

PHÂN TÍCH THIẾT KẾ GIẢI THUẬT

Giảng viên phụ trách:
NGUYỄN THÁI DƯ

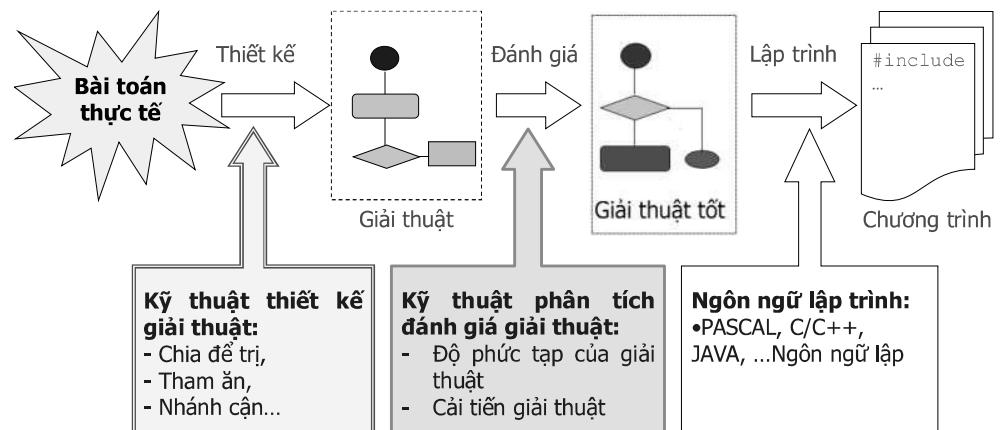
Mục tiêu

- ◆ Biết các kỹ thuật thiết kế giải thuật: từ ý tưởng cho đến giải thuật chi tiết.
- ◆ Hiểu rõ nguyên lý của các kỹ thuật phân tích thiết kế giải thuật
- ◆ Vận dụng kỹ thuật phân tích thiết kế để giải các bài toán thực tế: các bài toán dạng nào thì có thể áp dụng kỹ thuật này.

Chương 2. KỸ THUẬT THIẾT KẾ GIẢI THUẬT

- ◆ Mục tiêu
- ◆ Từ bài toán đến chương trình
- ◆ Các kỹ thuật thiết kế giải thuật
 - Chia để trị
 - Tham ăn (gready)
 - Quay lui
 - Vét cạn
 - Cắt tỉa Alpha-Beta
 - Nhánh cận

Mô hình từ bài toán đến chương trình



Kỹ thuật chia để trị

◆ Yêu cầu:

- Cần phải giải bài toán có kích thước n.

◆ Phương pháp:

- Ta chia bài toán ban đầu thành một số bài toán con đồng dạng với bài toán ban đầu có kích thước nhỏ hơn n.
- Giải các bài toán con được các lời giải con
- Tổng hợp lời giải con → ta có được lời giải của bài toán ban đầu.

◆ Chú ý

- Đối với từng bài toán con, ta lại chia chúng thành các bài toán con nhỏ hơn nữa.
- Quá trình phân chia này sẽ dừng lại khi kích thước bài toán **đủ nhỏ** mà ta có thể dễ dàng giải → gọi là **bài toán cơ sở**.

Nguyễn Thái Dư - AGU

5

Kỹ thuật chia để trị

```
solve(n) {  
    if (n đủ nhỏ để có thể giải được)  
        giải bài toán → KQ  
        return KQ;  
    else {  
        Chia bài toán thành các bài toán con  
        kích thước n1, n2, ...  
  
        KQ1 = solve(n1); //giải bài toán con 1  
        KQ2 = solve(n2); //giải bài toán con 2  
        ...  
        Tổng hợp các kết quả KQ1, KQ2, ... → KQ  
        return KQ;  
    }  
}
```

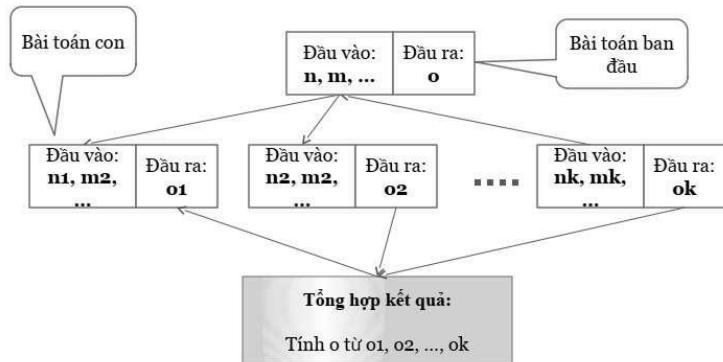
Nguyễn Thái Dư - AGU

7

Kỹ thuật chia để trị

◆ Kỹ thuật chia để trị bao gồm hai quá trình:

- Phân tích bài toán đã cho thành các bài toán cơ sở
- **Tổng hợp** kết quả từ bài toán cơ sở để có lời giải của bài toán ban đầu



Nguyễn ... - AGU

6

Nhìn lại giải thuật QuickSort và MergeSort

◆ Giải thuật QuickSort

- Sắp xếp dãy n số theo thứ tự tăng dần

◆ Áp dụng kỹ thuật chia để trị

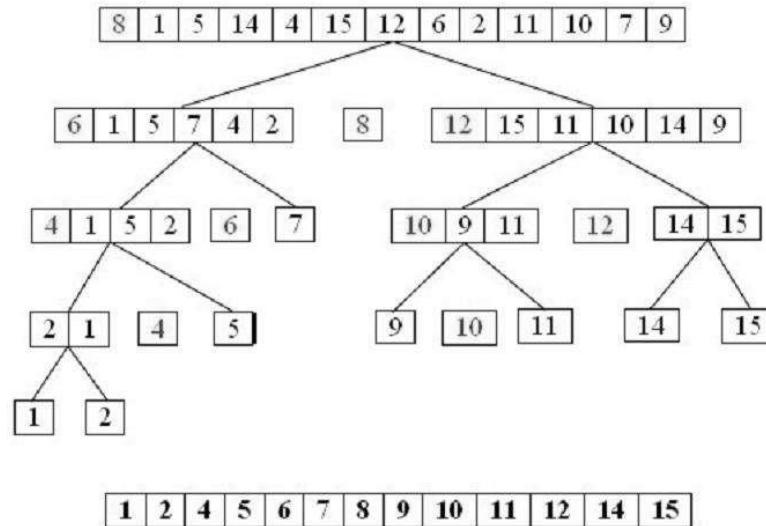
- **Phân chia:** Tìm một giá trị chốt và phân hoạch danh sách đã cho thành hai danh sách con “bên trái” và “bên phải”
 - Sắp xếp “bên trái” và “bên phải” thì ta được danh sách có thứ tự.
 - Bài toán cơ sở: Sắp xếp danh sách có 1 phần tử hoặc nhiều phần tử có giá trị giống nhau.
- **Tổng hợp:** đã bao hàm trong giai đoạn phân chia.

Nguyễn Thái Dư - AGU

8

Nhìn lại giải thuật QuickSort và MergeSort

◆ Ví dụ QuickSort:



Ngu

1 2 4 5 6 7 8 9 10 11 12 14 15

Độ phức tạp của QuickSort

◆ Xấu nhất

- Dãy n số đã đúng thứ tự tăng dần
- Phân hoạch bị lệch: phần tử chót là phần tử nhỏ nhất
=> cần n phép so sánh để biết nó là phần tử đầu tiên
- Độ phức tạp trong trường hợp này là: **O(n²)**

◆ Tốt nhất:

- Phân hoạch cân bằng: phần tử chót là phần tử giữa dãy
=> C(n) = 2C(n/2) + n
- Độ phức tạp trong trường hợp này là: **O(nlogn)**

Nhìn lại giải thuật QuickSort và MergeSort

◆ Giải thuật MergeSort

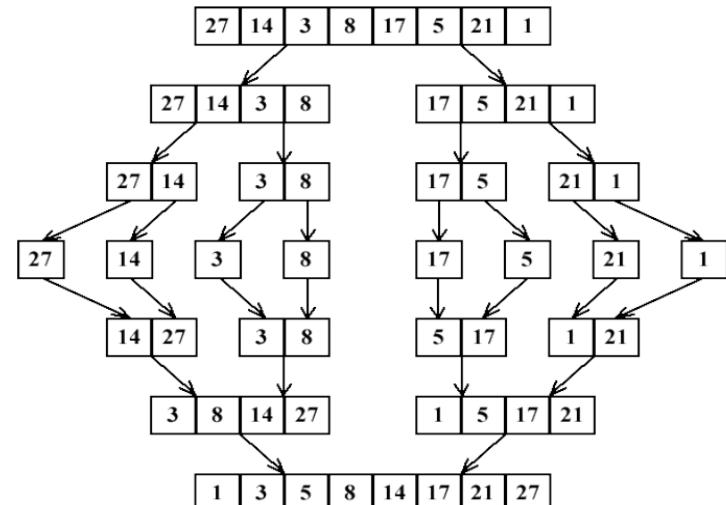
- Sắp xếp dãy n số theo thứ tự tăng dần

◆ Áp dụng kỹ thuật chia để trị

- Phân chia:** chia danh sách có n phần tử thành 2 danh sách có n/2 phần tử.
 - Quá trình phân chia sẽ dẫn đến các danh sách chỉ có 1 phần tử, là bài toán cơ sở.
- Tổng hợp:** trộn (merge) 2 danh sách có thứ tự thành một danh sách có thứ tự.

Nhìn lại giải thuật QuickSort và MergeSort

◆ Ví dụ Merge Sort:



Ngu

11

Ngu

12

Bài toán xếp lịch thi đấu thể thao

◆ Bài toán:

- Xếp lịch thi đấu vòng tròn 1 lượt cho n đấu thủ.
- Mỗi đấu thủ phải đấu với n-1 đấu thủ còn lại
- Mỗi đấu thủ chỉ đấu nhiều nhất là 1 trận mỗi ngày.

◆ Yêu cầu

- Xếp lịch sao cho số ngày thi đấu là ít nhất.

