Tên bài giảng

Phương pháp phân tích thiết kế quy hoạch động

Môn học: **Phân tích thiết kế thuật toán**

Chương: 6

Hệ: Đại học

Giảng viên: TS. Phạm Đình Phong

Email: phongpd@utc.edu.vn

Nội dung bài học

1. **Phân tích thiết kế, đánh giá thuật toán quy hoạch động**
2. **Một số thuật toán quy hoạch động trên dữ liệu một chiều**
3. **Một số thuật toán quy hoạch động trên dữ liệu nhiều chiều**

Phương pháp quy hoạch động

- Giới thiệu
  - Tên gọi "*Quy hoạch động – Dynamic Programming*" được Richard Bellman đề xuất vào những năm 1940 để mô tả một phương pháp giải các bài toán mà người giải cần đưa ra lần lượt một loạt các quyết định tối ưu
  - Tới năm 1953, Bellman chỉnh lại tên gọi này theo nghĩa mới và đưa ra nguyên lý giải quyết các bài toán bằng phương pháp quy hoạch động
  - Không có một thuật toán tổng quát để giải tất cả các bài toán quy hoạch động

Phương pháp quy hoạch động

- Một số thuật ngữ
  - Bài toán giải theo phương pháp quy hoạch động được 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 nhỏ nhất có ngay lời giải để từ đó giải quyết các bài toán lớn hơn gọi là *cơ sở của quy hoạch động*
  - Không gian lưu trữ lời giải các bài toán con để tìm cách phối hợp chúng gọi là *bảng quy hoạch* hay *bảng phương án* của quy hoạch động

Phương pháp quy hoạch động

- Ba tính chất
  - Bài toán lớn có thể được phân rã thành những bài toán con đồng dạng, những bài toán con đó có thể phân rã thành những bài toán nhỏ hơn nữa … (recursive form)
  - Lời giải tối ưu của các bài toán con có thể được sử dụng để tìm ra lời giải tối ưu của bài toán lớn (optimal substructure)
  - Hai bài toán con trong quá trình phân rã có thể có chung một số bài toán con khác (overlapping subproblems).

Phương pháp quy hoạch động

- Ba tính chất
  - Tính chất thứ nhất và thứ hai là điều kiện cần của một bài toán quy hoạch động.
  - Tính chất thứ ba nêu lên đặc điểm của một bài toán mà cách giải bằng phương pháp quy hoạch động hiệu quả hơn hẳn so với phương pháp giải đệ quy thông thường
  - Khác với thuật toán đệ quy, phương pháp quy hoạch động thêm vào cơ chế lưu trữ nghiệm hay một phần nghiệm của mỗi bài toán khi giải xong nhằm mục đích sử dụng lại, hạn chế những thao tác thừa trong quá trình tính toán

Phương pháp quy hoạch động

- Mục đích
  - Cải tiến thuật toán chia để trị hoặc quay lui vét cạn để giảm thời gian thực hiện
- Ý tưởng
  - Lưu trữ các kết quả của các bài toán con trong **bảng quy hoạch**
  - Đổi **bộ nhớ** lấy **thời gian**

Phương pháp quy hoạch động

- Kỹ thuật thiết kế
  - Phân tích bài toán dùng kỹ thuật chia để trị/quay lui
    - Chia bài toán thành các bài toán con
    - Xác định cơ sở của quy hoạch động
    - Tìm quan hệ giữa kết quả của bài toán lớn và kết quả của các bài toán con → công thức truy hồi
  - Lập bảng quy hoạch

