

Nội dung

- Chương 1: Giới thiệu
- Chương 2: Độ quy và Giải thuật đệ quy
- **Chương 3: Quy hoạch động**
- Chương 4: Danh sách tuyến tính
- Chương 5: Stack và Queue
- Chương 6: Cây và Ứng dụng
- Chương 7: Đồ thị và Ứng dụng
- Chương 8: Các thuật toán sắp xếp
- Chương 9: Các thuật toán tìm kiếm

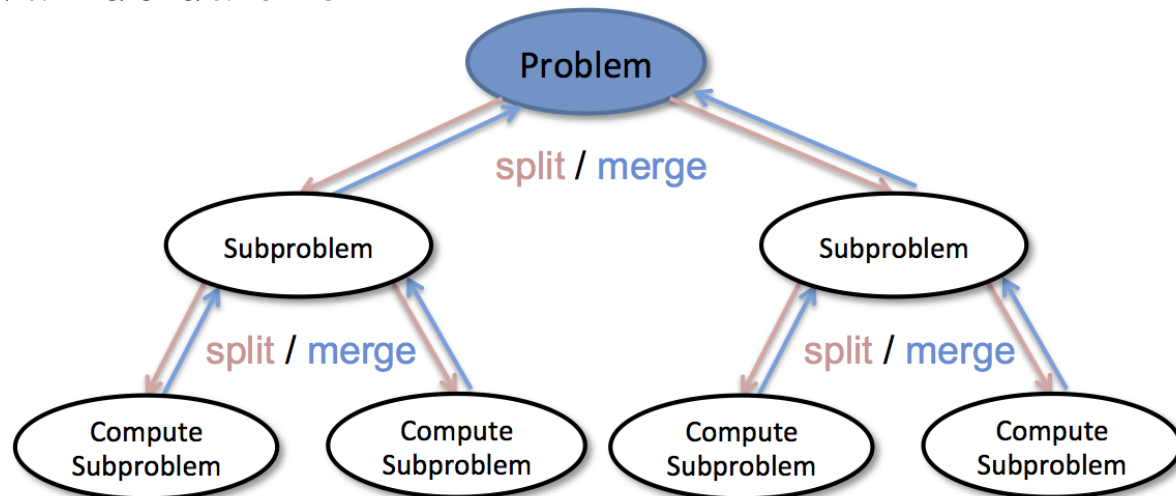
Các chiến lược thiết kế giải thuật

- Đứng trước một vấn đề đặt ra, chúng ta cần suy nghĩ về cách (chiến lược) giải quyết vấn đề đó
- Có nhiều chiến lược để thiết kế giải thuật
 - Chia để trị (divide and conquer)
 - Độ quy quay lui (backtracking)
 - Quy hoạch động (dynamic programming)
 - Tham lam (greedy strategy)
 - ...
- Mỗi chiến lược phù hợp cho một số dạng bài toán
- Mỗi chiến lược có ưu và nhược điểm riêng
- Nhiệm vụ: cần quyết định chọn chiến lược phù hợp

