việc sử dụng đệ quy đã khiến cho chương trình phải tính đi tính lại nhiều lần các bài toán con F0, F1, F2,…. Suy ra độ phức tạp của thuật toán này là O(a to the power of n) với a almost equal to 1.61803.
* Cột bên phải xử lí các bài toán nhỏ hơn , và dựa vào đó làm cơ sở để giải các bài toán lớn hơn và vì thế chương trình đảm bảo rằng mỗi giá trị *Fibonanci* chỉ phải tính 1 lần. Vì thế giải thuật này có độ phức tạp tuyến tính là O(n).

**III. ỨNG DỤNG THUẬT TOÁN CHO MỘT BÀI TOÁN CỤ THỂ ( BÀI TOÁN TÌM DÃY CON ĐƠN ĐIỆU TĂNG DÀI NHẤT)**

*III.1 Mô tả bài toán*

Cho dãy số nguyên a gồm *n* phần tử. Một dãy con của *a* là một cách chọn trong *a* một số phần từ giữ nguyên thứ tự ( có tất cả 2*n* dãy con). Hãy tìm dãy con đơn điệu tăng của *a*  có độ dài lớn nhất.

Ví dụ:

Dãy con đơn điệu tăng dài nhất của *a =* (5, 2, 3, 4, 9, 10, 5, 6 ,7, 8) là (2, 3, 4, 5, 6, 7, 8).

*III.2 Cách giải*

Giả sử dãy tăng ban đầu gồm *a*[1], *a*[2],… *a*[*n*-1]. Bổ sung vào *a* hai phần tử *a*[0]equals negative infinity và *a*[*n*+1]equals plus infinity ( khi viết chương trình negative infinityvà plus infinity sẽ được cài đặt các giá trị thích hợp). Khi đó dãy con đơn điệu tăng dài nhất sẽ bắt đầu từ *a*[0] và kết thúc ở *a*[*n*+1].

Đặt *L*(i) là độ dài dãy con đơn điệu tăng dài nhất bắt đầu tại *a*[i] (với điều kiện 0 less or equal than i less or equal than n plus 1). Ta cần tính tất cả *L*(i) này. Đáp số bài toán sẽ là dãy ứng với *L*(*i*0) có giá trị lớn nhất.

***III.2.1 Cơ sở quy hoạch động (bài toán nhỏ nhất)***

Trường hợp đặc biệt *L*(*n*+1) là độ dài dãy con đơn điệu tăng dài nhất bắt đầu tại *a*[*n*+1]equals plus infinity. Do đó dãy con này chỉ gồm một phần từ (là plus infinity) nên *L*(*n*+1) = 1.

***III.2.2 Công thức truy hồi***

Ta cần tính *L*(*i*) với *I* chạy từ *n* về 0. Giá trị *L*(*i*) sẽ được tính trong điều kiện *L*(*i+1*),… *L*(*n*+1) đã biết.

Ta nhận thấy dãy con đơn điệu tăng dài nhất bắt đầu từ *a*[*i*] sẽ được thành lập bằng cách lấy *a*[*i*] ghép vào đầu một trong số những dãy con đơn điệu tăng dài nhất bắt đầu tại một vị trí *a*[*j*] nào đó đứng sau *a*[*i*]. Tất nhiên ta phải chọn một trong các vị trí *a*[*j*] lớn hơn *a*[*i*] (để đảm bảo tính tăng) và chọn dãy dài nhất trong số đó để ghép *a*[*i*] vào đầu (để đảm bảo tính dài nhất). Như vậy *L*(*i*) được tính bằng cách xét tất cả các chỉ số *j* trong đoạn từ *i*+1 đến *n*+1 mà *a*[*j*]> *a*[*i*], chọn ra chỉ số *jmax* có *L*(*jmax*) lớn nhất.

Vậy ta có *L*(*i*) = *L*(*jmax*) +1 hay

*L*(*i*) = max { *L*(*j*) + 1 / i less than j less or equal than n plus 1, *a*[*i*] less than *a*[*j*] }.

***III.2.3 Tính bảng phương án và truy vết***

Tại bước xây dựng dãy *L*, mỗi khi gán *L*(*i*) = *L*(*jmax*) +1, ta đặt giá trị *T*(*i*) = *jmax* để lưu lại rằng dãy con dài nhất sẽ bắt đầu tại *a*[*i*] sẽ có phần tử kế tiếp là *a*[*jmax*].

Như vậy, sau khi tính xong hai dãy *L* và *T*, ta sẽ bắt đầu từ *T*[0] và được dãy kết quả: *T*[0] rightwards arrow *T*[*T*[0]] rightwards arrow *T*[*T*[*T*[0]]] rightwards arrow …

Ví dụ.

