Quy hoạch động

1. Quy hoạch động là gì?

+) Định Nghĩa: Quy hoạch động là một kĩ thuật thiết kế thuật toán theo kiểu phân rã bài toán lớn thành các bài toán con, kết hợp lời giải của các bài toán con để tìm lời giải cho bài toán ban đầu.

+) Khi nào thì sử dụng Quy Hoạch Động?

- Bài toán quy hoạch động không có công thức nào cụ thể. Tuy nhiên,có 1 số tính chất mà khi gặp,ta có thể nghĩ đến việc sử dụng QHĐ:

-Bài toán có các bài toán con gối nhau. VD: Dãy Fibonaci…

-Bài toán có cấu trúc con tối ưu. VD:Bài toán tìm đường đi ngắn nhất…

+)Phân tích thuật toán Quy Hoạch Động:

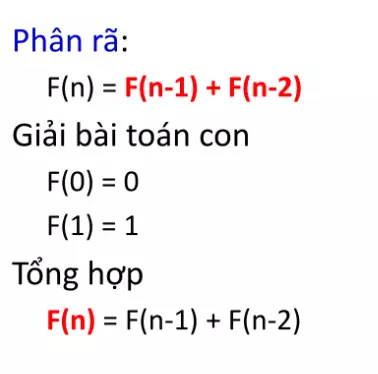
* Bước 1: Giải tất cả các bài toán cơ sở, lưu “Trạng Thái” vào bảng phương án.

-Trạng thái là một trường hợp, một bài toán con của bài toán lớn.

Ví dụ, trạng thái trong bài này là số lượng xu nhỏ nhất để tổng bằng i, với i≤S. Để tìm ra trạng thái i, cần phải tìm tất cả các trạng thái j mà (j<i). Một khi đã tìm ra trạng thái i, ta có thể dễ dàng tìm ra trạng thái của i+1..

* Bước 2: Sử dụng công thức truy hồi, phối hợp lời giải của các bài toán con, từ đó tìm ra lời giải của bài toán lớn và tiếp tục lưu vào bảng phương án. Thực hiện như vậy tới khi tìm ra lời giải của bài toán ban đầu.
* Bước 3: Dựa vào bảng phương án, truy vết tìm ra nghiệm tối ưu của bài toán.

VD: Bài toán tìm số fibonaci:



+) So sánh Quy hoạch động và Đệ Quy

|  |  |
| --- | --- |
| Đệ Quy | Quy Hoạch Động |
| -Phân rã bài toán lớn thành những bài toán con, sau đó giải quyết từng bài toán con.  -Mỗi bài toán con lại đưa về phân rã thành những bài toán con nhỏ hơn và đi tìm lời giải tiếp,bất kể nó đã được giải trước đó hay chưa. | -Giải các bài toán con nhỏ,lưu lại vào 1 bảng đáp án.  -Kết hợp lời giải của các bài toán con để giải bài toán lớn. |

**+) Các loại quy hoạch động.**

- Quy hoạch động ngược:

+Là duyệt qua tất cả các bài toán con (từ nhỏ đến lớn), với mỗi bài toán đó, chúng ta tính toán kết quả dựa vào bài toán con trước đó. Việc dựa vào kết quả các bài toán trước đó được gọi là “nhìn ngược lại”

+Phương pháp quy hoạch động ngược này được sử dụng rộng rãi, vì nó khá tương ứng với suy nghĩ tự nhiên của chúng ta.

+Cách giải đó yêu cầu phải giải những bài toán nhỏ hơn. Ta cứ suy nghĩ các bài toán nhỏ để rồi tổng hợp lại thành bài toán lớn.

+Kiểu quy hoạch động này gần với đệ quy.

VD: Bài toán FIbonaci: F[n]=F[n-1]+ F[n-2]. Vì thế nên phải nhìn lại kết quả của F[n-1] và của F[n-2].

-Quy hoạch động xuôi:

+Không quá phổ biến, kiểu quy hoạch động xuôi cũng khá khó áp dụng, nhưng quy hoạch động “xuôi” mang đến cho chúng ta nhiều tiện lợi.