◆ Chia để trị

- chia
 - Xếp cho $n/2, n/4, n/8, \dots$
 - Cuối cùng xếp cho 2 đấu thủ
- Tổng hợp

Giải thuật chia để trị cho bài toán xếp lịch thi đấu

◆ Xuất phát từ bài toán cơ sở:

- Lịch thi đấu cho 2 đấu thủ 1 và 2 trong ngày thứ 1
- Như vậy ta có $O(1,1) = "2"$ và $O(2,1) = "1"$.

2 đấu thủ

1	1
2	2
2	1

Giải thuật chia để trị cho bài toán xếp lịch thi đấu

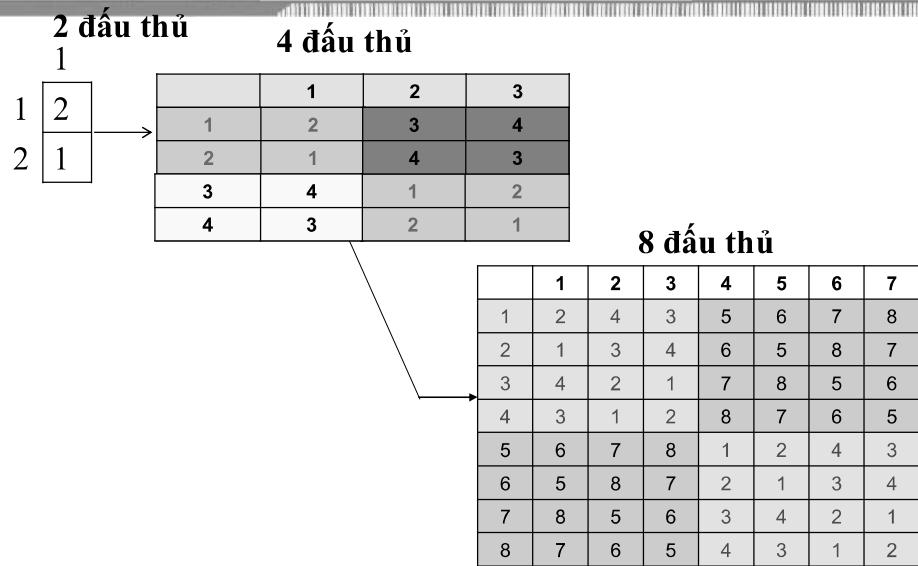
- ◆ Lịch thi đấu là 1 bảng gồm n dòng (tương ứng với n đấu thủ) và n-1 cột (tương ứng với n-1 ngày). Ô (i,j) biểu diễn đấu thủ mà i phải đấu trong ngày j.
- ◆ Chia để trị: thay vì xếp cho n người, ta sẽ xếp cho $n/2$ người sau đó dựa trên kết quả của lịch thi đấu của $n/2$ người ta xếp cho n người.
- ◆ Quá trình phân chia sẽ dừng lại khi ta phải xếp lịch cho 2 đấu thủ. Việc xếp lịch cho 2 đấu thủ rất dễ dàng: ta cho 2 đấu thủ này thi đấu 1 trận trong 1 ngày.
- ◆ Bước khó khăn nhất chính là bước xây dựng lịch cho 4, 8, 16, ... đấu thủ từ lịch thi đấu của 2 đấu thủ.

Giải thuật chia để trị cho bài toán xếp lịch thi đấu

- ◆ Xuất phát từ lịch thi đấu cho hai đấu thủ ta có thể xây dựng lịch thi đấu cho 4 đấu thủ như sau:

- Lịch thi đấu cho 4 đấu thủ sẽ là một bảng 4 dòng, 3 cột.
- Lịch thi đấu cho 2 đấu thủ 1 và 2 trong ngày thứ 1 chính là lịch thi đấu của hai đấu thủ (**bài toán cơ sở**).
- Như vậy ta có $O(1,1) = "2"$ và $O(2,1) = "1"$.
- Tương tự ta có lịch thi đấu cho 2 đấu thủ 3 và 4 trong ngày thứ 1. Nghĩa là $O(3,1) = "4"$ và $O(4,1) = "3"$.
- Bây giờ để hoàn thành lịch thi đấu cho 4 đấu thủ, ta lấy góc trên bên trái của bảng lấp vào cho góc dưới bên phải và lấy góc dưới bên trái lấp cho góc trên bên phải.

Xây dựng lịch thi đấu



Bài toán con cân bằng

- ◆ Sẽ tốt hơn nếu ta chia bài toán cần giải thành các bài toán con có kích thước gần bằng nhau.
- ◆ Ví dụ:
 - **MergeSort** phân chia bài toán thành hai bài toán con có cùng kích thước $n/2$ và do đó thời gian của nó chỉ là $O(n\log n)$.
 - Ngược lại trong trường hợp xấu nhất của **QuickSort**, khi mảng bị phân hoạch lệch thì thời gian thực hiện là $O(n^2)$.
- ◆ Nguyên tắc chung: Chia bài toán thành các bài toán con có kích thước xấp xỉ bằng nhau thì hiệu suất sẽ cao hơn.

Bài toán tối ưu tổ hợp

- Cho hàm $f(X)$, là hàm mục tiêu, xác định trên một tập hữu hạn các phần tử D .
- Mỗi phần tử $X \in D$ có dạng $X = (x_1, x_2, \dots, x_n)$ được gọi là một phương án.
- Cần tìm một phương án $X^* \in D$ sao cho $f(X^*)$ đạt min/max.
 - Phương án X^* như thế được gọi là phương án tối ưu.
- Có nhiều phương pháp để giải
- Phương pháp “vét cạn” cần một thời gian mű.

Kỹ thuật “tham ăn”/“háu ăn” (Greedy)

- ◆ Đây là một kỹ thuật được dùng nhiều để giải các bài toán tối ưu tổ hợp.
- ◆ Áp dụng kỹ thuật này tuy không cho chúng ta lời giải tối ưu nhưng sẽ cho một lời giải “tốt”; bù lại chúng ta được lợi khá nhiều về thời gian.

Kỹ thuật “tham ăn”/“háu ăn” (Greedy)

◆ Phương pháp Greedy:

- Giải bài toán tối ưu tổ hợp: xây dựng một phương án X.
- Phương án X được xây dựng bằng cách:
 - Sắp xếp các lựa chọn cho mỗi bước theo thứ tự nào đó “có lợi” (tăng dần hoặc giảm dần tùy theo cách lập luận)
 - Lựa chọn từng X_i cho đến khi đủ n thành phần
$$X = (x_1, x_2, \dots x_n)$$
 - Với mỗi X_i , ta sẽ chọn X_i tối ưu.
→ Với cách này thì có thể ở bước cuối cùng ta không còn gì để chọn mà phải chấp nhận một giá trị cuối cùng còn lại.

◆ Kỹ thuật Greedy: thường chọn một khả năng mà xem như tốt nhất tại lúc đó. → Tức là, giải thuật chọn một khả năng tối ưu cục bộ với hy vọng sẽ dẫn đến một lời giải tối ưu toàn cục.

Bài toán trả tiền của máy rút tiền tự động ATM

Ví dụ: Máy rút tiền ATM

- ◆ Gọi $X = (X_1, X_2, X_3, X_4)$ là một phương án trả tiền.
- X_1 là số tờ giấy bạc 100.000 đồng,
 - X_2 là số tờ giấy bạc 50.000 đồng,
 - X_3 là số tờ giấy bạc 20.000 đồng,
 - X_4 là số tờ giấy bạc 10.000 đồng.

→ Mục tiêu là $X_1 + X_2 + X_3 + X_4$ đạt nhỏ nhất với ràng buộc là:

$$X_1 * 100.000 + X_2 * 50.000 + X_3 * 20.000 + X_4 * 10.000 = n.$$

Bài toán trả tiền của máy rút tiền tự động ATM

Ví dụ: Máy rút tiền ATM

- Trong máy ATM, có sẵn các loại tiền có mệnh giá 100.000 đồng, 50.000 đồng, 20.000 đồng và 10.000 đồng.
- Giả sử mỗi loại tiền đều có số lượng không hạn chế.
- Khách hàng cần rút một số tiền n đồng (tính chẵn đến 10.000 đồng, tức là n chia hết cho 10.000).
- Hãy tìm một phương án trả tiền sao cho trả đủ n đồng và số tờ giấy bạc phải trả là ít nhất.

Bài toán trả tiền của máy rút tiền tự động ATM

◆ Ý tưởng:

- Để $(X_1 + X_2 + X_3 + X_4)$ nhỏ nhất thì các tờ giấy bạc mệnh giá lớn phải được chọn nhiều nhất.
- Trước hết ta chọn tối đa các tờ giấy bạc 100.000 đồng, nghĩa là X_1 là số nguyên lớn nhất sao cho $X_1 * 100.000 \leq n$. Tức là $X_1 = n \text{ DIV } 100.000$.
- Xác định số tiền cần rút còn lại là hiệu $n - X_1 * 100000$
- Chuyển sang chọn loại giấy bạc 50.000 đồng, và cứ tiếp tục như thế cho các loại mệnh giá khác...

Bài toán trả tiền của máy rút tiền tự động ATM

Ví dụ: Khách hàng cần rút 1.290.000 đồng

$n = 1.290.000$, phương án trả tiền như sau:

- $X_1 = 1.290.000 \text{ DIV } 100.000 = 12$
 - Số tiền còn lại là $1.290.000 - 12 * 100.000 = 90.000$
- $X_2 = 90.000 \text{ DIV } 50.000 = 1$
 - Số tiền còn lại là $90000 - 1 * 50000 = 40000$
- $X_3 = 40.000 \text{ DIV } 20.000 = 2$
 - Số tiền còn lại là $40.000 - 2 * 20.000 = 0$
- $X_4 = 0 \text{ DIV } 10.000 = 0$
- Vậy ta có $X = (12, 1, 2, 0)$ tức là máy ATM sẽ trả cho khách hàng 12 tờ 100.000 đồng, 1 tờ 50.000 đồng và 2 tờ 20.000 đồng.

Nguyễn Thái Dư - AGU

25

Mô hình hóa Bài toán cái ba lô

◆ Gọi $X=(X_1, X_2, \dots, X_n)$ với X_i là số nguyên không âm, là một phương án. X_i là số đồ vật thứ i .