Với *A* = (5, 2, 3, 4, 9, 10, 5, 6, 7, 8). Dãy *L* và *T* sẽ được tính từ 11 về 0 với giá trị cụ thế như trong bảng dưới đây. Kết quả truy vết là

*T*[0] rightwards arrow *T*[2] rightwards arrow *T*[3] rightwards arrow *T*[4] rightwards arrow *T*[7] rightwards arrow *T*[8] rightwards arrow *T*[9] rightwards arrow *T*[10] rightwards arrow *T*[11];

Nhờ đó, dãy con tăng dài nhất là (2, 3, 4, 5, 6, 7, 8).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| a[i] | negative infinity | 5 | 2 | 3 | 4 | 9 | 10 | 5 | 6 | 7 | 8 | plus infinity |
| L[i] | 9 | 5 | 8 | 7 | 6 | 3 | 2 | 5 | 4 | 3 | 2 | 1 |
| T[i] | 2 | 8 | 3 | 4 | 7 | 6 | 11 | 8 | 9 | 10 | 11 |  |

***III.2.4 Phân tích độ phức tạp thuật toán***

void dayDaiNhat (int n, int a[], int L[], int T[])

{

    int kq=1,jmax; (1)

    a[0]=-1000; (2)

    a[n+1]=1000; (3)

    L[n+1]=1; (4)

    for (int i=n;i>=0;i--) (5)

    {

        jmax=n+1; (6)

        for (int j=i+1;j<=n+1;j++) (7)

        {

            if (a[j]>a[i] && L[j]>L[jmax]) (8)

jmax=j; (9)

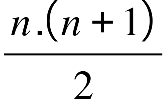
        }

        L[i]=L[jmax]+1; (10)

        T[i]=jmax; (11)

    }

}

* Câu lệnh (1), (2), (3), (4) có thời gian thực thi O open parentheses 1 close parentheses.
* Câu lệnh (9) có thời gian thực thi O open parentheses 1 close parentheses.
* Câu lệnh (8) có thời gian thực thi là O open parentheses 1 close parentheses.
* Câu lệnh (6), (10), (11) có thời gian thực thi là O open parentheses 1 close parentheses.
* Ứng với mỗi giá trị *i* của câu lệnh (5), thì:
* Khi *i*=0, lệnh lặp (7) thực hiện *n* lần.
* Khi *i*=1, lênh lặp (7) thực hiện *n*-1 lần.
* Khi *i*=2, lênh lặp (7) thực hiện *n*-2 lần.
* Như vậy lệnh lặp (7) có số lần thực hiện là , từ đó suy ra cả lệnh lặp (5) có thời gian thực hiện almost equal to O open parentheses n squared close parentheses
* Cả chương trình có thời gian thực thi là O open parentheses n squared close parentheses

**IV. THIẾT KẾ BÀI TOÁN ỨNG DỤNG**

*IV.1 Mô hình xử lí bài toán*

* Đặt số nguyên *n* là số lượng phần tử mảng.
* Tạo mảng lưu trữ dãy số cần xét là *a*[101].
* Bảng phương án *L*[101], bảng truy vết *T*[101].
* Kết quả số lượng chuỗi con dài nhất sẽ là max (*L*[*i*], với 1 less or equal than i less or equal than n).
* Truy vết: *T*[0] rightwards arrow *T*[*T*[0]] rightwards arrow *T*[*T*[*T*[0]]] rightwards arrow …

*IV.2 Các chức năng chính*

Ngoài các chức năng nhập xuất cơ bản ra thì còn có thêm giao diện cơ bản giúp chương trình dễ nhìn hơn và các hàm xử lí chương trình khác.

* Hàm giao diện:

int Menu(string menu[], int focus, int len); // trả về lựa chọn của người dùng

void MenuChinh(); // tạo giao diện menu chính khi chạy chương trình lần đầu tiên

void LuaChon(int a, string menu[], int focus, int len); //Xử lí các lựa chọn của user

* Hàm xử lí phụ:

void docDuLieu(int& n, int a[], ifstream& in);// đọc dữ liệu từ file

int timMax(int n, int a[]); // tìm chuỗi con dài nhất từ bảng phương án

* Các hàm chính:
* Hàm tạo bảng phương án:

void dayDaiNhat(int n, int a[], int L[], int T[])

{

int kq = 1, jmax;

a[0] = -1000;

a[n + 1] = 1000;

L[n + 1] = 1;

for (int i = n; i >= 0; i--)

{

jmax = n + 1;

for (int j = i + 1; j <= n + 1; j++)

{

if (a[j] > a[i] && L[j] > L[jmax]) jmax = j;

}

L[i] = L[jmax] + 1;

T[i] = jmax;

}

}

* Hàm truy vết:

void truyVet(int n, int a[], int T[])

{

int i = T[0];

int dem = 0;

while (i != n + 1)

{

cout << a[i] << " ";

i = T[i];

dem++;

}

}

*IV.3 Dữ liệu đầu vào, đầu ra*

* Input: dòng đầu là một số nguyên n, n dòng tiếp theo mỗi dòng là một số nguyên dương *i.*
* Output: dòng đầu là số lượng chuỗi con dài nhất, dòng thứ hai xuất ra chuỗi con dài nhất đó.

|  |  |
| --- | --- |
| Input | Output |
| 10  5 2 3 4 9 10 5 6 7 8 | 7  2 3 4 5 6 7 8 |

**V. CÀI ĐẶT BÀI TOÁN ỨNG DỤNG ỨNG VỚI 10 BỘ TEST CASE**

*V.1 Tổng quan về chương trình*

Bài toán ứng dụng được chọn để chạy chương trình ở đây là Dãy con tăng dài nhất, được viết bằng ngôn ngữ C++, có thêm một ít giao diện cơ bản để người dùng có thể dễ nhìn hơn mỗi khi sử dụng.