+Kiểu xuôi này cũng cần duyệt qua các bài toán con từ nhỏ đến lớn, nhưng với mỗi bài toán con, chúng ta tính toán kết quả và từ đó tìm cách thực hiện một số phép tính để giải bài toán lớn hơn.

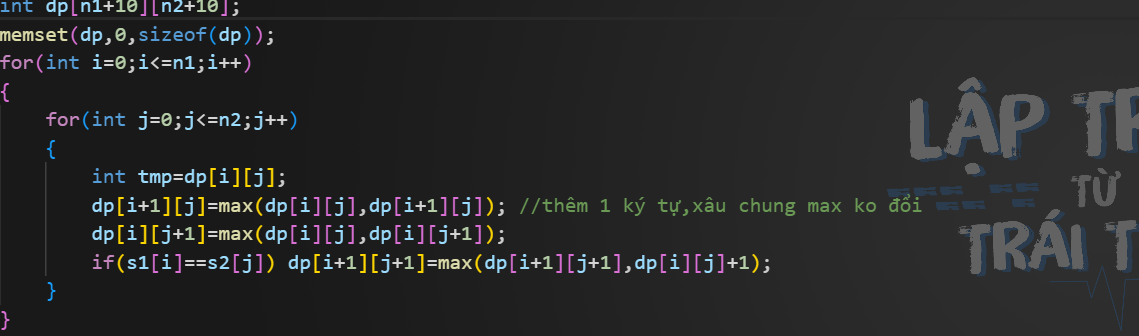
+ Với mỗi bài toán, việc xác định bước tiếp theo tương đối khó khăn, thậm chí việc kiểm tra tính đúng sai của phương pháp cũng không hề dễ dàng.

+ Với mỗi bài toán, việc xác định bước tiếp theo tương đối khó khăn, thậm chí việc kiểm tra tính đúng sai của phương pháp cũng không hề dễ dàng.

VD: Xâu con chung nhỏ nhất

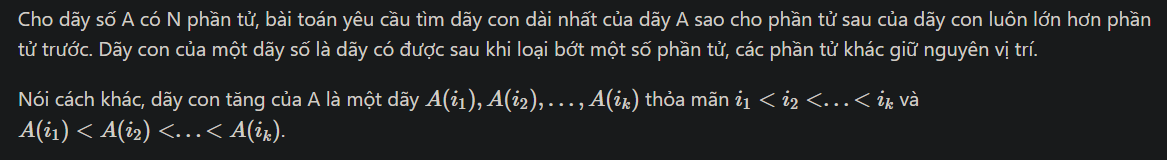
Hướng giải: dp(0,0) = 0 là bài toán với hai xâu rỗng

* Với mỗi bài toán dp(i, j) chúng ta sẽ tìm cách tính toán kết quả cho các bài toán lớn hơn. Lúc này, có 3 hướng phát triển tiếp:
  1. Lấy thêm một ký tự từ xâu thứ nhất => Kết quả không thay đổi.
  2. Lấy thêm một ký tự từ xâu thứ hai => Kết quả cũng không thay đổi.
  3. Nếu ký tự tiếp theo của cả hai xâu giống nhau => Lấy tự từ này và độ dài xâu con chung tăng lên 1.



II> Các bài toán điển hình:

1. Bài toán Dãy con tăng dài nhất:



- Ý tưởng: Quy Hoạch Động:

+Tạo 1 mảng DP[n+5] để lưu trữ: DP[i] Lưu độ dài dãy con tăng lớn nhất từ 1-> I.