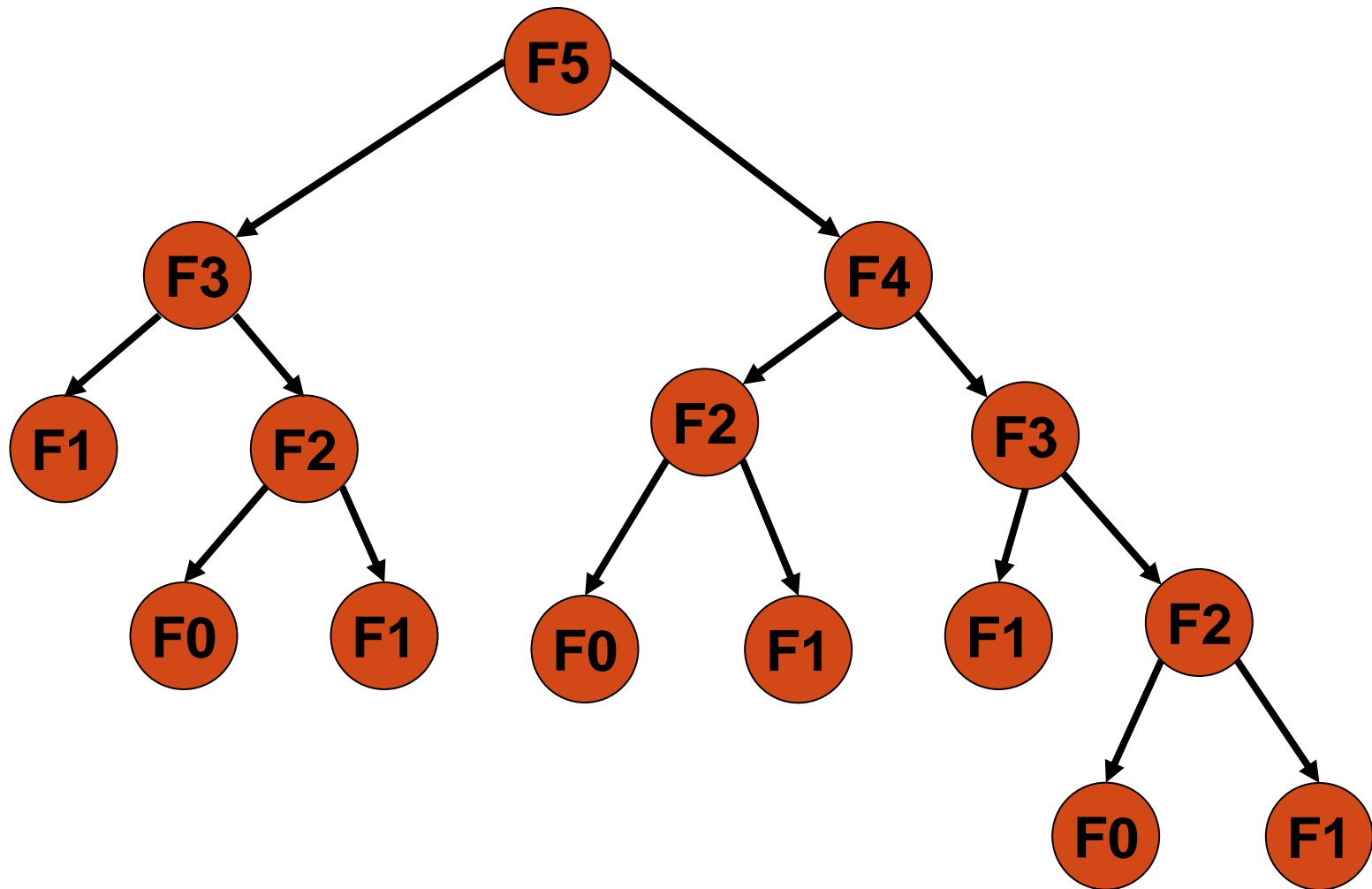
Chiến lược: Chia để trị

- Ý tưởng

- Chia vấn đề cần giải thành một số vấn đề con cùng dạng với vấn đề đã cho, chỉ khác là cỡ của chúng nhỏ hơn.
- Mỗi vấn đề con được giải quyết độc lập, nếu vấn đề con chưa tìm được nghiệm ngay thì lại tiếp tục chia nhỏ theo tư tưởng như trên
- Sau đó, ta kết hợp nghiệm của các vấn đề con để nhận được nghiệm của vấn đề đã cho