◆ Cần tìm X sao cho:

- $X_1g_1 + X_2g_2 + \dots + X_ng_n \leq W$
- $F(X) = X_1v_1 + X_2v_2 + \dots + X_nv_n \rightarrow \text{Max}$

Nguyễn Thái Dư - AGU

27

Kỹ thuật tham ăn - Bài toán cái ba lô

Bài toán cái ba lô

- ◆ Cho một cái ba lô có thể đựng một trọng lượng W
- ◆ Có n loại đồ vật (tất cả các loại đồ vật đều có số lượng không hạn chế), mỗi đồ vật i có một
 - Trọng lượng g_i
 - Giá trị v_i .
- ◆ **Vấn đề:** 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.

Nguyễn Thái Dư - AGU

26

Bài toán cái ba lô

❖ Áp dụng kỹ thuật Greedy

1. Tính đơn giá cho các loại đồ vật.
2. Xét các loại đồ vật theo **thứ tự đơn giá từ lớn đến nhỏ** (giảm dần)
3. Với mỗi đồ vật được xét sẽ lấy một số lượng tối đa mà trọng lượng còn lại của ba lô cho phép.
4. Xác định trọng lượng còn lại của ba lô và quay lại bước 3 cho đến khi không còn có thể chọn được đồ vật nào nữa.

Nguyễn Thái Dư - AGU

28

Bài toán cái ba lô

◆ **Ví dụ:** Ta có một ba lô có trọng lượng là 37 và 4 loại đồ vật với trọng lượng và giá trị tương ứng được cho trong bảng bên dưới:

Loại đồ vật	Trọng lượng	Giá trị
A	15	30
B	10	25
C	2	2
D	4	6

Bài toán cái ba lô

Loại đồ vật	Trọng lượng	Giá trị	Đơn giá
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

❖ Theo bảng thứ tự ưu tiên là **B, A, D** và **C**:

- Vật B, chọn tối đa là 3. Mỗi vật B có trọng lượng là 10
→ Trọng lượng còn lại của ba lô là $37 - 3*10 = 7$.
- Vật A, Không chọn được vì trọng lượng vật A là 15 trong khi ba lô chỉ còn 7.
- Vật D, chọn được 1. → Trọng lượng còn lại của ba lô: $7-4 = 3$.
- Vật C, chọn được 1
- Tổng trọng lượng là: $3*10 + 0 + 1*4 + 1*2 = 36$
- Tổng giá trị là: $3*25 + 0 + 1*6 + 1*2 = 83$

Bài toán cái ba lô

◆ Từ bảng trên ta tính đơn giá và sắp lại theo đơn giá

Loại đồ vật	Trọng lượng	Giá trị	Đơn giá
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

Biến thể của bài toán cái ba lô

◆ Có một số biến thể của bài toán cái ba lô như sau:

- Mỗi đồ vật **i** chỉ có một số lượng s_i .
 - Với bài toán này khi lựa chọn vật **i** ta không được lấy một số lượng vượt quá s_i .
- Mỗi đồ vật chỉ có một cái.
 - Với bài toán này thì với mỗi đồ vật ta chỉ có thể chọn hoặc không chọn.

Kỹ thuật quay lui (backtracking)

- ◆ Kỹ thuật quay lui (backtracking) là một quá trình phân tích đi xuống và quay lui trở lại theo con đường đã đi qua.
- ◆ Tại mỗi bước phân tích chúng ta chưa giải quyết được vấn đề do còn thiếu dữ liệu nên cứ phải phân tích cho tới các điểm dừng, nơi chúng ta xác định được lời giải của chúng hoặc là xác định được là không thể (*hoặc không nên*) tiếp tục theo hướng này.
- ◆ Từ các điểm dừng này chúng ta quay ngược trở lại theo con đường mà chúng ta đã đi qua để giải quyết các vấn đề còn tồn đọng và cuối cùng ta sẽ giải quyết được vấn đề ban đầu.

Nguyễn Thái Dư - AGU

33

Kỹ thuật quay lui (backtracking)

- ◆ Để giải bài toán A ta cần giải các bài toán con A_1, \dots, A_n
- ◆ Một số bài toán con A_i chưa giải được ta phải đi giải các bài toán con $A_{i1}, A_{i2}, \dots, A_{ik}$
- ◆ Sau đó quay lại giải A_i
- ◆ Trong quá trình giải:
 - Giải tất cả các bài toán con \rightarrow vét cạn
 - một số bài toán con không cần giải ta bỏ qua \rightarrow cắt tỉa

Nguyễn Thái Dư - AGU

34

Kỹ thuật quay lui (backtracking)

- Giải bài toán tối ưu
 - Tìm một **lời giải tối ưu** trong số các lời giải
 - Mỗi **lời giải** gồm thành *n* **thành phần**.
 - Quá trình xây dựng một lời giải được xem như quá trình tìm *n* thành phần. Mỗi thành phần được tìm kiếm trong 1 bước.
 - Các **bước** phải có **dạng giống nhau**.
 - Ở mỗi bước, ta có thể dễ dàng chọn lựa một thành phần.
 - Sau khi thực hiện đủ *n* bước ta được 1 lời giải

Nguyễn Thái Dư - AGU

35

Kỹ thuật quay lui (backtracking)

- ◆ **2 kỹ thuật quay lui:**
 - “Vét cạn” (*brute force*) là kỹ thuật phải đi tới tất cả các điểm dừng rồi mới quay lui.
 - Tìm tất cả các lời giải
 - Độ phức tạp là thời gian lũy thừa
 - “Cắt tỉa Alpha-Beta” và “Nhánh-Cận” (*branch and bound*) là hai kỹ thuật cho phép không cần thiết phải đi tới tất cả các điểm dừng, mà chỉ cần đi đến một số điểm nào đó và dựa vào một số suy luận để có thể quay lui sớm.
 - Chỉ tìm lời giải có lợi
 - Cải tiến thời gian thực hiện

Nguyễn Thái Dư - AGU

36

Kỹ thuật quay lui - Kỹ thuật vét cạn

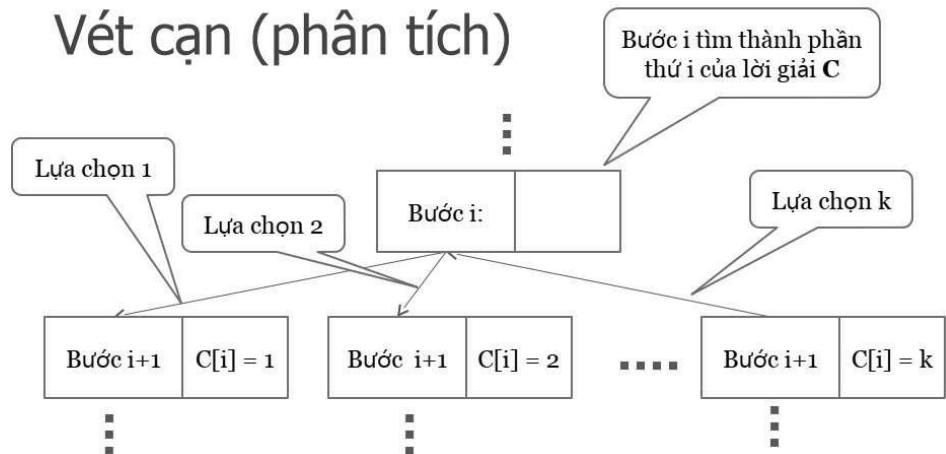
- **Ý tưởng:**
 - Gần giống chia để trị nhưng xây dựng lời giải từ dưới lên trong khi chia để trị là phân tích từ trên xuống
- **Một phương án/lời giải C:**
 - Gồm n thành phần $C[1], C[2], \dots, C[n]$
- **Ở mỗi bước i , có một số lựa chọn cho thành phần i .**
 - Chọn một giá trị nào đó cho thành phần i
 - Gọi đệ quy để tìm thành phần $i + 1$
 - Hủy bỏ sự lựa chọn, quay lui lại bước 1 chọn giá trị khác cho thành phần i
- **Chú ý:**
 - Quá trình đệ quy kết thúc khi $i > n$
 - Khi tìm được lời giải, so sánh với các lời trước đó để chọn lời giải tối ưu

Nguyễn Thái Dư - AGU

37

Kỹ thuật vét cạn

Vét cạn (phân tích)



Nguyễn Thái Dư - AGU

38

Kỹ thuật vét cạn

◆ Giải thuật vét cạn

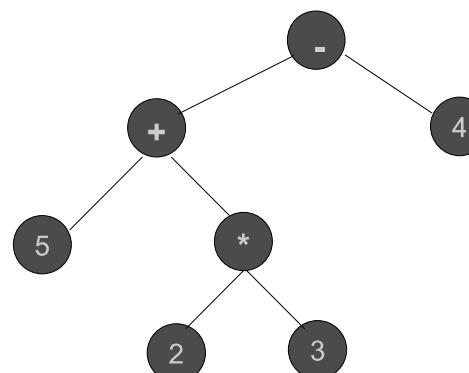
```
search(int i) {
    if (i > n)
        Kiểm tra, so sánh lời giải với các
        lời giải hiện có ➔ Lời giải tối ưu
    else {
        for (j ∈ lựa chọn có thể có của bước i) {
            C[i] = j; //Lựa chọn p/a j cho bước i
            search(i + 1); //Gọi đệ quy
            C[i] = null; //Hủy bỏ lựa chọn
        }
    }
}
```

Nguyễn Thái Dư - AGU

39

Kỹ thuật quay lui - Kỹ thuật vét cạn

- ◆ **Kỹ thuật vét cạn:** là kỹ thuật phải đi tới tất cả các điểm dừng rồi mới quay lui.
- ◆ **Ví dụ:** định trị cây biểu thức số học $5 + 2 * 3 - 4$



Nguyễn Thái Dư - AGU

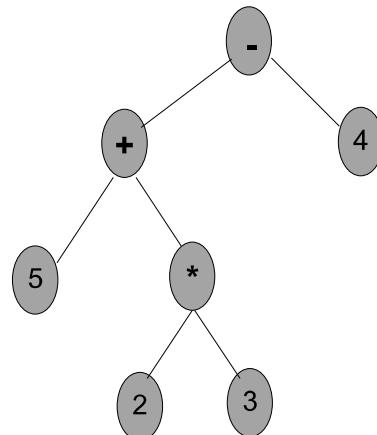
40

Kỹ thuật Vét cạn – Định vị cây biểu thức số học

- ◆ Trong ngôn ngữ lập trình đều có các biểu thức số học, việc dịch các biểu thức này đòi hỏi phải đánh giá (định trị) chúng.
- ◆ Để làm được điều đó cần phải có một biểu diễn trung gian cho biểu thức.
- ◆ Một trong các biểu diễn trung gian cho biểu thức là cây biểu thức.