+ Công thức truy hồi:

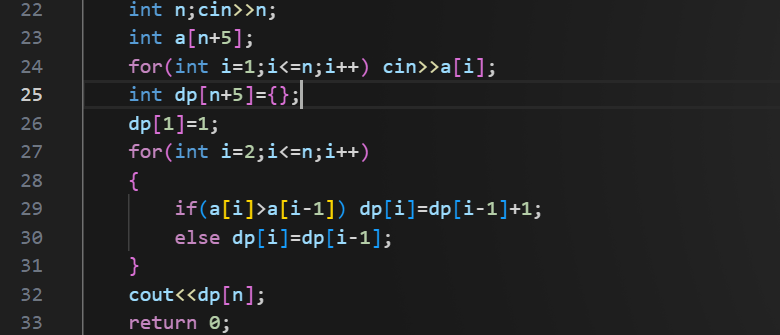
. DP[1]=1;

. DP[k] (K>1)=: + Nếu a[k]>a[k-1] DP[k]= DP[k-1] +1;

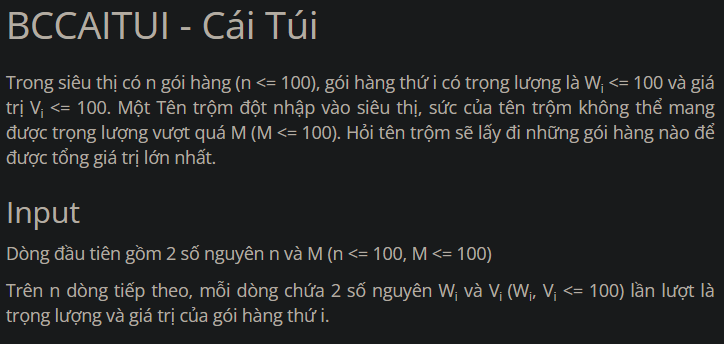
+ Nếu a[k]<=a[k-1] DP[k]= DP[k-1];

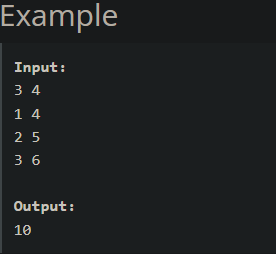
=> Kết quả -DP[N].

- Code:



1. Bài toán Cái Túi:





//Ta sẽ lấy được các túi: 1 và 3.

-Ý tưởng: Ta sẽ sử dụng Quy Hoạch Động 2 chiều:

+Tạo 1 mảng 2 chiều dp[i][j]=k : với: +k là tổng giá trị của các túi đã lấy được.

+ j là Trọng lượng các túi còn đựng được..

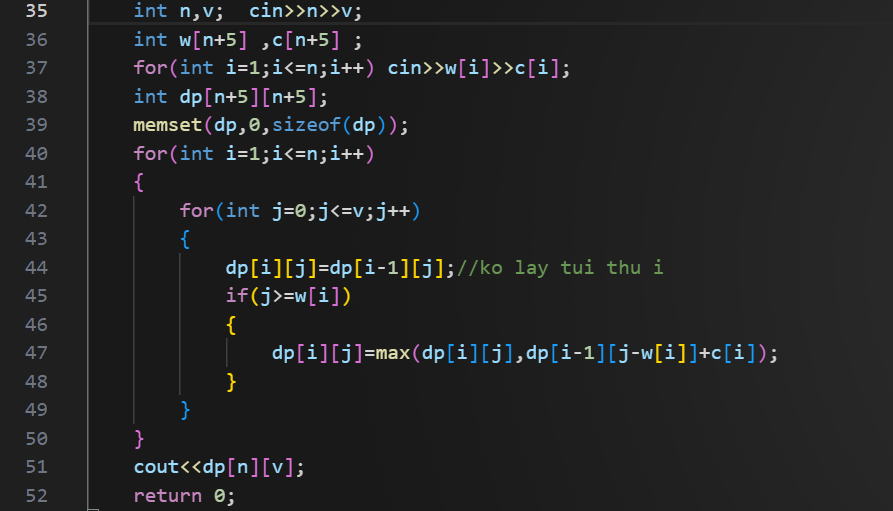
+Các túi lấy được là các túi trong khoảng từ túi thứ 1=> I

+Công thức truy hồi:

-dp[I][j]= max[ (dp[I-1][j-w[I]]) +c[I]) ,dp[I-1][j] ]

//Bằng MAX của ( số tiền lớn nhất nếu có lấy túi thứ I và không lấy túi thứ I)

-Code:

s