QUY HOẠCH ĐỘNG

<DYNAMIC PROGRAMMING>

1. **Khái niệm**
2. **Quy hoạch động là gì? Trạng thái là gì**
   1. **Tư tưởng:**

* Trong giải thuật chia để trị, người ta phân bài toán cần giải thành các bài toán con. Các bài toán con lại được tiếp tục phân thành các bài toán con nhỏ hơn, cứ thế tiếp tục cho tới khi ta nhận được các bài toán con có thể giải được 1 cách trực tiếp.
* Tuy nhiên, trong quá trình phân chia như vậy, có thể ta sẽ gặp rất nhiều lần cùng một bài toán con*.* Tư tưởng cơ bản của phương pháp quy hoạch động là sử dụng một bảng để lưu giữ lời giải của các bài toán con đã được giải. Khi giải một bài toán con cần đến nghiệm của bài toán con cỡ nhỏ hơn, ta chỉ cần lấy lời giải ở trong bảng mà không cần phải giải lại, do đó làm giảm thời gian tính toán. Chính vì thế mà các thuật toán được thiết kế bằng quy hoạch động sẽ rất hiệu quả.
  1. **Định nghĩa:**
* **Trạng thái là** một trường hợp, một bài toán con của 1 bài toán lớn.
* **Quy hoạch động là** kĩ thuật được dùng khi có một công thức hoặc một (một vài) trạng thái bắt đầu. Một bài toán được tính bởi các bài toán nhỏ hơn đã được tìm ra từ trước, và kết quả các bài toán sẽ được lưu lại để những lần tính toán tiếp theo nếu cần đến những kết quả đó thì không cần tốn thêm thời gian thực hiện lại những bài toán này nữa.

1. **Khi nào thì dùng thuật toán quy hoạch động**

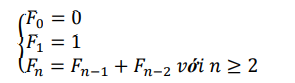
Sau đây là hai tính chất chính của một bài toán gợi ý rằng bài toán đã cho có thể được giải quyết bằng cách sử dụng Lập trình động.

* + ***Bài toán có các bài toán con gối nhau:***

Bài toán con này được gọi đi gọi lại. Phương pháp quy hoạch động sẽ lưu kết quả của bài toán con này, và khi được gọi, nó sẽ không cần phải tính lại, do đó làm giảm thời gian tính toán.

* ***Bài toán có cấu trúc con tối ưu:*** Cấu trúc con tối ưu là một tính chất là lời giải của bài toán lớn sẽ là tập hợp lời giải từ các bài toán nhỏ hơn.
* **Ví dụ: Số Fibonacci thứ n**

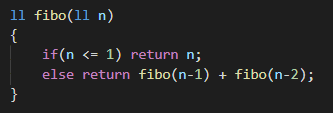
Số Fibonacci được xác định bởi công thức:



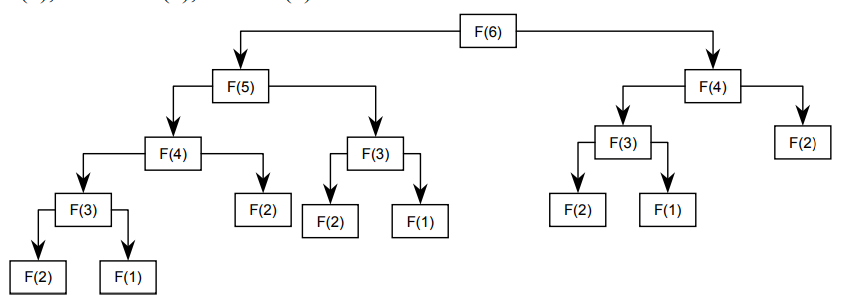
**Cách 1: *Áp dụng phương pháp chia để trị, ta tính Fn dựa vào Fn–1 và Fn–2.***

* + **Độ phức tạp:** O(2n)

1. **Cài đặt**



1. **Cây trạng thái (n=6)**



* + **Nhận xét:** Để tính F(6) cần tính 1 lần F(5), hai lần F(4), ba lần F(3), năm lần F(2), ba lần F(1).