Kỹ thuật Vét cạn – Định vị cây biểu thức số học

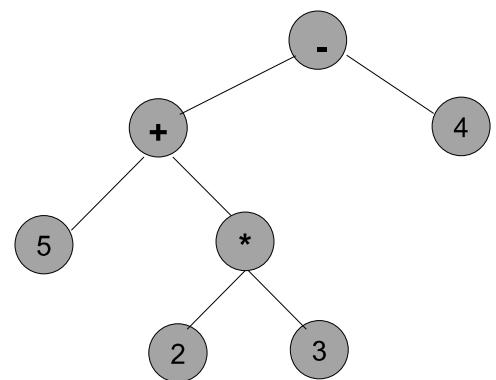
- ◆ Trị của nút lá chính là trị của toán hạng mà nút đó biểu diễn
- ◆ Trị của một nút trong có được bằng cách lấy toán tử mà nút đó biểu diễn áp dụng vào các con của nó.
- ◆ Trị của nút gốc chính là trị của biểu thức.



Kỹ thuật Vét cạn – Định vị cây biểu thức số học

- ◆ Cây biểu thức số học là một cây nhị phân, trong đó:
 - Các nút lá biểu diễn cho các toán hạng,
 - Các nút trong biểu diễn cho các toán tử.

- ◆ Biểu thức $5 + 2 * 3 - 4$ sẽ được biểu diễn bởi cây trong hình bên

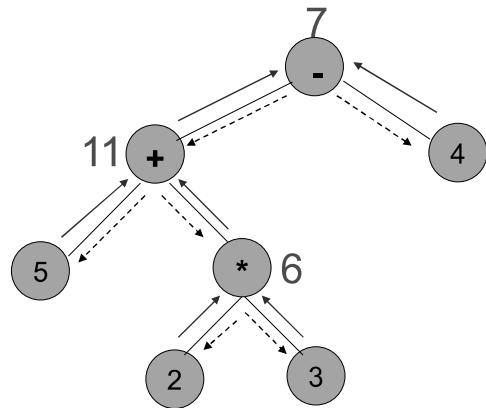


Kỹ thuật Vét cạn – Định vị cây biểu thức số học

- ◆ Định trị cho nút gốc bằng cách:
 - Định trị cho hai con của nó,
 - đối với mỗi con ta xem nó có phải là nút lá hay không, nếu không phải ta lại phải xét hai con của nút đó.
- ◆ Quá trình cứ tiếp tục như vậy cho tới khi gặp các nút lá mà giá trị của chúng đã được biết,
 - quay lui để định trị cho các nút cha của các nút lá
 - và cứ như thế mà định trị cho tổ tiên của chúng.
- ◆ Đó chính là kỹ thuật quay lui vét cạn, vì chúng ta phải lùi đến tất cả các nút lá mới định trị được các nút trong và từ đó mới định trị được cho nút gốc.

Kỹ thuật Vét cạn – Định vị cây biểu thức số học

Ví dụ: định trị cây biểu thức số học $5 + 2 * 3 - 4$



Quá trình định trị cây biểu thức số học

Nguyễn Thái Dư - AGU

45

Kỹ thuật Vét cạn – Định vị cây biểu thức số học

- Giải thuật quay lui (vét cạn) cho việc định trị cây biểu thức số học

```
float Eval(node n) {  
    if (n là lá)  
        return (trị của toán hạng trong n);  
    else  
        return (Toán tử trong n (Eval (Con trái của n), Eval (Con phải của n)));  
}
```

- Muốn định trị cho cây biểu thức T, ta gọi Eval(ROOT(T)).

Nguyễn Thái Dư - AGU

46

Kỹ thuật Vét cạn – Cây trò chơi – Mô tả

- Xét một trò chơi trong đó hai người thay phiên nhau đi nước của mình như cờ vua, cờ tướng, carô...
- Trò chơi có một trạng thái bắt đầu và mỗi nước đi sẽ biến đổi trạng thái hiện hành thành một trạng thái mới.
- Trò chơi sẽ kết thúc theo một quy định nào đó, theo đó thì cuộc chơi sẽ dẫn đến một trạng thái phản ánh:
 - có một người thắng cuộc
 - hoặc một trạng thái mà cả hai đấu thủ không thể phát triển được nước đi của mình, ta gọi nó là trạng thái hòa cờ.
- Ta tìm cách phân tích xem từ một trạng thái nào đó sẽ dẫn đến đấu thủ nào sẽ thắng với điều kiện cả hai đấu thủ đều có trình độ như nhau.

Nguyễn Thái Dư - AGU

47

Kỹ thuật Vét cạn – Cây trò chơi – Biểu diễn

- Một trò chơi có thể được biểu diễn bởi một cây trò chơi.
- Mỗi một nút của cây biểu diễn cho một trạng thái.
- Nút gốc biểu diễn cho trạng thái bắt đầu của cuộc chơi.
- Mỗi nút lá biểu diễn cho một trạng thái kết thúc của trò chơi (trạng thái thắng, thua hoặc hòa).
- Nếu trạng thái x được biểu diễn bởi nút n thì các con của n biểu diễn cho tất cả các trạng thái kết quả của các nước đi có thể xuất phát từ trạng thái x.

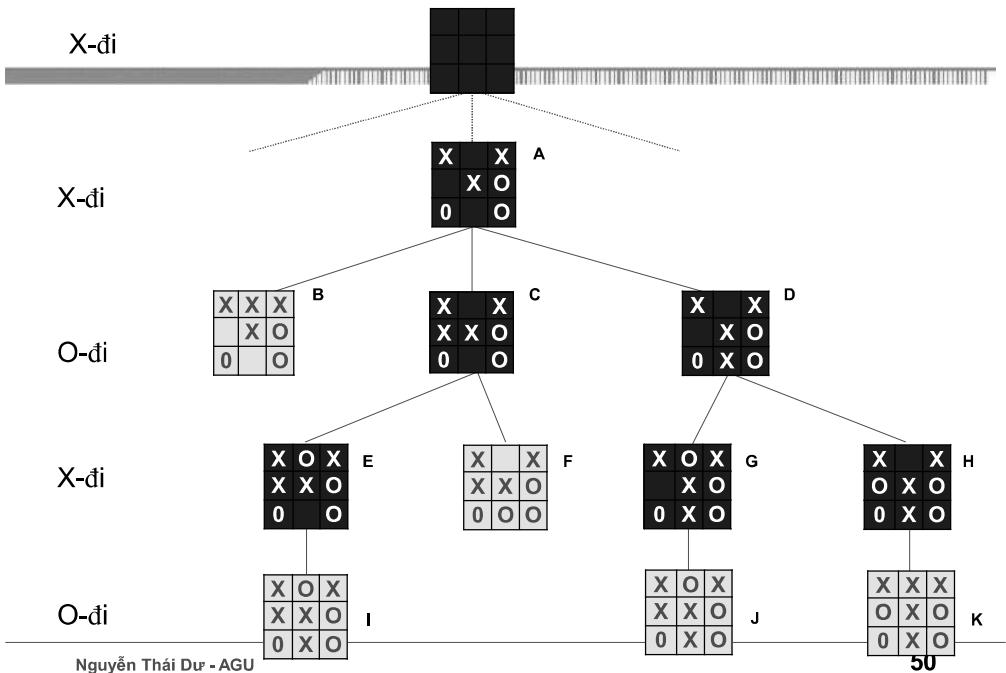
Nguyễn Thái Dư - AGU

48

Kỹ thuật Vết cạn – Cây trò chơi

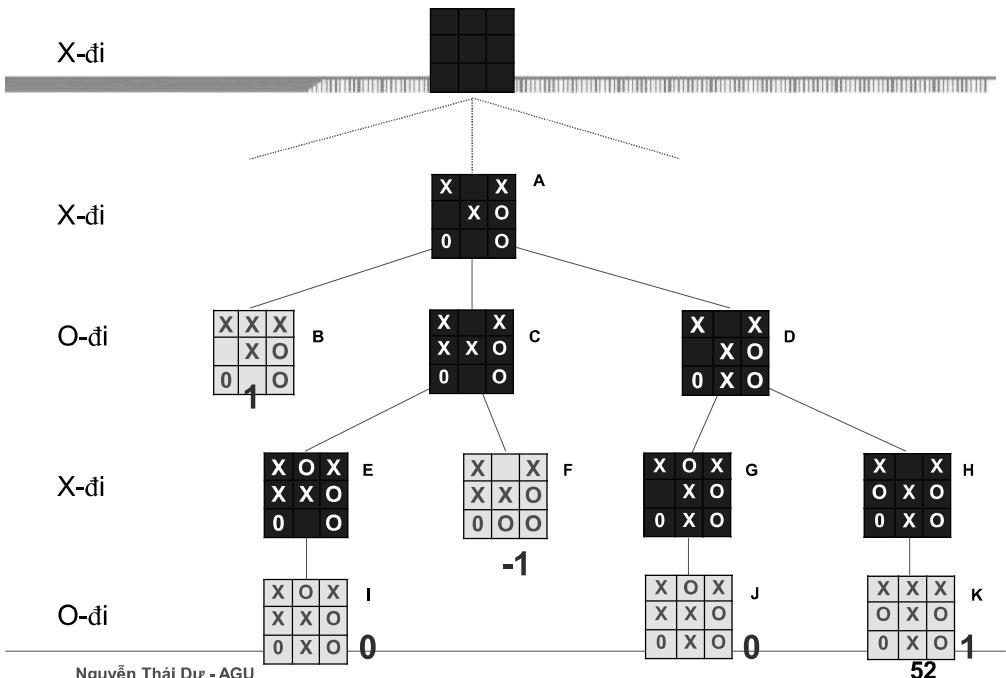
◆ **Ví dụ:** Xét trò chơi carô có 9 ô.

