**Dạng bài toán con phủ chồng:** Từ bài toán lớn có thể chia thành bài toán nhỏ hơn có dạng tương tự như vậy. Quá trình phân rã bài toán lớn cứ tiếp tục cho đến khi gặp bài toán con mà có thể giải được ngay (bài toán cơ sở, neo). Tìm mối liên hệ giữa bài toán con với bài toán lớn (bài toán tổng quát, công thức truy hồi).

**Thuật toán quay lui:**

Phương pháp quay lui, vét cạn, thử sai, duyệt … là một số tên gọi tuy không đồng nghĩa nhưng cùng chỉ một phương pháp trong tin học: tìm nghiệm của một bài toán bằng cách xem xét tất cả các phương án có thể.

***Ưu điểm:*** luôn đảm bảo tìm ra nghiệm đúng, chính xác.

***Nhược điểm:*** thời gian thực thi lâu, độ phức tạp lớn. Do đó chỉ phù hợp với các bài toán có kích thước nhỏ.

**Thuật toán nhánh cận:**

* Khó khăn: tìm ra nghiệm cận.

**Thuật toán quy hoạch động:**

* Khó khăn: tìm ra công thức truy hồi của bài toán, đưa bài toán phước tạp nhiều ngữ cảnh về lớp bài toán thường gặp. Không gian lưu trữ nghiệm của các bài toán con bị giới hạn.

Phương pháp quy hoạch động dùng để giải bài toán tối ưu có bản chất đệ quy, tức là việc tìm phương án tối ưu cho bài toán đó có thể đưa về tìm phương án tối ưu của một số hữu hạn các bài toán con.

Ðối với một số bài toán đệ quy, nguyên lý chia để trị (divide and conquer) thường đóng vai trò chủ đạo trong việc thiết kế thuật toán. Ðể giải quyết một bài toán lớn, ta chia nó thành nhiều bài toán con cùng dạng với nó để có thể giải quyết độc lập.

Trong phương án quy hoạch động, nguyên lý chia để trị càng được thể hiện rõ: Khi không biết phải giải quyết những bài toán con nào, ta sẽ đi giải quyết toàn bộ các bài toán con và lưu trữ những lời giải hay đáp số của chúng với mục đích sử dụng lại theo một sự phối hợp nào đó để giải quyết những bài toán tổng quát hơn.

Ðó chính là điểm khác nhau giữa Quy hoạch động và phép phân giải đệ quy và cũng là nội dung phương pháp quy hoạch động:

* Phép phân giải đệ quy bắt đầu từ bài toán lớn phân ra thành nhiều bài toán con và đi giải từng bài toán con đó. Việc giải từng bài toán con lại đưa về phép phân ra tiếp thành nhiều bài toán nhỏ hơn và lại đi giải các bài toán nhỏ hơn đó bất kể nó đã được giải hay chưa.
* Quy hoạch động bắt đầu từ việc giải tất cả các bài toán nhỏ nhất (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).

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

Công thức phối hợp nghiệm của các bài toán con để có nghiệm của bài toán lớn gọi là công thức truy hồi của quy hoạch động.

Tập các bài toán có ngay lời giải để từ đó giải quyết các bài toán lớn hơn gọi là cơ sở quy hoạch động.

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

Trước khi áp dụng phương pháp quy hoạch động ta phải xét xem bài toán đó có thỏa mãn những yêu cầu dưới đây không:

* Bài toán lớn phải phân rã được thành nhiều bài toán con, mà sự phối hợp lời giải của các bài toán con đó cho ta lời giải của bài toán lớn.
* Vì quy hoạch động là đi giải tất cả các bài toán con, nên nếu không đủ **không gian vật lý lưu trữ lời giải** (bộ nhớ, đĩa, …) để phối hợp chúng thì phương pháp quy hoạch động cũng không thể thực hiện được.
* Quá trình từ bài toán cơ sở tìm ra lời giải bài toán ban đầu phải qua hữu hạn bước. Các bước cài đặt một chương trình sử dụng quy hoạch động:
* Giải tất cả các bài toán cơ sở (thông thường rất dễ), lưu các lời giải vào bảng phương án.
* Dùng công thức truy hồi phối hợp những lời giải của các bài toán nhỏ đã lưu trong bảng phương án để tìm lời giải của các bài toán lớn hơn rồi lưu chúng vào bảng phương án. Cho tới khi bài toán ban đầu tìm được lời giải.
* Dựa vào bảng phương án, truy vết tìm ra nghiệm tối ưu.
* **Các bước thực hiện quy hoạch động**

Giải bài toán bằng phương pháp quy hoạch động phải qua 3 giai đoạn:

* ***Phân rã:*** Chia bài toán cần giải thành những bài toán con nhỏ hơn có cùng dạng với bài toán ban đầu sao cho bài toán con kích thước nhỏ nhất có thể giải một cách trực tiếp. Bài toán xuất phát có thể coi là bài toán con có kích thước lớn nhất.
* ***Giải các bài toán con và ghi nhận lời giải:*** Lưu trữ lời giải của các bài toán con vào một bảng để sử dụng lại nhiều lần do đó không phải giải lặp lại cùng một bài toán.
* ***Tổng hợp lời giải:*** Lần lượt từ lời giải của các bài toán con kích thước nhỏ hơn xây dựng lời giải của bài toán kích thước lớn hơn, cho đến khi thu được lời giải của bài toán xuất phát (là bài toán con có kích thước lớn nhất).