Quy hoạch động = Chia để trị +
Cơ chế lưu trữ nghiệm để sử dụng lại

Phương pháp quy hoạch động

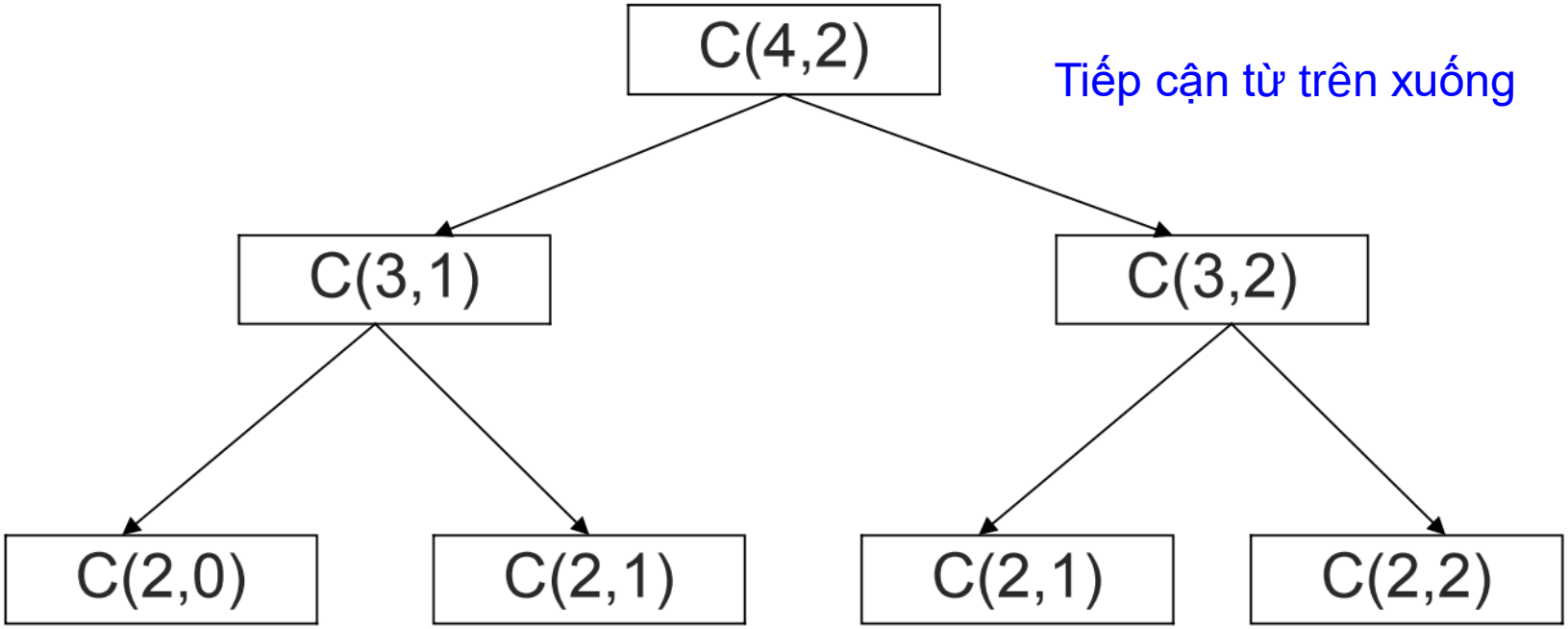
- Lập bảng quy hoạch
  - Số chiều = số biến trong công thức truy hồi
  - Xuất phát từ cơ sở của quy hoạch động →Thiết lập quy tắc điền kết quả vào bảng quy hoạch theo công thức truy hồi
  - Tra bảng tìm kết quả
  - Lần vết trên bảng để tìm lời giải tối ưu

Phương pháp quy hoạch động

- Một số điểm chú ý
  - Không có một "thuật toán" nào cho chúng ta biết một bài toán có thể giải bằng phương pháp quy hoạch động hay không?
  - Khi gặp một bài toán cụ thể, chúng ta phải phân tích bài toán, sau đó kiểm chứng ba tính chất của một bài toán quy hoạch động trước khi tìm lời giải bằng phương pháp quy hoạch động

Tính tổ hợp

- C(n, k) = 1 nếu (n = k) hoặc k = 0
- C(n, k) = C(n-1, k-1) + C(n-1, k)



Tiếp cận từ trên xuống

Tính tổ hợp

- C(n, k) = 1 nếu (n = k) hoặc k = 0
- C(n, k) = C(n-1, k-1) + C(n-1, k)
- **Đệ quy**

```
int comb(int n, int k) {
    if((k == 0) || (k == n))
        return 1;
    else
        return comb(n-1, k-1) + comb(n-1, k);
}
```

Tính tổ hợp

- **Độ phức tạp của thuật toán đệ quy**
  - $T(n)$ là thời gian để tính số tổ hợp chập k của n thì ta có phương trình đệ quy:

    $T(1) = C1$

    $T(n) = 2T(n\text{-}1) + C2$

    → Độ phức tạp quá lớn: $T(n) = O(2^n)$

Tính tổ hợp

- **Quy hoạch động: $T(n) = O(n^2)$**

k

C	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

n

Tính tổ hợp

- **Quy hoạch động: $T(n) = O(n^2)$**

```
int comb(int n, int k) {
    int c[n][n], i, j;
    c[0][0] = 1;
    for(i = 1; i<=n; i++) {
        c[i][0] = 1; c[i][i] = 1;
        for(j=1; j<i; j++)
            c[i][j] = c[i-1][j-1] + c[i-1][j];
    }
    return c[n][k];
}
```