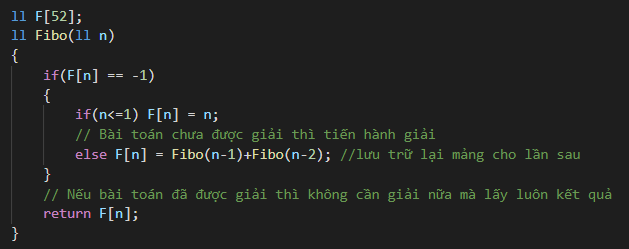
**Cách 2:** ***Phương pháp quy hoạch động***

\*Ý tưởng: Sử dụng mảng F[0…maxN], F[i] để lưu lại lời giải cho bài toá tính số Fibonacci thứ i

**(Sẽ trình bày sau, ở cách tiếp cận QHĐ)**

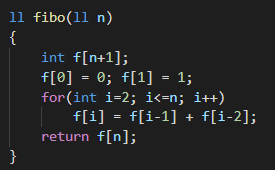
1. **Các loại QHĐ**
   1. **QHĐ sử dụng cách tiếp cận *top-down (từ trên xuống)***

Bài toán được chia thành các bài toán con, các bài toán con này được giải và lời giải được ghi nhớ để phòng trường hợp cần dùng lại chúng, và các bài toán con này tiếp tục gọi đến bài toán con của chúng đến khi đủ dữ kiện để lấy ra kết quả tính toán.

****

* ***Nhận xét:*** Mỗi bài toán con chỉ được giải đúng 1 lần
  + ***Độ phức tạp: O(n)***
  1. **QHĐ sử dụng cách tiếp cận *bottom-up (từ dưới lên)***

Tất cả các bài toán con có thể cần đến đều được giải trước, sau đó được dùng để xây dựng lời giải cho các bài toán lớn hơn. Cách tiếp cận này tốt hơn về không gian bộ nhớ dùng cho ngăn xếp và số lời gọi hàm. Tuy nhiên, đôi khi việc xác định tất cả các bài toán con cần thiết cho việc giải quyết bài toán cho trước không được trực quan như Top-down.



* ***Nhận xét:*** Tính sẵn f[0], f[1], từ đó tính tiếp f[2], lại tính được f[3], f[4], ..., f[n].Đảm bảo mỗi giá trị fibonacci chỉ phải tính 1 lần

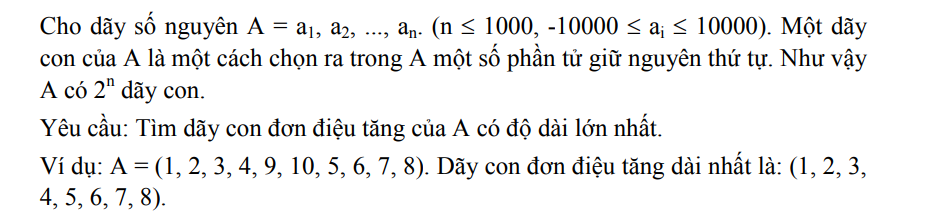
1. **Giải quyết bài toán QHĐ**

* **Những bước cơ bản**

Để giải quyết 1 bài toán bằng phương pháp quy hoạch động, chúng ta cần tiến hành những công việc sau:

* Tìm ra nghiệm của các bài toán con nhỏ nhất
* Tìm ra công thức (hoặc quy tắc) xây dựng nghiệm của bài toán con thông qua nghiệm của các bài toán con cỡ nhỏ hơn.
* Tạo ra một bảng lưu giữ các nghiệm của các bài toán con. Sau đó tính nghiệm của các bài toán con theo công thức đã tìm ra và lưu vào bảng.
* Từ các bài toán con đã giải để tìm nghiệm của bài toán.

1. **Bài toán QHĐ kinh điển**
2. **Dãy con tăng dài nhất**
   1. **Đề bài**

****

* 1. **Ý tưởng**

Bổ sung vào A hai phần tử: a0 = -∞ và an+1 = +∞. *Khi đó dãy con đơn điệu tăng dài nhất chắc chắn sẽ bắt đầu từ a0 và kết thúc ở an+1.*

Với ∀ i: 0 ≤ i ≤ n + 1. Ta sẽ tính L[i] = độ dài dãy con đơn điệu tăng dài nhất bắt đầu tại ai.