- Hai người thay phiên nhau đi X hoặc O.
- Người nào đi được 3 ô thẳng hàng (ngang, dọc, chéo) thì thắng cuộc.
- Nếu đã hết ô đi mà chưa phân thắng bại thì hai đấu thủ hòa nhau.
- Một phần của trò chơi này được biểu diễn bởi cây ở trang sau (*cây trò chơi carô 9 ô*)



Kỹ thuật Vết cạn – Cây trò chơi

- ◆ Trong cây trò chơi, các nút lá được tô nền và viền khung đôi để dễ phân biệt với các nút khác.
- ◆ Ta gắn cho mỗi nút một chữ cái (A, B, C...) để tiện trong việc trình bày các giải thuật.
- ◆ Ta có thể gán cho mỗi nút lá một giá trị để phản ánh trạng thái thắng thua hay hòa của các đấu thủ. Chẳng hạn ta gán cho nút lá các giá trị như sau:
 - 1 nếu tại đó người đi X đã thắng,
 - -1 nếu tại đó người đi X đã thua và
 - 0 nếu hai đấu thủ đã hòa nhau.



Kỹ thuật Vét cạn – Cây trò chơi

- ◆ Người đi X sẽ chọn cho mình một nước đi sao cho dẫn đến trạng thái có giá trị lớn nhất (trong trường hợp này là 1).
 - Ta nói X chọn nước đi MAX, nút mà từ đó X chọn nước đi của mình được gọi là nút MAX.
- ◆ Người đi O đến lượt mình sẽ chọn một nước đi sao cho dẫn đến trạng thái có giá trị nhỏ nhất (trong trường hợp này là -1, khi đó X sẽ thua và do đó O sẽ thắng).
 - Ta nói O chọn nước đi MIN, nút mà từ đó O chọn nước đi của mình được gọi là nút MIN.
- ◆ Do hai đấu thủ luân phiên nhau đi nước của mình nên các mức trên cây trò chơi cũng luân phiên nhau là MAX và MIN. Cây trò chơi vì thế còn có tên là cây MIN-MAX

Kỹ thuật Vét cạn – Giải thuật định vị Cây trò chơi

```
float Search(NodeType n, ModeType mode) {  
    NodeType C; /*C là một nút con của nút n*/  
    float value;  
    if(is_leaf(n))  
        return Payoff(n);  
    if(mode == MAX) value = -∞;  
    else value = ∞;  
    for (với mỗi con C của n)  
        if(mode == MAX)  
            value = max(value, Search(C, MIN));  
        else  
            value = min(value, Search(C, MAX));  
    return value;  
}
```

/*Khởi tạo giá trị tạm cho n
Lúc đầu ta cho value một giá trị tạm,
sau khi đã xét hết tất cả các con của nút
n thì value là giá trị của nút n*/

/*Xét tất cả các con của n, mỗi lần xác định được
giá trị của một nút con, ta
phải đặt lại giá trị tạm
value. Khi đã xét hết tất
cả các con thì value là
giá trị của n*/

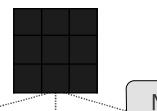
Kỹ thuật Vét cạn – Cây trò chơi

- ◆ Ta có thể đưa ra một quy tắc định trị cho các nút trên cây để phản ánh tình trạng thắng thua hay hòa và khả năng thắng cuộc của hai đấu thủ.
- ◆ Nếu một nút là nút lá thì trị của nó là giá trị đã được gán cho nút đó.
- ◆ Ngược lại, nếu nút là nút MAX thì trị của nó bằng giá trị lớn nhất của tất cả các trị của các con của nó. Nếu nút là nút MIN thì trị của nó là giá trị nhỏ nhất của tất cả các trị của các con của nó.
- ◆ **Ví dụ:** Vận dụng quy tắc quay lui vét cạn để định trị cho nút A trong cây trò chơi trong ví dụ trước

Kỹ thuật Vét cạn – Cây trò chơi

- ◆ Hàm Search nhận vào một nút n và kiểu mode của nút đó (MIN hay MAX) trả về giá trị của nút.
- ◆ Nếu nút n là nút lá thì trả về giá trị đã được gán cho nút lá.
- ◆ Ngược lại ta cho n một giá trị tạm value tương ứng như sau:
 - Nếu n là nút MAX thì gán value = -∞
 - Nếu n là nút MIN thì gán value = ∞
- ◆ Sau khi một con của n có giá trị V thì đặt lại value như sau:
 - Nếu n là nút MAX thì gán value = max(value, V)
 - Nếu n là nút MIN thì gán value = min(value, V)
- ◆ Khi tất cả các con của n đã được xét thì giá trị tạm value của n trở thành giá trị của nó

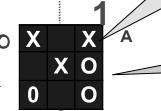
X-đi - Max



X thắng

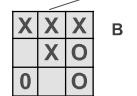
$$\text{Max}(0;1)=1$$

X-đi - Max

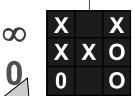


$$\text{Max}(-1;1)=1$$

O-đi - Min



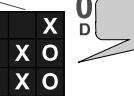
$$\text{Max}(-\infty;1)=1$$



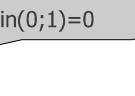
$$\text{Max}(-\infty;1)=1$$



$$\text{Max}(-\infty;1)=1$$

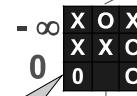


$$\text{Max}(-\infty;1)=1$$

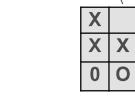


$$\text{Min}(0;1)=0$$

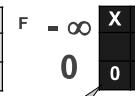
X-đi - Max



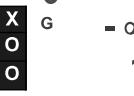
$$\text{Max}(-\infty;0)=0$$



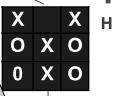
$$\text{Max}(-\infty;0)=0$$



$$\text{Max}(-\infty;0)=0$$



$$\text{Max}(-\infty;0)=0$$



$$\text{Max}(-\infty;0)=0$$

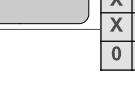
O-đi - Min



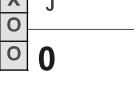
$$\text{Max}(-\infty;0)=0$$



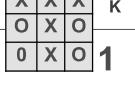
$$\text{Max}(-\infty;0)=0$$



$$\text{Max}(-\infty;0)=0$$



$$\text{Max}(-\infty;0)=0$$



$$\text{Max}(-\infty;0)=0$$

Nguyễn Thái Dư - AGU

Nguyễn Thái Dư - AGU

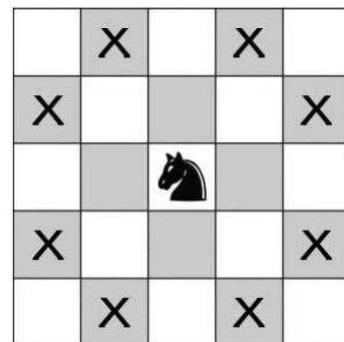
58

Kỹ thuật Vét cạn – Cây trò chơi

- Trong hình trên, các nút lá có giá trị gán ghi phía dưới mỗi nút.
- Đối với các nút trong, bên trái ghi các giá trị tạm theo thứ tự từ trên xuống, các giá trị thực được ghi bên phải hoặc phía trên bên phải

Kỹ thuật vét cạn – Bài toán mã đi tuần

- Bàn cờ vua có kích thước 8x8
- In đường đi của mã trên khắp bàn cờ



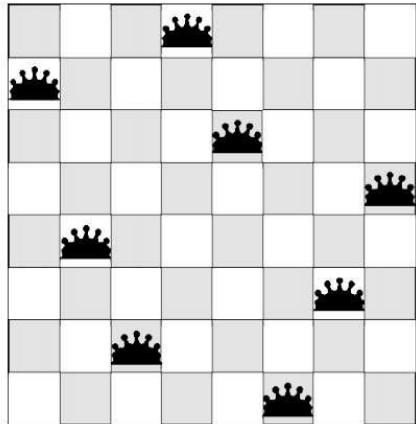
Kỹ thuật vét cạn – Bài toán mã đi tuần

- Bàn cờ vua có kích thước 8x8
- In 1 đường đi của mã trên khắp bàn cờ

1	48	31	50	33	16	63	18
30	51	46	3	62	19	14	35
47	2	49	32	15	34	17	64
52	29	4	45	20	61	36	13
5	44	25	56	9	40	21	60
28	53	8	41	24	57	12	37
43	6	55	26	39	10	59	22
54	27	42	7	58	23	38	11

Kỹ thuật vét cạn – Bài toán 8 hậu

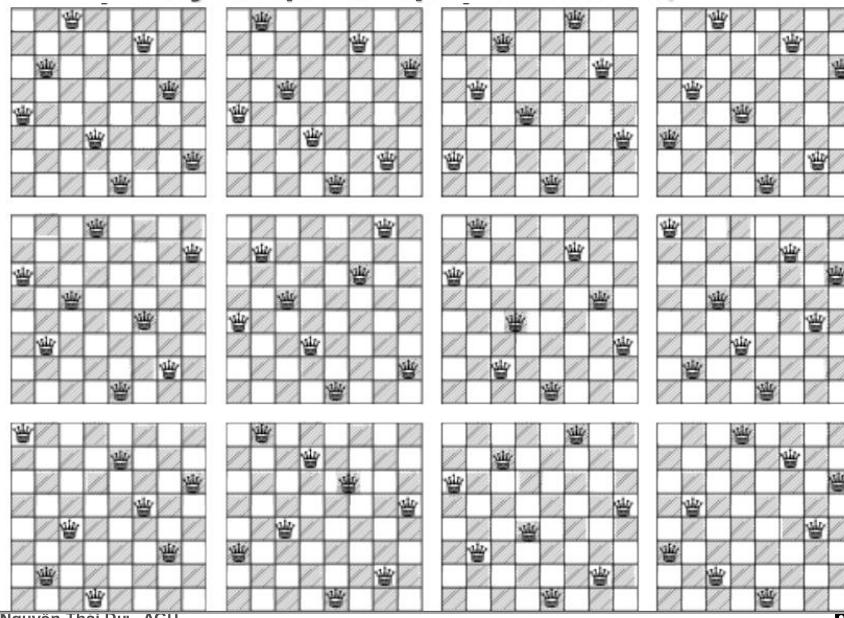
- Bàn cờ vua có kích thước 8x8
- Đặt 8 con hậu sao cho chúng không ăn nhau