Tính tổ hợp

- **Quy hoạch động: $T(n) = O(n^2)$**

```
int comb(int n, int k) {
    int c[n], i, j, p1, p2;
    c[0] = 1; c[1] = 1;
    for (i = 2; i<=n; i++) {
        p1 = c[0];
        for (j=1; j<i; j++) {
            p2 = c[j]; c[j] = p1 + p2; p1 = p2;
        }
        c[i] = 1; //j = i
    }
    return c[k];
}
```

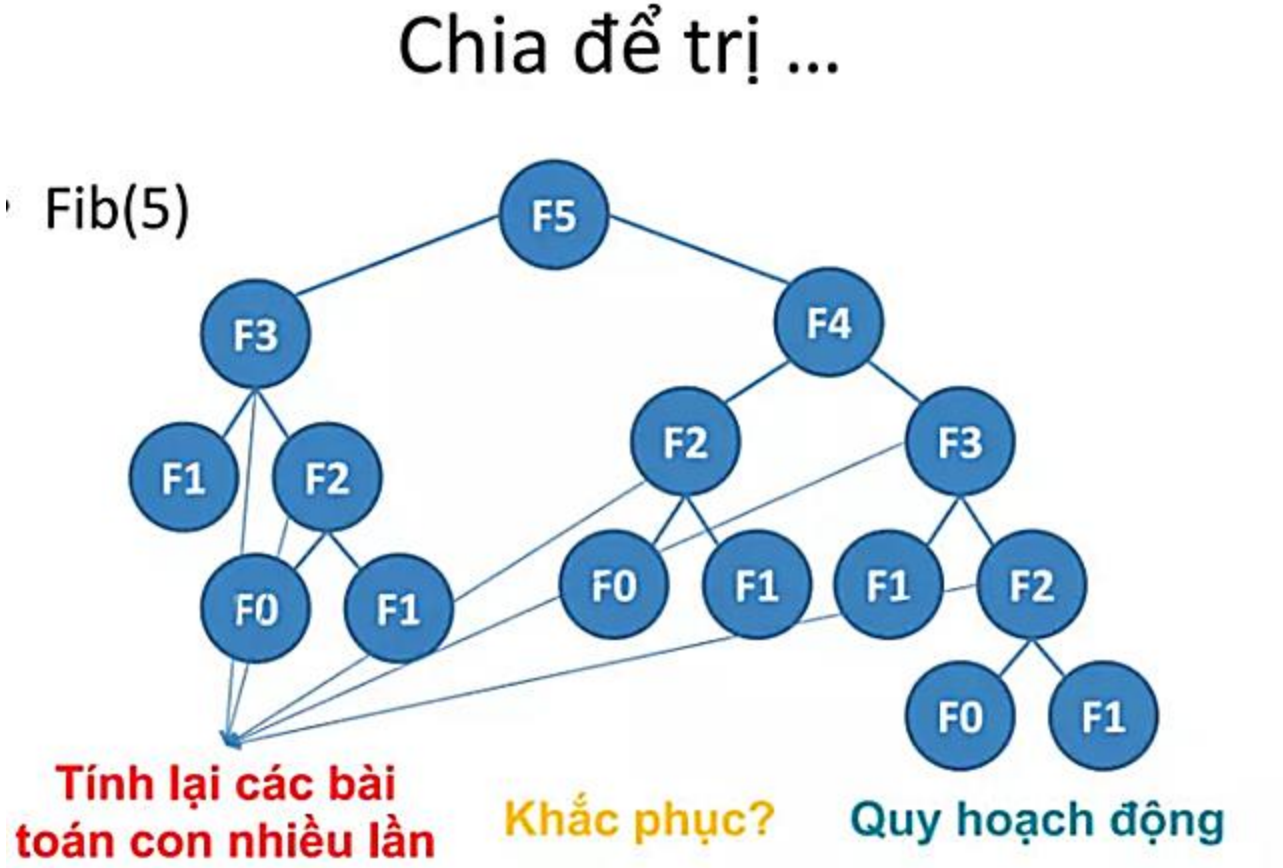
Dãy Fibonacci

- Phương pháp đệ quy

```
int fibonacci (int n) {
        if (n < 2)
            return n;
        return fibonacci(n-1) + fibonacci(n-2);
}
```