Ở màn hình chính bao gồm 3 chức năng: Nhập dữ liệu từ bàn phím, nhập dữ liệu từ file và thoát chương trình, người dùng sẽ dùng các phìm mũi tên và nút Enter để lựa chọn các chức năng:

Text

Description automatically generated

* Ở phần nhập dữ liệu từ bàn phím, người dùng sẽ nhập lần lượt số lượng phần từ và các phần tử của mảng.

Text

Description automatically generated

* Ở phần nhập dữ liệu từ file, chương trình sẽ tự động nhập số lượng phần từ và các phần từ của mảng ở trong file dữ liệu cho trước.
* Dù là nhập từ bàn phìm hay nhập từ file thì đều trả ra cùng một kết quả là dãy con tăng dài nhất và số lượng phần từ của dãy.

Text

Description automatically generated

*V.2 Kết quả 10 bộ test case cho chương trình*

|  |  |  |
| --- | --- | --- |
| 1. | 10  5 2 3 4 9 10 5 6 7 8 |  |
| 2. | 6  1 2 5 4 6 2 |  |
| 3. | 9  3 4 1 3 7 8 19 2 3 |  |
| 4. | 5  7 4 3 3 1 |  |
| 5. | 12  5 20 9 12 4 19 8 3 2 10 14 1 |  |
| 6. | 8  -9 -10 12 17 4 -5 2 13 |  |
| 7. | 15  -5 1 -2 29 25 20 30 0 27 4 -9 -7 -1 11 13 |  |
| 8. | 18  20 22 15 12 30 -4 25 18 0 -5 6 4 26 11 24 2 23 8 |  |
| 9. | 30  25 15 -4 -2 21 3 7 17 12 27 29 1 13 30 5 16 9 24 10 0 26 -7 23 22 28 18 20 6 19 -10 |  |
| 10. | 40  21 19 -8 -5 15 23 24 4 25 10 7 -6 1 0 27 29 17 2 18 22 8 5 -2 -9 -7 -10 16 14 -1 3 -4 9 12 30 13 26 28 6 -3 11 |  |

*V.3 Kết quả đạt được*

Sau khoảng thời gian nghiên cứu và hoàn thiện bài tiểu luận, em cũng đã rút ra được những kết luận sau:

Về mặt lí thuyết: bài tiểu luận đã trình bày được cơ sở lí thuyết của thuật toán quy hoạch động, các dạng bài toán, cách tiếp cận cũng như ưu nhược điểm của thuật toán này.

Về mặt thực tiễn: bài tiểu luận đã mô phỏng chương trình bài toán ứng dụng là bài tìm dãy con tăng dài nhất một cách tối ưu.

*V.4 Hạn chế*

- Nội dung về lí thuyết quy hoạch động chưa được đi sâu hơn.

- Việc phân tích độ phức tạp thuật toán quy hoạch động còn hạn chế vì không có những bài toán cụ thể hơn.

- Hạn chế về mặt chương trình khi sử dụng ngôn ngữ C++ để viết. (thiếu các chức năng giao diện hiển thị hoặc báo lỗi cho người dùng).

*V.5 Hướng phát triển*

- Tìm hiểu sâu hơn cũng như cài đặt đa dạng các bài toán quy hoạch động hơn

- Tạo giao diện đẹp đẽ và thuận tiện hơn cho người dùng.

- Tìm hiểu những framwork khác có thể để tạo giao diện trực quan hơn.

- Tìm cách cải tiến thuật toán cũng như các bài toán đã được thực thi và ứng dụng.

**TÀI LIỆU THAM KHẢO**

1. Trần Đan Thư (2014). *Kỹ Thuật Lập Trình.* Nhà Xuất Bản Khoa học và Kỹ thuật.
2. Lê Minh Hoàng (1999 – 2002). *Giải Thuật và Lập Trình.* Đại học Sư phạm Hà Nội.
3. Ngọc Anh Thư (2001). *Giáo trình Thuật Toán.* Nhà Xuất Bản Thống Kê.
4. Hồ Sĩ Đàm (2009). *Tài liệu giáo khoa chuyên Tin (Quyển 1).* Nhà Xuất Bản Giáo dục Việt Nam.

**LINK CHƯƠNG TRÌNH**

https://drive.google.com/file/d/1IxTYA2TFtOkH4z5oMYaIvvGtkTpfe\_2P/view?usp=sharing