Nguyễn Thái Dư - AGU

61

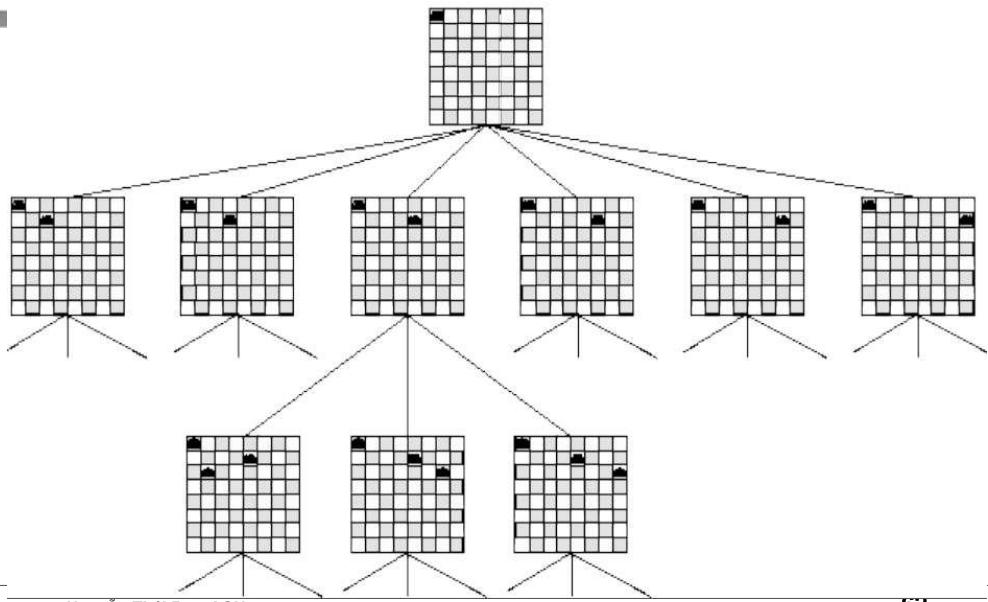
Kỹ thuật vét cạn – Bài toán 8 hậu



Nguyễn Thái Dư - AGU

62

Kỹ thuật vét cạn – Bài toán 8 hậu



Nguyễn Thái Dư - AGU

62

Kỹ thuật cắt tỉa Alpha-Beta (Alpha-Beta Pruning)

- ◆ **Nhận xét:** trong giải thuật vét cạn ở trên:
 - Để định trị cho một nút nào đó,
 - ta phải định trị cho tất cả các nút con cháu của nó
 - và muốn định trị cho nút gốc ta phải định trị cho tất cả các nút trên cây.
 - Số lượng các nút trên cây trò chơi tuy hữu hạn nhưng không phải là ít.
 - Chẳng hạn trong cây trò chơi ca rô nói trên, nếu ta có bàn cờ bao gồm n ô thì có thể có tới $n!$ nút trên cây (trong trường hợp trên là $9!$).
 - Đối với các loại cờ khác như cờ vua chẳng hạn, thì số lượng các nút còn lớn hơn nhiều.
 - Ta gọi là một sự bùng nổ tổ hợp các nút.

Nguyễn Thái Dư - AGU

64

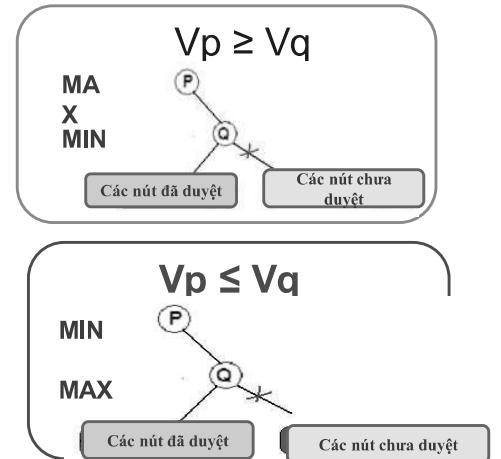
Kỹ thuật cắt tỉa Alpha-Beta (Alpha-Beta Pruning)

Ý tưởng cắt tỉa Alpha-Beta: Khi định trị một nút thì không nhất thiết phải định trị cho tất cả các nút con cháu của nó.

◆ Nếu P là một nút MAX và ta đang xét một nút con Q của nó (đã nhiên Q là nút MIN).

- Giả sử V_p là một giá trị tạm của P, V_q là một giá trị tạm của Q và nếu ta có $V_p \geq V_q$ thì ta không cần xét các con chưa xét của Q nữa.

◆ Nếu P là nút MIN (tất nhiên Q là nút MAX) và $V_p \leq V_q$ thì ta cũng không cần xét đến các con chưa xét của Q nữa.



Kỹ thuật cắt tỉa Alpha-Beta

◆ Quy tắc định trị cho một nút không phải là nút lá:

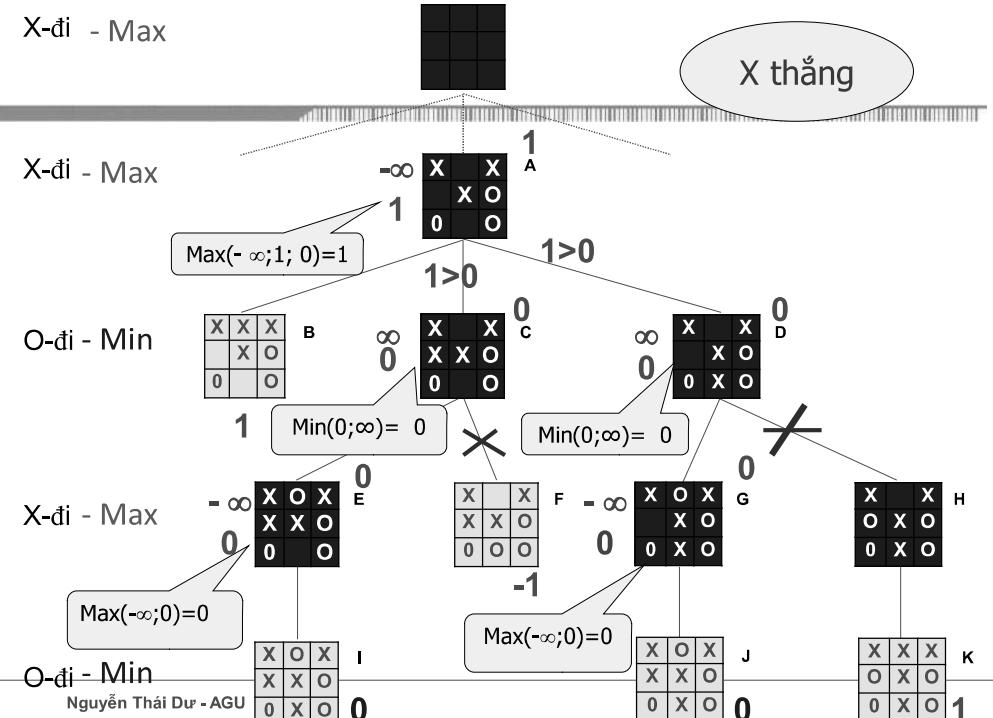
1. Khởi đầu nút MAX có giá trị tạm là $-\infty$ và nút MIN có giá trị tạm là ∞ .
2. Nếu tất cả các nút con của một nút đã được xét hoặc bị cắt tỉa thì giá trị tạm của nút đó trở thành giá trị của nó.
3. Nếu một nút MAX n có giá trị tạm là V_1 và một nút con của nó có giá trị là V_2 thì đặt giá trị tạm mới của n là $\max(V_1, V_2)$. Nếu n là nút MIN thì đặt giá trị tạm mới của n là $\min(V_1, V_2)$.
4. Vận dụng quy tắc cắt tỉa Alpha-Beta nói trên để hạn chế số lượng nút phải xét.

Kỹ thuật cắt tỉa Alpha-Beta

● Giải thuật cắt tỉa Alpha-Beta định trị cây trò chơi

```
float cat_tia(NodeType Q, ModeType mode, float Vp) {
    NodeType C; /* C là một nút con của nút Q */
    float Vq;
    /* Vq là giá trị tạm của Q, sau khi tất cả các con của nút Q đã xét hoặc bị cắt tỉa thì Vq là giá trị của nút Q */
    if (is_leaf(Q)) return Payoff(Q);
    /* Khởi tạo giá trị tạm cho Q */
    if (mode == MAX) Vq = -∞;
    else Vq = ∞;
    /* Xét các con của Q, mỗi lần xác định được giá trị của một nút con của Q, ta phải đặt lại giá trị tạm Vq và so sánh với Vp để có thể cắt tỉa hay không */
}
```

```
/* Xét C là con trái nhất của Q; */
while (C là con của Q) {
    if (mode == MAX) {
        Vq = max(Vq, Cat_tia(C, MIN, Vq));
        if (Vp <= Vq) return Vq;
        /* cắt tỉa tất cả các con còn lại của Q */
    }
    else {
        Vq = min(Vq, Cat_tia(C, MAX, Vq));
        if (Vp >= Vq) return Vq;
    }
}
return Vq;
```



Kỹ thuật nhánh cận – Bài toán cái Ba lô

- ◆ Nhánh cận là kỹ thuật xây dựng cây tìm kiếm phương án tối ưu, nhưng không xây dựng toàn bộ cây mà sử dụng giá trị cận để hạn chế bớt các nhánh.
- ◆ Với mỗi nút trên cây ta sẽ xác định một giá trị cận. Giá trị cận là một giá trị gần với giá của các phương án.
- ◆ Với bài toán tìm min ta sẽ xác định cận dưới còn với bài toán tìm max ta sẽ xác định cận trên.
 - Cận dưới là giá trị nhỏ hơn hoặc bằng giá của phương án,
 - ngược lại cận trên là giá trị lớn hơn hoặc bằng giá của phương án.