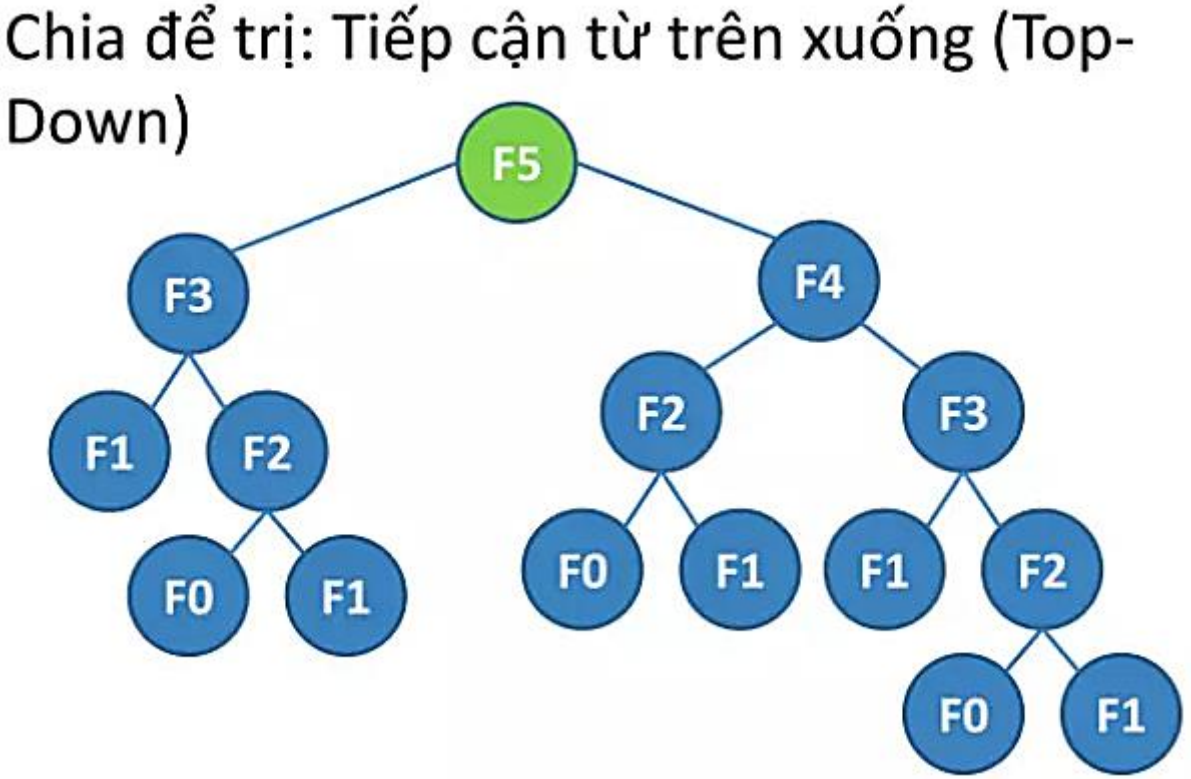
Dãy Fibonacci

- Phương pháp đệ quy



Dãy Fibonacci

- Phương pháp đệ quy



Dãy Fibonacci

- Phương pháp đệ quy

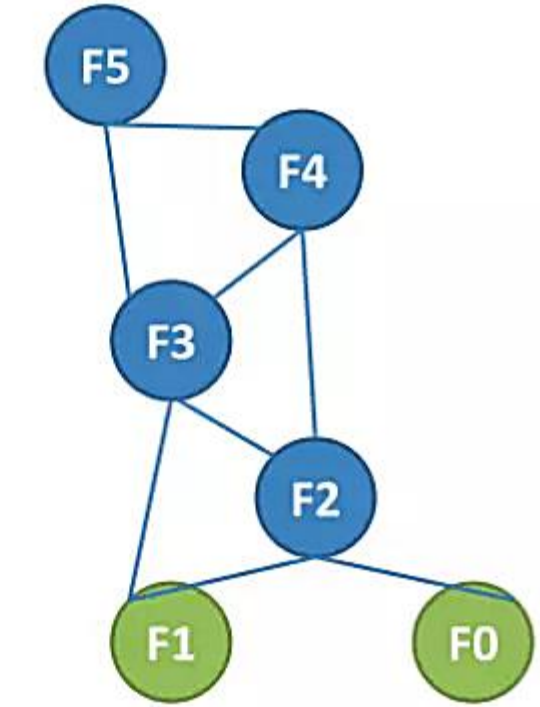
```
int fibonacci (int n) {
        if (n < 2)
            return n;
        return fibonacci(n-1) + fibonacci(n-2);
}
```