Các yếu tố của một giải thuật quy hoạch động giải bài toán tối ưu:

* ***Cơ sở của quy hoạch động:*** Những trường hợp đơn giản có thể tính trực tiếp.
* ***Cấu trúc con tối ưu:*** Phương pháp chia nhỏ các bài toán cho đến khi gặp được bài toán cơ sở.
* ***Tổng hợp:*** Hệ thức truy hồi tính giá trị tối ưu của hàm mục tiêu của bài toán lớn qua giá trị tối ưu của các bài toán con thành phần.

Hiệu quả của quy hoạch động:

* Khi có các bài toán con lồng nhau, phương pháp chia để trị sẽ tỏ ra không hiệu quả, khi nó phải lặp đi lặp lại việc giải các bài toán con chung đó.
* Quy hoạch động sẽ giải mỗi bài toán con một lần và lời giải của các bài toán con sẽ được ghi nhận, để thoát khỏi việc giải lại bài toán con mỗi khi ta đòi hỏi lời giải của nó.
* Quy hoạch động thường được áp dụng để giải các bài toán tối ưu. Trong các bài toán tối ưu, ta có một tập các lời giải, và một hàm mục tiêu nhận giá trị số. Ta cần tìm một lời giải để hàm mục tiêu đạt giá trị nhỏ nhất hoặc lớn nhất.

**Hãy code 2 bài tập dưới đây bằng hai thuật toán: Quay lui và quy hoạch động**

**Bài 1: Bài toán tính N!** **(GT)**

Ta có định nghĩa như sau: *n*! =  nếu 

Cho một số nguyên dương *n* (0 ≤ *n* ≤ 13).

**Yêu cầu:** Hãy tính *n*! bằng phương pháp quy hoạch động .

**Dữ liệu vào:** Ghi trong file văn bản GT.INP có cấu trúc như sau:

*- Dòng 1:* Ghi số nguyên dương *n*.

**Dữ liệu ra**: Ghi ra file văn bản GT.OUT theo cấu trúc như sau:

*- Dòng 1:* Ghi giá trị tính được của *n*!

**Ví dụ:**

|  |  |
| --- | --- |
| GT.INP | GT.OUT |
| 3 | 6 |

**Hướng dẫn:**

-Bài toán cơ sở: gt(0) = 1 khi n = 0

-Bài toán tổng quát: gt(n) = n \* gt(n-1) khi n >= 1

**Code:**

**-Thuật toán quay lui**

fi=open("GT.INP","r")

fo=open("GT.OUT","w")

li=fi.readline()

n=int(li)

def bt(k):

if k == 0:

return 1

return k\*bt(k-1)

kq = bt(n)

print(kq,file = fo)

fi.close()

fo.close()

**-Thuật toán quy hoạt động:**

fi=open("GT.INP","r")

fo=open("GT.OUT","w")

li=fi.readline()

n=int(li)

f = []

f.append(1)

for i in range(1,n+1):

f.append(f[i-1]\*i)

kq = f[n]

print(kq,file = fo)

fi.close()

fo.close()

**Bài 2: Xác định số Fibonacci thứ n** **(FIBO)**

Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1, các phần tử sau đó được thiết lập theo quy tắc *mỗi phần tử luôn bằng tổng hai phần tử trước nó*. Công thức truy hồi của dãy Fibonacci là:

Fn = F(n) =

Cho một số nguyên dương *n* (0 ≤ *n* ≤ 50).

***Yêu cầu:*** Hãy tính *F*(*n*) bằng phương pháp quy hoạch động.

***Dữ liệu vào:*** Ghi trong file văn bản FIBO.INP có cấu trúc như sau:

*- Dòng 1:* Ghi số nguyên dương *n*.

***Dữ liệu ra***: Ghi ra file văn bản FIBO.OUT theo cấu trúc như sau:

*- Dòng 1:* Ghi giá trị tính được của *F*(*n)*.