1. **Bài toán nhỏ nhất**

L[n + 1] = Độ dài dãy con đơn điệu tăng dài nhất bắt đầu tại an+1 = +∞. Dãy con này chỉ gồm mỗi một phần tử (+∞) nên L[n + 1] = 1

1. **Công thức**

* Giả sử với i từ n đến 0, ta cần tính L[i]: độ dài dãy con tăng dài nhất bắt đầu tại ai. L[i] được tính trong điều kiện L[i + 1], L[i + 2], ..., L[n + 1] đã biết:
* Dãy con đơn điệu tăng dài nhất bắt đầu từ ai sẽ được thành lập bằng cách lấy ai ghép vào đầu một trong số những dãy con đơn điệu tăng dài nhất bắt đầu tại vị trí aj đứng sau ai.
* Ta sẽ ghép ai vào đầu những dãy con bắt đầu tại aj nào đó lớn hơn ai (để đảm bảo tính tăng) và dĩ nhiên ta sẽ chọn dãy dài nhất để ghép ai vào đầu (để đảm bảo tính dài nhất).

**🡪** Vậy L[i] được tính như sau:

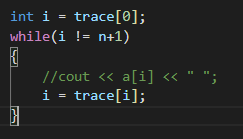
*Xét tất cả các chỉ số j trong khoảng từ i + 1 đến n + 1 mà aj > ai, chọn ra chỉ số jMax có L[jMax] lớn nhất. Đặt L[i] = L[jMax] + 1.*

1. **Truy vết**

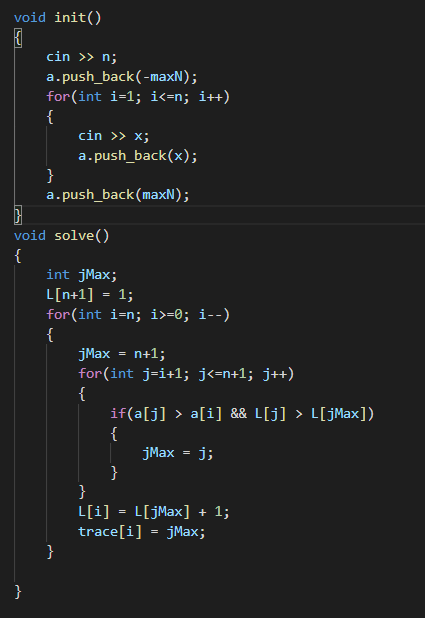
Tại bước xây dựng dãy L, mỗi khi tính L[i] = L[jmax] + 1, ta đặt T[i]

jMax để lưu lại rằng: Dãy con dài nhất bắt đầu tại ai sẽ có phần tử thứ hai kế tiếp là ajMax. Sau khi tính xong hay dãy L và T, ta bắt đầu từ 0.

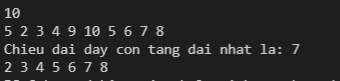
* T[0] là phần tử đầu tiên được chọn,
* T[T[0]] là phần tử thứ hai được chọn
* T[T[T[0]]] là phần tử thứ ba được chọn...
  + Quá trình truy vết có thể diễn tnhư sau:

****

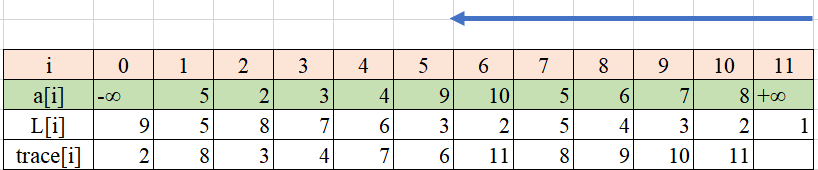
* 1. **Cài đặt**

****

* **Kết quả:**

****

* ***Kiểm nghiệm bài toán***

****

1. **Bài toán cái túi**
   1. **Đề bài**

Trong siêu thị có n gói hàng (n ≤ 100), gói hàng thứ i có trọng lượng là Wi ≤ 100 và trị giá Vi ≤ 100. Một tên trộm đột nhập vào siêu thị, sức của tên trộm không thể mang được trọng lượng vượt quá M (M ≤ 100). Hỏi tên trộm sẽ lấy đi những gói hàng nào để được tổng giá trị lớn nhất.