  - Độ phức tạp: $\Omega(\Phi^n)$, trong đó $\Phi = (1+\sqrt{5})/2$ là tỉ lệ vàng

Dãy Fibonacci

- Phương pháp quy hoạch động

Quy hoạch động: Tiếp cận từ dưới lên (Bottom-up)



Dãy Fibonacci

- Phương pháp quy hoạch động

```
int fibonacci (int F[], int n) {
    F[0] = 0;
    F[1] = 1;
    for (int i = 2; i<=n; i++)
        F[i] = F[i-1] + F[i-2];
  return F[n];
}
```

  - Độ phức tạp: $O(n)$

Dãy Fibonacci

- Phương pháp quy hoạch động

Phân rã:

F(n) = **F(n-1) + F(n-2)**

Giải bài toán con

F(0) = 0

F(1) = 1

Tổng hợp

F(n) = F(n-1) + F(n-2)

  - Độ phức tạp: $O(n)$

Dãy Fibonacci

- Phương pháp quy hoạch động
  - Từ định nghĩa ta thấy F_n chỉ phụ thuộc vào F_{n-1} và F_{n-2}, chính vì thế ở mỗi bước lặp chúng ta chỉ cần lưu lại hai giá trị này là đủ

```
int fibonacci(int n) {
    if (n < 2) return n;
    int n1 = 1, n2 = 0; // lưu cơ sở vào các biến
    for (int i = 2; i < n; i++) {
        int n0 = n1 + n2;
        n2 = n1;
        n1 = n0;
    }
    return n1 + n2;
}
```

Bài toán tổng tập con (Subset Sum)

- Mô tả bài toán
  - Cho một mảng *n* phần tử *X*[1, …, *n*] không âm và một số *T*. Tồn tại tập con của *X* có tổng là *T?*
  - **Ví dụ 1**: *X* = {8, 6, 7, 5, 3, 10, 9} và *T* = 12

    → tồn tại tập con {7, 5} của *X* có tổng là 12
  - **Ví dụ 2**: *X* = {3, 34, 4, 12, 5, 2} và *T* = 9

    → tồn tại tập con {4, 5} của *X* có tổng là 9

Bài toán tổng tập con (Subset Sum)

- Phương pháp
  - Xét một phần tử $x \in X$, tồn tại một dãy con có tổng bằng T nếu một trong hai điều kiện sau là đúng:
    1. Tồn tại một tập con của $X \backslash \{x\}$ có tổng bằng $T-x$
    2. Tồn tại một tập con của $X \backslash \{x\}$ có tổng bằng T

Bài toán tổng tập con (Subset Sum)

- Phương pháp
  - Chọn bài con:
    - Xác định xem trong tập con có *i* phần tử X_i = {*X*[1], *X*[2], …, *X*[i]} của *X* tồn tại một tập con (của tập X_i) có tổng bằng *t* hay không?
    - Ta kí hiệu *S*[i,t]=*True* nếu câu trả lời là có tồn tại và *False* nếu ngược lại. Như vậy ta có công thức truy hồi:

$$S[i,t] = \begin{cases} True, & \text{if } t = 0 \\ False, & \text{if } t < 0 \text{ or } i < 0 \\ S[i-1, t-X[i]] \vee S[i-1,t], & \text{otherwise} \end{cases}$$

Bài toán tổng tập con (Subset Sum)

- Phương pháp
  - Ví dụ: $X = \{3, 4, 5, 2\}$, $T = 6$

	0	1	2	3	4	5	6
0	T	F	F	F	F	F	F
3	T	F	F	T	F	F	F
4	T	F	F	T	T	F	F
5	T	F	F	T	T	T	F
2	T	F	T	T	T	T	T

Bài toán tổng tập con (Subset Sum)

- Độ phức tạp
  - Phương pháp quay lui: $T(n) = O(2^n)$
  - Phương pháp quy hoạch động: $T(n) = O(nT)$