**Ví dụ:**

|  |  |
| --- | --- |
| FIBO.INP | FIBO.OUT |
| 6 | 8 |

**Hướng dẫn:**

-Bài toán cơ sở:

+f(0) = 0 khi n = 0

+f(1) = 1 khi n = 1

-Bài toán tổng quát:

f(n) = f(n-1) + f(n-2) khi n>1

**Code:**

**-Thuật toán quay lui:**

fi=open("FIBO.INP","r")

fo=open("FIBO.OUT","w")

li=fi.readline()

n=int(li)

def bt(k):

if k == 0:

return 0

if k == 1:

return 1

return bt(k-1)+bt(k-2)

kq = bt(n)

print(kq,file = fo)

fi.close()

fo.close()

**-Thuật toán quy hoạch động**

fi=open("FIBO.INP","r")

fo=open("FIBO.OUT","w")

li=fi.readline()

n=int(li)

f = []

f.append(0)

f.append(1)

if n>1:

for i in range(2,n+1):

f.append(f[i-1]+f[i-2])

kq = f[n]

print(kq,file = fo)

fi.close()

fo.close()

**\*Khảo sát tốc độ chạy của hai thuật toán:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bài** | **GTBT** | **GTQHD** | **FIBOBT** | **FIBOQHD** |
| N = 10 | Thời gian ≈ 0.230516518 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.262644344 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.234492163 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.230283807 giây  Kết quả khớp đáp án! |
| N = 15 | Thời gian ≈ 0.230446118 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.254700388 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.231484518 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.229012207 giây  Kết quả khớp đáp án! |
| N = 20 | Thời gian ≈ 0.241326831 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.240149586 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.239222653 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.224983273 giây  Kết quả khớp đáp án! |
| N = 30 | Thời gian ≈ 0.231526074 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.231999318 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.910416338 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.224983273 giây  Kết quả khớp đáp án! |
| N = 40 | Thời gian ≈ 0.227740118 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.234364074 giây  Kết quả khớp đáp án! | Chạy quá thời gian | Thời gian ≈ 0.243073142 giây  Kết quả khớp đáp án! |
| N = 50 | Thời gian ≈ 0.228166918 giây  Kết quả khớp đáp án! | Thời gian ≈ 0.233726563 giây  Kết quả khớp đáp án! | Chạy quá thời gian | Thời gian ≈ 0.229208251 giây  Kết quả khớp đáp án! |

**Xem thêm**

**Bài 3:** Bài toán Mê cung **(MECUNG.PAS)**

Phát biểu bài toán: Trong một chuyến thám hiểm mạo hiểm, một đoàn thám hiểm không may lọt vào mê cung với nhiều cạm bẫy. Trong mê cung đó chỉ có một lối ra duy nhất, lối ra gồm các ô hình vuông được xếp thành một hàng dài. Muốn đi được ra ngoài mọi người phải bước qua một hàng các ô hình vuông đó và phải bước theo quy tắc sau:

Quy tắc 1: Mỗi bước chỉ có thể bước một ô hoặc hai ô hoặc ba ô.

Quy tắc 2: Từ người thứ hai trở đi bước theo quy tắc 1 và không trùng với cách bước của tất cả những người trước đó.

Hỏi đoàn thám hiểm đó còn lại tối thiểu bao nhiêu người không thể thoát ra khỏi mê cung đó được?

***Dữ liệu vào:*** Ghi trong file văn bản MECUNG.INP có cấu trúc như sau:

* *Dòng 1:* Ghi một số nguyên dương m (m ≤ 1018) là số người trong đoàn thám hiểm.
* *Dòng 2:* Ghi một số nguyên dương n (n ≤ 70) là tổng số ô vuông.

***Dữ liệu ra***: Ghi ra file văn bản MECUNG.OUT gồm một số nguyên duy nhất là số người còn lại tối thiểu không thể thoát ra khỏi mê cung.

**Ví dụ:**

|  |  |
| --- | --- |
| MECUNG.INP | MECUNG.OUT |
| 20  5 | 7 |

Thực chất của bài toán là tìm xem có bao nhiêu cách bước ra ngoài.

Công thức truy hồi cho bài toán này được tính như sau:

Để bước lên ô thứ n ta có 3 cách bước:

Cách 1: Bước tới ô thứ n – 3 rồi bước 3 bước nữa.

Cách 2: Bước tới ô thứ n – 2 rồi bước 2 bước nữa.

Cách 3: Bước tới ô thứ n – 1 rồi bước 1 bước nữa.

Theo nguyên lý cộng ta có tổng số cách bước tới ô thứ n: F(n) = F(n-3) + F(n-2) + F(n-1) trong đó: F(1) = 1, F(2) = 2, F(3) = 4.

Sử dụng mảng A để lưu kết quả.

A[1] := 1;

A[2] := 2;

A[3] := 4;

For i := 4 to n do

A[i] := A[i-1] + A[i-2] + A[i-3];