Ví dụ: Tìm số fibonacci



Chiến lược: Quy hoạch động

- QHĐ được dùng để giải bài toán tối ưu có bản chất đệ quy
- Lời giải tối ưu được đưa về việc tìm lời giải tối ưu của một số hữu hạn các bài toán con
- QHĐ gần giống với CĐT là đều chia bài toán lớn thành hữu hạn các bài toán con
- Nhưng khác nhau ở chỗ
 - CĐT: phân rã bài toán lớn thành nhiều bài toán con và đi giải chúng, việc giải các bài toán con này lại tiếp tục phân rã thành nhiều bài toán con nhỏ hơn, bất kể các bài toán con đó đã được giải hay chưa (top-down)
 - QHĐ: bắt đầu bằng việc giải tất cả các bài toán con nhỏ nhất, để từ đó giải được bài toán lớn hơn, và cứ thế cho khi giải được bài toán lớn nhất đặt ra (bottom-up)

Các giai đoạn giải bài toán theo QHĐ

- **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
- **Giải các bài toán con:** 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

Các bước giải bài toán theo QHĐ

- Giải các bài toán cơ sở (bài toán biên)
- Xây dựng công thức truy hồi để giải bài toán lớn hơn
- Dựa vào lời giải của bài toán cơ sở và công thức truy hồi để xây dựng bảng phương án
 - Bảng phương án được dùng để lưu trữ lời giải của các bài toán con để tránh phải giải lại các bài toán đã có lời giải
 - Tùy thuộc từng bài toán: bảng phương án có thể dùng mảng 1 chiều hoặc 2 chiều
- Dựa vào bảng phương án để lần vết tìm lời giải của bài toán ban đầu

Hiệu quả của QĐH

- Khi có các bài toán con lồng nhau, phương pháp CĐT sẽ tỏ ra không hiệu quả, khi nó phải lặp đi lặp lại việc giải các bài toán con chung đó.
- QĐH 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.
- Hạn chế của QĐH là kích thước của bảng phương án lưu trữ lời giải của các bài toán trung gian

Một số ví dụ

- Tìm dãy con đơn điệu tăng dài nhất
- Tìm xâu con chung dài nhất của hai xâu
- Bài toán cái túi
- Bài toán nhân ma trận
-

Tìm dãy con đơn điệu tăng dài nhất

- **Bài toán:** Cho một dãy số nguyên A gồm n phần tử, một dãy con trong A là một cách chọn ra một số phần tử giữ nguyên thứ tự. Hãy tìm dãy con đơn điệu tăng có độ dài lớn nhất?
- Ví dụ: $A = [1, 3, 6, 2, 7, 8, 5, 4, 9]$. Dãy con đơn điệu tăng dài nhất là: $[1, 3, 6, 7, 8, 9]$
- **Lời giải:** Để đơn giản trong cài đặt, bổ sung thêm vào dãy A 2 phần tử $A[0] = -\text{maxInt}$ và $A[n+1] = +\text{maxInt}$, như vậy dãy con đơn điệu tăng dài nhất sẽ bắt đầu từ $A[0]$ và kết thúc tại $A[n+1]$

Tìm dãy con đơn điệu tăng dài nhất

- Đặt $L[i]$ = độ dài của dãy con ĐĐTĐN bắt đầu tại $A[i]$.
- Cơ sở QHD: $L[n + 1] = 1$
- Công thức truy hồi:

$$L[i] = \max_{\substack{i+1 \leq j \leq n+1 \\ a[i] < a[j]}} L[j] + 1$$

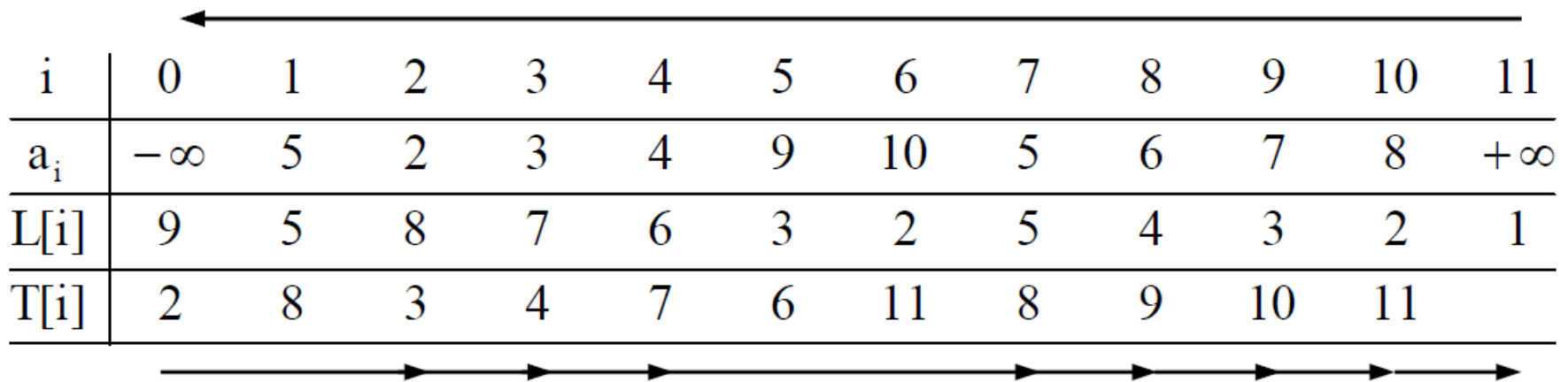
- Truy vết tìm dãy

$T[i]$ = vị trí trước $a[i]$ trong dãy ĐĐTĐN

- Sau khi tính được $L[.]$ và $T[.]$ thì lần vết từ $T[0]$ để tìm dãy con
- Trong đó: $T[0]$ là phần tử đầu tiên,
 $T[T[0]]$ là phần tử thứ 2, ...

Tìm dãy con đơn điệu tăng dài nhất

với $A = (5, 2, 3, 4, 9, 10, 5, 6, 7, 8)$. Hai dãy L và T sau khi tính sẽ là:

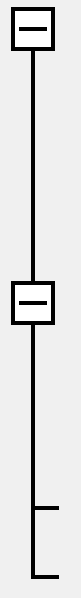


| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|-------|-----------|---|---|---|---|---|----|---|---|----|----|-----------|
| a_i | $-\infty$ | 5 | 2 | 3 | 4 | 9 | 10 | 5 | 6 | 7 | 8 | $+\infty$ |
| L[i] | 9 | 5 | 8 | 7 | 6 | 3 | 2 | 5 | 4 | 3 | 2 | 1 |
| T[i] | 2 | 8 | 3 | 4 | 7 | 6 | 11 | 8 | 9 | 10 | 11 | |

Xây dựng bảng phương án

```
void find() {  
    int i, j, jmax;  
  
    A[0] = -32000;  
    A[n+1] = 32000;  
    L[n+1] = 1;  
    for (i = n; i >= 0; i--) {  
        jmax = n + 1;  
        for (j = i + 1; j <= n; j++)  
            if ((A[i] < A[j]) && (L[j] > L[jmax])) jmax = j;  
        L[i] = L[jmax] + 1;  
        T[i] = jmax;  
    }  
}
```

Truy vết tìm kết quả

A vertical call stack diagram on the left side of the code block. It consists of a vertical line with two square boxes at the top and bottom, each containing a minus sign. A horizontal line connects the two boxes. A bracket on the right side of the line indicates the return path from the bottom box to the top box.

```
void result() {  
    int i, j;  
    cout << "Day con don dieu tang dai nhat la: ";  
    i = T[0];  
    while (i != (n+1)) {  
        cout << A[i] << "  ";  
        i = T[i];  
    }  
}
```

Bài toán ba lô

- Trong một cửa hàng có n gói hàng ($n \leq 100$), gói hàng thứ i nặng $W[i]$ và có giá trị $V[i]$. Một tên trộm mang một ba lô có thể chứa được trọng lượng tối đa là M , bạn cần chỉ cho tên trộm lấy các đồ vật nào để được tổng giá trị là lớn nhất?