Bài toán cái ba lô

- Mô tả bài toán
  - Cho một cái ba lô có thể đựng một trọng lượng *W* và *n* loại đồ vật, mỗi đồ vật *i* có một trọng lượng *g*[i] và một giá trị *v*[i]. Tất cả các loại đồ vật đều có số lượng không hạn chế. Tìm một cách lựa chọn các đồ vật đựng vào ba lô, chọn các loại đồ vật nào, mỗi loại lấy bao nhiêu sao cho tổng trọng lượng không vượt quá *W* và tổng giá trị là lớn nhất

Bài toán cái ba lô

- Công thức
  - Gọi L(i, j) là tổng giá trị lớn nhất khi chọn được i vật từ 1 đến i cho vào ba lô với tổng khối lượng không vượt quá j
  - L(n, W) sẽ là đáp số của bài toán (là giá trị lớn nhất có được nếu chọn n vật và tổng khối lượng không vượt quá W

Bài toán cái ba lô

- Công thức
  - Công thức tính L(i, t) như sau:
    - $L(i, 0) = 0$
    - $L(0, t) = 0$
    - $L(i, t) = L(i-1, t)$ nếu $t < g_i$
    - $L(i, t) = \max(L(i-1, t), L(i, t-g_i) + v_i)$ nếu $t \geq g_i$
    - Trong đó, $L(i-1, t)$ là giá trị có được nếu không đưa vật i vào ba lô, $L(i, t-g_i) + v_i$ là giá trị có được nếu chọn vật i.

Bài toán cái ba lô

- Thuật toán:
  - Sử dụng mảng hai chiều

```
L[i][t] = 0; //với mọi i và t
int xepDovat() {
    int t=0;
    for (int i = 1; i <= n; i++) {
        for (t = 0; t <= W; t++) {
            if (t < g[i]) L[i][t] = L[i-1][t];
            else L[i][t] = max(L[i-1][t], L[i][t-g[i]] + v[i]);
        }
    }
    return L[n][W];
}
```

Bài toán cái ba lô

- Thuật toán:
  - Sử dụng hai mảng một chiều

```
int xepDovat() {
        int t = 0;
        L[t] = 0; //Với mọi t
        for (int i = 0; i <= n; i++) {
            P = L;
            for (t = 0; t <= W; t++)
                if (t < g[i]) L[t] = P[t];
                else L[t] = max(P[t], L[t-g[i]] + v[i]);
        }
        return L[W];
}
```

Dãy con tăng dài nhất

- Bài toán
  - Cho một mảng n phần tử $A[1, 2, \ldots, n]$. Tìm một dãy dài nhất các chỉ số $1 \le i_1 < i_2 < \ldots < i_k \le n$ sao cho $A[i_1] \le A[i_2] \le \ldots \le A[i_k]$.
  - Ví dụ: $A[10]$ = 1, 6, 5, 4, 7, 8, 2, 9, 0, 10. Một trong số dãy con tăng dài nhất là 1, 6, 7, 8, 9, 10 với chiều dài 6.

Dãy con tăng dài nhất

- Bài toán dễ hơn
  - Tìm chiều dài của dãy con tăng dài nhất của $A[1, 2, \ldots, n]$.
  - Ví dụ: $A[10]$ = 1, 6, 5, 4, 7, 8, 2, 9, 0, 10. Số cần tìm là 6.
  - *Quá trình tìm chiều dài thông thường sẽ cho ta phương pháp tìm dãy con bằng cách truy vết.*

Dãy con tăng dài nhất

- Ghi chú
  - *Đối với hầu hết các thuật toán quy hoạch động ta thường chỉ cần tìm một đặc tính đơn giản của lời giải như chiều dài hoặc chi phí, sau đó lời giải có thể tìm được thông qua truy vết*

Dãy con tăng dài nhất

- Phân tích bài toán con
  - Bài toán con xuất phát từ việc xét mảng con $A[1, 2, \ldots, i]$, $i < N$
    - Giả sử giải bài toán gốc trên mảng con trên, ta sẽ tìm chiều dài của dãy con tăng dài nhất của mảng $A[1, 2, \ldots, i]$
    - Nếu đã tìm được chiều dài của dãy con tăng dài nhất của mảng con $A[1, 2, \ldots, i]$, làm thế nào có thể mở rộng lời giải cho mảng $A[1, 2, \ldots, i + 1]$?

      → khó có thể mở rộng nếu định nghĩa bài toán con một cách trực tiếp như vậy

Dãy con tăng dài nhất

- Phân tích bài toán con
  - Giả sử ngoài việc xác định được chiều dài của bài toán con, nếu bằng cách nào đó có thể lưu trữ được phần tử cuối cùng của dãy con tăng dài nhất, liệu ta có thể mở rộng ra lời giải cho mảng *A*[1,2,…,i+1] được không?