* 1. **Ý tưởng**

Nếu gọi B[i, j] là giá trị lớn nhất có thể có bằng cách chọn trong các gói {1, 2, ..., i} với giới hạn trọng lượng j. Thì giá trị lớn nhất khi được chọn trong số n gói với giới hạn trọng lượng M chính là B[n, M].

* + ***Nhận xét:***  Giá trị của của cái túi phụ thuộc vào 2 yếu tố: có bao nhiêu gói hàng đang được xét và trọng lượng của các gói hàng. Do đó bảng phương án sẽ là bảng 2 chiều - B[i, j]. Chú ý rằng khi xét đến B[i, j] thì các giá trị trên bảng phương án đều đã được tối ưu.

1. **Bài toán nhỏ nhất**

Dễ thấy B[0, j] = giá trị lớn nhất có thể bằng cách chọn trong số 0 gói = 0.

* + B[0, j] = 0.

1. **Công thức tính B[i, j]**

Với giới hạn trọng lượng j, việc chọn tối ưu trong số các gói {1, 2, ..., i - 1, i} để có giá trị lớn nhất sẽ có hai khả năng:

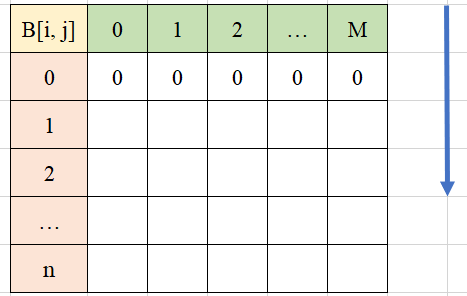
• Nếu không chọn gói thứ i thì B[i, j] là giá trị lớn nhất có thể bằng cách chọn trong số các gói {1, 2, ..., i - 1} với giới hạn trọng lượng là j. Tức là B[i, j] = B[i - 1, j]

• Nếu có chọn gói thứ i (tất nhiên chỉ xét tới trường hợp này khi mà Wi ≤ j) thì B[i, j] bằng giá trị gói thứ i là Vi cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các gói {1, 2, ..., i - 1} với giới hạn trọng lượng j - Wi .

* + Tức là về mặt giá trị thu được: B[i, j] = Vi + B[i - 1, j - Wi]

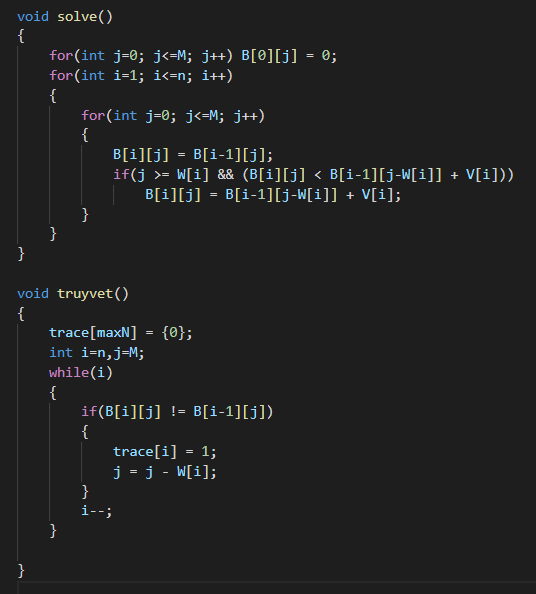
Vì theo cách xây dựng B[i, j] là giá trị lớn nhất có thể nên nó sẽ là max trong hai giá trị thu được ở trên.

1. **Bảng phương án**

****

1. **Truy vết**

* Bài toán cái túi thực hiện truy vết dựa vào tính chất ta xác định xem phần tử (i, j) được cập nhật theo phần tử (i-1, j’) nào trước đó.
* Giá trị lớn nhất thu được khi chọn trong cả n gói với giới hạn trọng lượng M là B[n, M]
* Nếu B[n, M] = b[n - 1, M] thì tức là không chọn gói thứ n, ta truy tiếp b[n - 1, M].
* Nếu B[n, M] ≠ b[n - 1, M] thì ta thông báo rằng phép chọn tối ưu có chọn gói thứ n và truy tiếp B[n - 1, M - Wn].
* Cứ tiếp tục cho tới khi truy lên tới hàng 0 của bảng phương án.
  1. **Cài đặt**

****