- Ví dụ:

```
Cho n: 6
Cho M: 20
Nhap thong tin cac goi do:
Goi thu 1:
W[1]= 10
V[1]= 12
Goi thu 2:
W[2]= 5
V[2]= 14
Goi thu 3:
W[3]= 8
V[3]= 10
Goi thu 4:
W[4]= 4
V[4]= 12
Goi thu 5:
W[5]= 12
V[5]= 6
Goi thu 6:
W[6]= 8
V[6]= 15
Thong tin da nhap: <10,12> <5,14> <8,10> <4,12> <12,6> <8,15>
Tong gia tri an trom duoc la: 41
Cac vat da chon la: 6 4 2
```

Cách giải

- Gọi $F[i, j]$ là giá trị lớn nhất có thể có bằng cách chọn trong các gói $\{1, 2, \dots, i\}$ với giới hạn j . Thì chúng ta cần đi tìm $F[n, M] = ?$
- Công thức truy hồi để tính $F[i, j]$ như sau: với giới hạn trọng lượng j , việc chọn trong số các gói $\{1, 2, \dots, i\}$ để có giá trị lớn nhất sẽ có các khả năng:
 - Nếu KHÔNG chọn gói thứ i , thì giá trị lớn nhất có thể bằng cách chọn trong số các gói $\{1, 2, \dots, i-1\}$, nghĩa là: $F[i, j] = F[i-1, j]$
 - Nếu chọn gói i , thì $F[i, j]$ bằng giá trị gói thứ i là $V[i]$ cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các gói $\{1, 2, 3, \dots, i-1\}$ với giới hạn trọng lượng $j - W[i]$, nghĩa là:
$$F[i, j] = F[i-1, j - W[i]] + V[i]$$

Cách giải

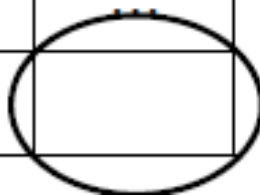

- Vì chúng ta cần tìm giá trị lớn nhất có thể, nên sẽ chọn giá trị lớn nhất trong hai giá trị đó, nghĩa là:

$$F[i, j] = \begin{cases} F[i - 1, j] & \text{nếu } W[i] > j \\ \max\{F[i - 1, j], F[i - 1, j - W[i]] + V[i]\} & \text{nếu } W[i] \leq j \end{cases}$$

- Cơ sở QHĐ: $F[0, j] = 0$
- Tính bảng phương án:
 - Dòng 0 toàn bằng 0
 - Dùng dòng 0 tính dòng 1
 - Dùng dòng 1 tính dòng 2
 -

Tính bảng phương án

| F | 0 | 1 | 2 | | M |
|-----|-----|-----|-----|---------|-----|
| 0 | 0 | 0 | 0 | ...0... | 0 |
| 1 | | | | | |
| 2 | | | | | |
| ... | ... | ... | ... | ... | ... |
| n | | | | | |



Truy vết tìm kết quả

- Giá trị lớn nhất là $F[n, M]$
- Để truy tìm được đã lấy những gói đồ nào ta làm như sau:
 - Nếu $F[n, M] = F[n - 1, M]$ thì nghĩa là đã không chọn gói n , truy tiếp $F[n - 1, M]$
 - Nếu $F[n, M] \neq F[n - 1, M]$ thì nghĩa là đã chọn gói n , truy tiếp $F[n - 1, M - W[n]]$
 - Cứ thế cho đến khi $n = 0$ thì dừng

Code

```
void find() {
    int i, j;

    for (j = 0; j <= M; j++) F[0][j] = 0; // co so QHD
    for (i = 1; i <= n; i++)
        for (j = 0; j <= M; j++) {
            F[i][j] = F[i-1][j]; // gia su khong chon vat thu i
            if ((j >= W[i]) && (F[i-1][j] < F[i-1][j-W[i]]+V[i]))
                F[i][j] = F[i-1][j-W[i]] + V[i];
        }
}

void result() {
    int i, j;
    cout << "Cac vat da chon la: ";
    while (n != 0) {
        if (F[n][M] != F[n-1][M]) {
            cout << n << " ";
            M = M - W[n];
        }
        n--;
    }
}
```