    → Có lẽ không vì một mảng có thể có nhiều dãy con tăng khác nhau

Dãy con tăng dài nhất

- Phân tích bài toán con
  - **Ví dụ**: Xét $n = 10$ và mảng A[10] = 1, 5, 2, 3, 6, 4, 8, 7, 9, 10 và i = 5. Tức mảng con $A[1, 2, \ldots, 5]$ = 1, 5, 2, 3, 6 và một dãy con tăng dài nhất là 1, 5, 6.
  - Nếu chỉ lưu phần tử cuối cùng của mảng con là 6 (và chiều dài 3) thì ta không mở rộng được lời giải cho mảng con $A[1, 2, \ldots, 6]$ vì $A[6] = 4 < 6$.
  - Tuy nhiên, $A[1, 2, \ldots, 5]$ có một dãy con khác là 1, 2, 3 cũng có chiều dài 3, và nếu ta lưu phần tử cuối cùng 3 thì ta có thể mở rộng ra tìm lời giải cho dãy con $A[1, 2, \ldots, 6]$ thành 1, 2, 3, 4 vì $A[6] = 4 > 3$.

Dãy con tăng dài nhất

- Phân tích bài toán con
  - **Ví dụ trên cho ta một gợi ý về bài toán con**:
  - Ngoài xác định chiều dài của bài toán con, nếu ta có thể lưu trữ được mọi phần tử cuối cùng của tất cả các dãy con dài nhất có thể có thì ta có thể mở rộng ra được lời giải cho mảng *A*[1, 2, …, i+1] bằng cách so sánh *A*[i+1] với từng phần tử kết thúc đó.
  - Nếu A[i+1] lớn hơn hoặc bằng bất kì phần tử kết thúc nào thì dãy con tăng trong A[1,2,…,i+1] sẽ có chiều dài lớn hơn (tăng thêm 1). Ngược lại, dãy con tăng dài nhất của A[1,…,i+1] và A[1,…,i] có cùng chiều dài.

Dãy con tăng dài nhất

- Phân tích bài toán con
  - Phân tích trên gợi ý cho ta cách "mã hoá" bài toán con như sau:
    - Gọi $L[i]$ với $i \geq j$ là **chiều dài** của dãy con tăng dài nhất của $A[1,2,\ldots,i]$ kết thúc bằng A[i]
    - Ban đầu, ta khởi tạo mảng $L[i]=1$ với mọi $i, 1 \leq i \leq n$ (khởi tạo 1 là vì $A[1, 2, \ldots, i]$ luôn có một dãy con có chiều dài 1 chỉ bao gồm A[i] và kết thúc tại A[i])
    - **Chú ý**: khi tìm lời giải của $A[1,2,\ldots,i+1]$, ta có thể giả sử lời giải của $A[1,\ldots,j]$ với mọi $j<i+1$ đều đã sẵn có trong bảng L.

Dãy con tăng dài nhất

- Phân tích bài toán con
  - Ví dụ: Xét n=10 và mảng A[10] = 1, 5, 2, 3, 6, 4, 8, 7, 9, 10 và i = 5. Ta sẽ có L[1] = 1, L[2] = 2, L[3] = 2, L[4] = 3, L[5] = 4.
  - Câu hỏi đặt ra là: L[6] có giá trị bao nhiêu?
    - Theo định nghĩa, L[i+1] chính là chiều dài của dãy con tăng dài nhất của A[1,2,…,i+1] kết thúc tại A[i+1] → trong dãy con tăng đó A[i+1] là lớn nhất.
      → chỉ cần tìm một chỉ số j < i+1 sao cho:
      (1) A[j] ≤ A[i+1] và (2) L[j] có giá trị lớn nhất trong số các chỉ số j thoả mãn (1). Khi đó, theo định nghĩa, L[i+1] = L[j]+1 vì A[i+1] ≥ A[j].

Dãy con tăng dài nhất

- Phân tích bài toán con
  - **Câu hỏi**: Giả sử đã điền được toàn bộ mảng L[1, 2, …, n] thì lời giải của bài toán là gì?
  - **Trả lời**: chiều dài của dãy con tăng dài nhất theo định nghĩa của mảng L là $\max_{1 \le i \le n} L[i]$.

Dãy con tăng dài nhất

- Thuật toán
  - Để tránh phải duyệt qua toàn bộ mảng L để tìm lời giải, ta áp dụng một mẹo nhỏ: thêm phần tử A[n+1] có giá trị cực lớn sao cho lớn hơn mọi phần tử của mảng A. Như vậy, dãy con tăng dài nhất sẽ luôn kết thúc tại A[n+1]

Dãy con tăng dài nhất

- Thuật toán

```
LongestIncreasingSubsequence(A[1, 2, ..., n]):
1.  for i ← 1 to n + 1
2.      L[i] ← 1
3.  A[n + 1] ← ∞
4.  for i ← 2 to n + 1
5.      for j ← 1 to i − 1
6.          if A[i] ≥ A[j]
7.              L[i] ← max(L[i], L[j] + 1)
8.  return L[n + 1] − 1
```

  - Độ phức tạp: $T(n) = O(n^2)$

Dãy con tăng dài nhất

- Ví dụ:
  - Xét *n* = 10 và mảng *A*[10] = 1, 5, 2, 3, 6, 4, 8, 7, 9, 10.
  - Ta sẽ có L[1] = 1, L[2] = 2, L[3] = 2, L[4] = 3, L[5] = 4, L[6] = 4, L[7] = 5, L[8] = 5, L[9] = 6, L[10] = 7
  - Hàm trả lại giá trị: L[11] = 8 – 1 = 7

Dãy con tăng dài nhất

- Truy vết lời giải
  - Dựa vào mảng L[1, 2, …, *n*] để truy vết tìm dãy con tăng dài nhất
  - Chiều dài của dãy con tăng dài nhất là $M = L[n+1]-1$
  - Vì L[i] là chiều dài của dãy con tăng lớn nhất của mảng A[1,2,…,i] kết thúc tại A[i], chỉ số i lớn nhất sao cho L[i] = M chính là chỉ số của phần tử cuối cùng của dãy con tăng lớn nhất của A[1,2,…,n]

Dãy con tăng dài nhất

- Truy vết lời giải

```
PrintLIS(A[1, 2, ..., n], L[1, ..., n]):
  m ← +∞
  for i ← n to 1
    if L[i] = M and A[i] ≤ m
      print A[i]
      m ← A[i]
      M ← M − 1
```

  - Độ phức tạp: $T(n) = O(n)$

TỔNG KẾT

Chia để trị

Ý tưởng:

- Phân rã thành các bài toán con
- Tổng hợp kết quả

Thuật toán:

- Đệ quy từ trên xuống
- Độ phức tạp thời gian lớn nếu có nhiều bài toán con giống nhau
- Không cần lưu trữ kết quả của tất cả các bài toán con

Quy hoạch động

Ý tưởng:

- Phân rã thành các bài toán con
- Tìm mối quan hệ

Thuật toán:

- Lập bảng quy hoạch và giải từ dưới lên
- Độ phức tạp thời gian nhỏ hơn nhờ sử dụng bảng quy hoạch
- Cần bộ nhớ để lưu bảng quy hoạch

TỔNG KẾT CHUNG

1. **Mỗi phương pháp/kỹ thuật chỉ phù hợp với một hoặc một số loại bài toán**
2. **Mỗi kỹ thuật đều có ưu và khuyết điểm, không có kỹ thuật nào là "trị bách bệnh"**
   - **Kỹ thuật vét cạn có độ phức tạp thời gian quá cao (lũy thừa) → Chỉ dùng khi *n* nhỏ hoặc khi không còn cách nào khác**
   - **Phương pháp tham lam thường cho nghiệm gần đúng trong thời gian đa thức**
   - **Quy hoạch động chỉ tốt khi số lượng bài toán con cần phải giải là đa thức (n, n^2 hoặc n^3)**

Bài tập

- Đổi tiền
  - Ở đất nước Omega người ta chỉ tiêu tiền xu. Có N loại tiền xu, loại thứ i có mệnh giá là A_i đồng. Một người khách du lịch đến Omega du lịch với số tiền M đồng. Ông ta muốn đổi số tiền đó ra tiền xu Omega để tiện tiêu dùng. Ông ta cũng muốn số đồng tiền đổi được là ít nhất (cho túi tiền đỡ nặng khi đi đây đi đó). Bạn hãy giúp ông ta tìm cách đổi tiền.

Bài tập

- Chia kẹo
  - Cho *n* gói kẹo, gói thứ i có a_i viên. Hãy chia các gói kẹo thành 2 phần sao cho chênh lệch giữa 2 phần là ít nhất.