Nguyễn Thái Dư - AGU

69

Kỹ thuật nhánh cận – Bài toán cái Ba lô

- ◆ Đây là bài toán tìm Max.
 - ◆ Danh sách các đồ vật được sắp xếp theo thứ tự giảm của đơn giá
1. Nút gốc biểu diễn cho trạng thái ban đầu của ba lô, ở đó ta chưa chọn một vật nào.
 - Tổng giá trị được chọn $TGT = 0$.
 - Cận trên của nút gốc $CT = W * \text{Đơn giá lớn nhất}$.

Nguyễn Thái Dư - AGU

70

Kỹ thuật nhánh cận – Bài toán cái Ba lô

2. Nút gốc sẽ có các nút con tương ứng với các khả năng chọn đồ vật có đơn giá lớn nhất. Với mỗi nút con ta tính lại các thông số:
 - $TGT = TGT (\text{của nút cha}) + \text{số đồ vật được chọn} * \text{giá trị mỗi vật}$.
 - $W = W (\text{của nút cha}) - \text{số đồ vật được chọn} * \text{trọng lượng mỗi vật}$.
 - $CT = TGT + W * \text{Đơn giá của vật sẽ xét kế tiếp}$.
3. Trong các nút con, ta sẽ ưu tiên phân nhánh cho nút con nào có cận trên lớn hơn trước. Các con của nút này tương ứng với các khả năng chọn đồ vật có đơn giá lớn tiếp theo. Với mỗi nút ta lại phải xác định lại các thông số TGT , W , CT theo công thức đã nói trong bước 2.

Nguyễn Thái Dư - AGU

71

Kỹ thuật nhánh cận – Bài toán cái Ba lô

4. Lặp lại bước 3 với chú ý: đối với những nút có cận trên nhỏ hơn hoặc bằng giá lớn nhất tạm thời của một phương án đã được tìm thấy thì ta không cần phân nhánh cho nút đó nữa (cắt bỏ).
5. Nếu tất cả các nút đều đã được phân nhánh hoặc bị cắt bỏ thì phương án có giá lớn nhất là phương án cần tìm.

Nguyễn Thái Dư - AGU

72

Kỹ thuật nhánh cận – Bài toán cái Ba lô

- ◆ Ví dụ: Ta có một ba lô có trọng lượng là 37 và 4 loại đồ vật với trọng lượng và giá trị tương ứng được cho trong bảng bên dưới:

ĐV	TL	GT
A	15	30
B	10	25
C	2	2
D	4	6

ĐV: đồ vật

TL: Trọng lượng

ĐV	TL	GT	ĐG
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

GT: Giá trị

ĐG: Đơn giá

Nguyễn Thái Dư - AGU

73

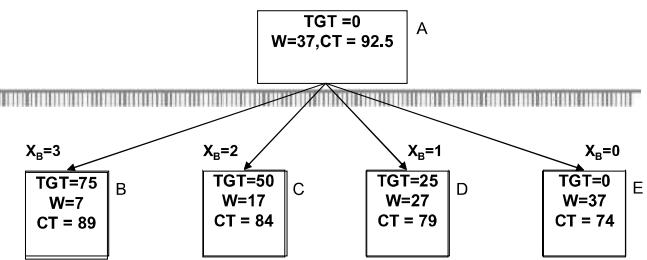
ĐV	TL	GT	ĐG
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

* Với nút B, ta có:

$$TGT = 0 + 3 * 25 = 75$$

$$W = 37 - 3 * 10 = 7$$

$$CT = 75 + 7 * 2 = 89$$



* Với nút D, ta có:

$$TGT = 0 + 1 * 25 = 25$$

$$W = 37 - 1 * 10 = 27$$

$$CT = 25 + 27 * 2 = 79$$

* Với nút E, ta có:

$$TGT = 0 + 0 * 25 = 0$$

$$W = 37 - 0 * 10 = 37$$

$$CT = 0 + 37 * 2 = 74$$

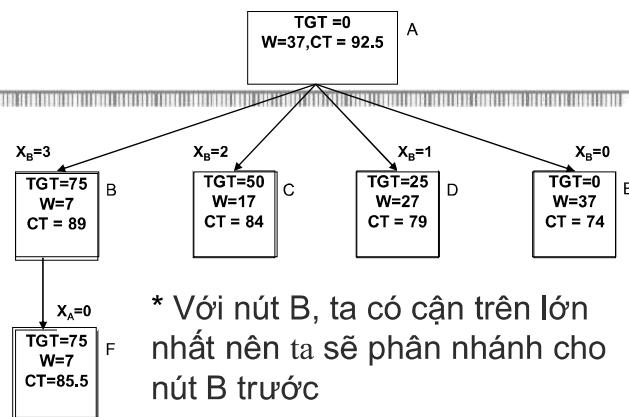
ĐV	TL	GT	ĐG
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

* Với nút F, ta có:

$$TGT = 75 + 0 * 30 = 75$$

$$W = 7 - 0 * 15 = 7$$

$$CT = 75 + 7 * 1.5 = 85,5$$



* Với nút B, ta có cận trên lớn nhất nên ta sẽ phân nhánh cho nút B trước

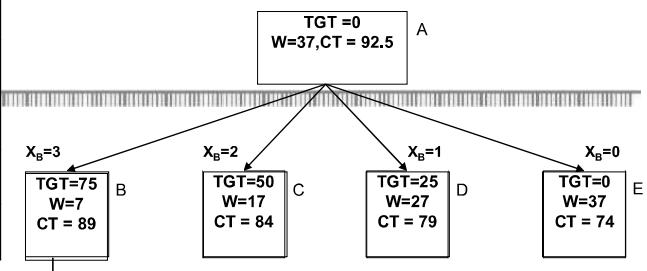
ĐV	TL	GT	ĐG
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

* Với nút G, ta có:

$$TGT = 75 + 1 * 6 = 81$$

$$W = 7 - 1 * 4 = 3$$

$$CT = 81 + 3 * 1 = 84$$



* Với nút F, ta có cận trên lớn nhất nên ta tiếp tục phân nhánh cho nút F

* Với nút H, ta có:

$$TGT = 75 + 0 * 6 = 75$$

$$W = 7 - 0 * 4 = 7$$

$$CT = 75 + 7 * 1 = 82$$

Nguyễn Thái Dư - AGU

75

76

ĐV	TL	GT	ĐG
B	10	25	2.5
A	15	30	2.0
D	4	6	1.5
C	2	2	1.0

* Với nút I, ta có:

$$TGT = 81 + 1 * 2 = 83$$

$$W = 3 - 1 * 2 = 1$$

* Với nút J, ta có:

$$TGT = 81 + 0 * 2 = 81$$

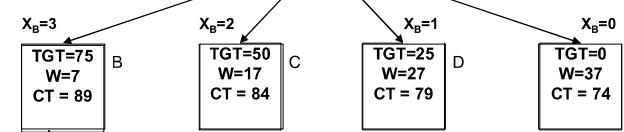
$$W = 3 - 0 * 2 = 3$$

Nguyễn Thái Dư - AGU

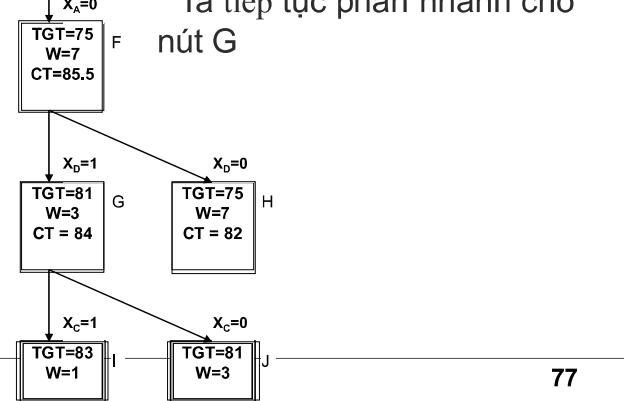
77

TGT = 0
W=37, CT = 92.5

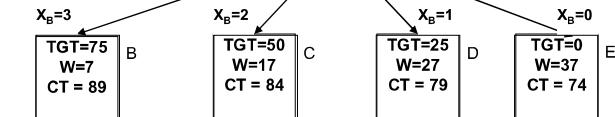
A



* Ta tiếp tục phân nhánh cho nút G



Nút I và J là hai nút lá (biểu diễn cho phương án) vì với mỗi nút thì số đồ vật đã được chọn xong.



Nút I biểu diễn cho p.a

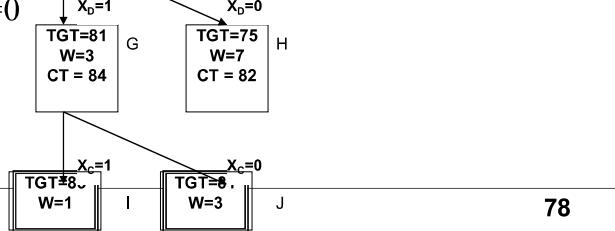
$$X_b=3, X_a=0, X_d=1, X_c=1$$

Giá: 83

Nút J biểu diễn cho p.a

$$X_b=3, X_a=0, X_d=1, X_c=0$$

Giá: 81T



Nguyễn Thái Dư - AGU

78

Như vậy giá trị lớn nhất tạm thời là 83

Quay lui lên nút H, ta thấy

cận trên của H là $82 < 83$

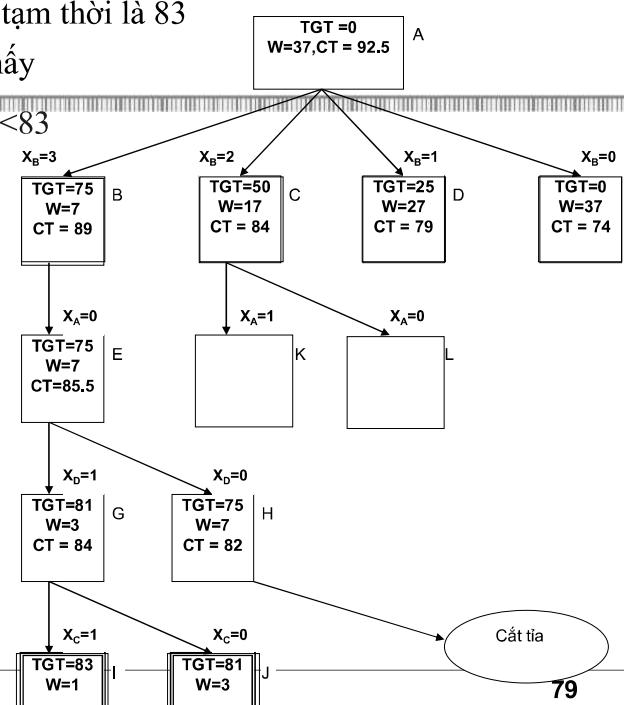
nên cắt tỉa nút H

Quay lui lên nút C

cận trên của C là $84 > 83$

Tiếp tục phân nhánh

cho nút C



Nguyễn Thái Dư - AGU

79

* Với nút K, ta có:

$$TGT = 50 + 1 * 30 = 80$$

$$W = 17 - 1 * 15 = 2$$

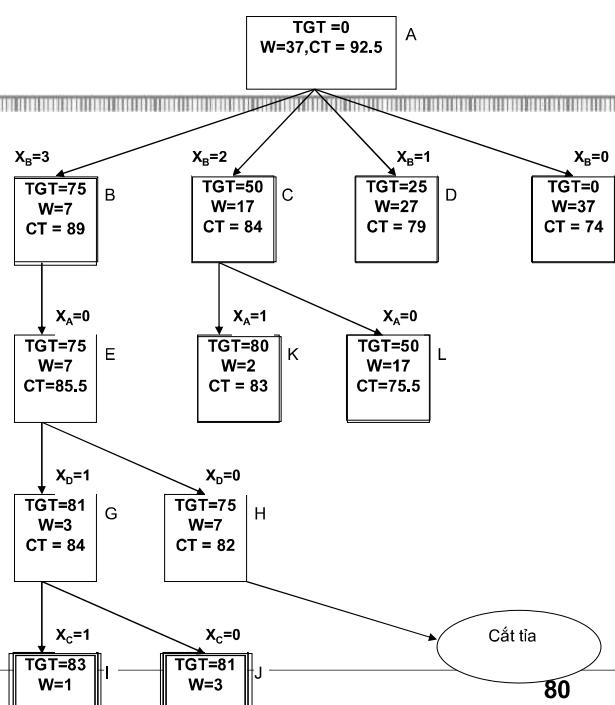
$$CT = 80 + 2 * 1,5 = 83$$

* Với nút L, ta có:

$$TGT = 50 + 0 * 30 = 50$$

$$W = 17 - 0 * 15 = 17$$

$$CT = 50 + 17 * 1,5 = 75,5$$



Nguyễn Thái Dư - AGU

80

Hai giá trị K, L đều không lớn hơn 83

Nên cả hai nút này đều bị cắt tỉa

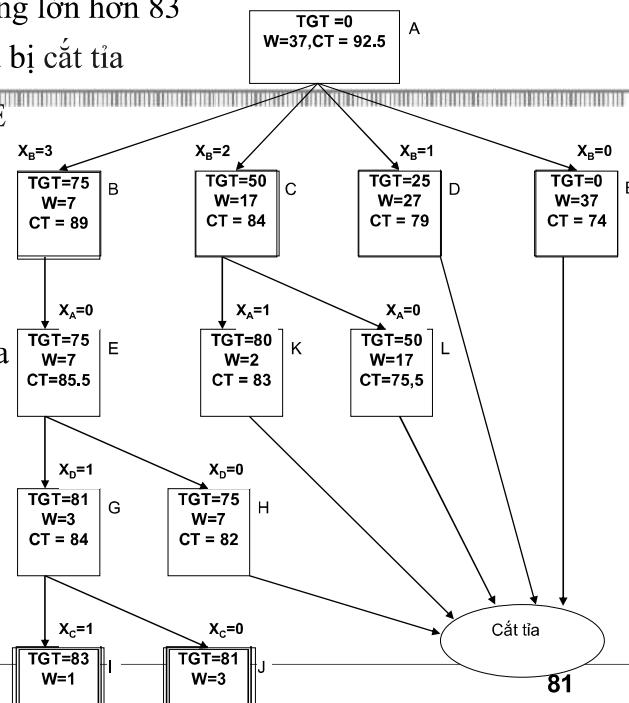
Tương tự các nút D và E

cũng bị cắt tỉa

Như vậy tất cả các nút

trên cây đều đã được

phân nhánh hoặc bị cắt tỉa



Nguyễn Thái Dư - AGU

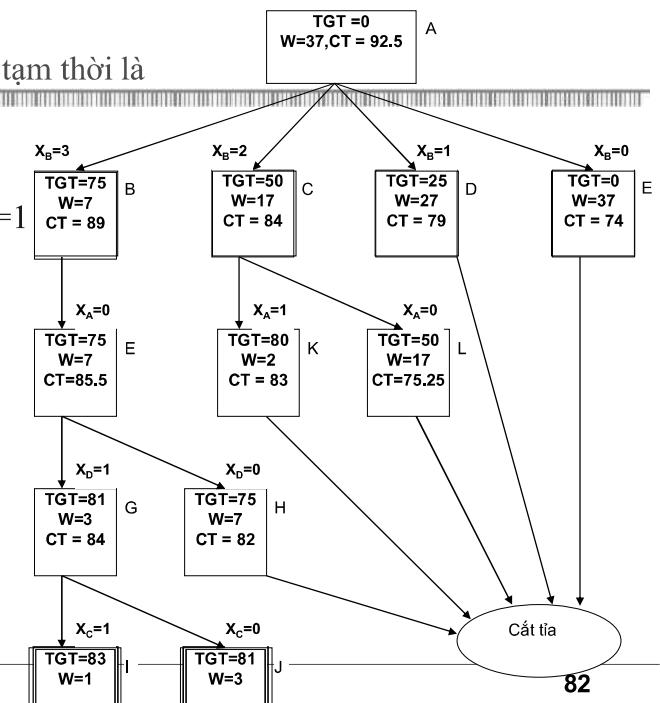
Vậy phương án tốt nhất tạm thời là
phương án cần tìm

Theo đó ta cần chọn:

$X_A=3, X_B=0, X_D=1, X_C=1$

Tổng giá trị: 83

Tổng trọng lượng: 36



Nguyễn Thái Dư - AGU

So sánh Vét cạn và Nhánh cạn

Vét cạn

```
else {
    for (j ∈ LC của i) {
        C[i] = j;
        search(i + 1);
        C[i] = null;
    }
}
```

Nhánh cạn

```
else {
    for (j ∈ LC của i)
        tính cận cho LC j
    S. xếp các LC theo cận
    for (j ∈ LC của i) {
        if (cận của j còn tốt) {
            C[i] = j;
            search (i + 1);
            C[i] = null;
        }
    }
}
```

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 (cơ chế caching)
- Đổi bộ nhớ lấy thời gian (trade memory for time)

Quy hoạch động

◆ Thiết kế giải thuật bằng kỹ thuật QHD

- Phân tích bài toán dung kỹ thuật chia để trị/quay lui
 - Chia bài toán thành các bài toán con
 - Tìm mối quan hệ giữa kết quả của bài toán trước và kết quả của các bài toán con (*công thức truy hồi*)
- Lập bảng quy hoạch

Quy hoạch động

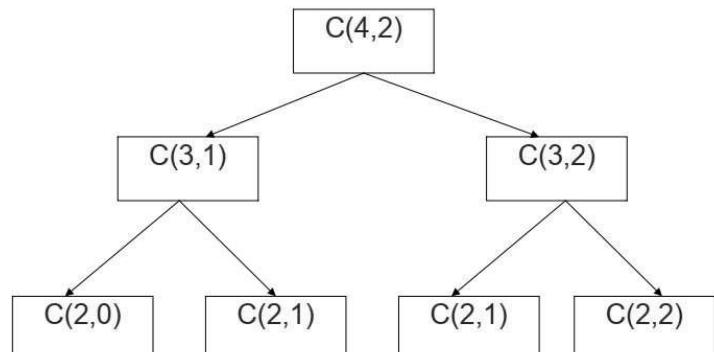
◆ Lập bảng quy hoạch

- Số chiều = số biến trong công thức truy hồi
- Thiết lập qui tắc đèn kết quả vào bảng quy hoạch
 - Điện các ô không phụ thuộc trước.
 - Điện các ô phụ thuộc sau.
- Tra bảng tìm kết quả (thường chỉ tìm được giá trị)
- Lần vết trên bảng để tìm lời giải tối ưu.

Quy hoạch động

◆ Ví dụ: Tính tổ hợp

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



Quy hoạch động

◆ Tính tổ hợp

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

Quy hoạch động

◆ Độ phức tạp giải thuật đệ quy:

- T(n) là thời gian để tính số tổ hợp chập k của n, thi ta có phương trình đệ quy:

$$T(1) = C_1$$

$$T(n) = 2T(n-1) + C_2$$

=> Vậy độ phức tạp quá lớn: $T(n) = O(2^n)$

Quy hoạch động

◆ 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	

Quy hoạch động

```
int comb(int n, int k) {
    int c[maxlen][maxlen], 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];
}
```

Quy hoạch động

```
int comb(int n, int k) {
    int c[maxlen], 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;
    }
    return c[k];
}
```

Chia để trị và Quy hoạch động

Chia để trị

- Ý tưởng
 - Phân rã thành các bài toán con
 - Tổng hợp kết quả
- Giải thuật:
 - Độ quy từ trên xuống
 - Độ phức tạp thời gian lớn nếu có nhiều bài toán 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
- Giải thuật:
 - 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 trữ bảng quy hoạch

Kết luận

- ◆ Mỗi kỹ thuật chỉ phù hợp với 1 hoặc 1 số loại bài toán
- ◆ Mỗi kỹ thuật đều có ưu và khuyết điểm, không có kỹ thuật nào là “trị bá bệnh”
 - Kỹ thuật nhánh cành cần phải có cách ước lượng cận tốt mới mong cắt được nhiều nhánh
 - 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

Kết hợp Quy hoạch động và đệ quy

- Sử dụng bảng quy hoạch để lưu kết quả bài toán con
- Không cần điền hết tất cả bảng quy hoạch
 - Điền bảng quy hoạch theo yêu cầu
 - Bắt đầu từ bài toán gốc
 - Nếu trong bảng quy hoạch chưa có KQ, gọi đệ quy để tìm kết quả và **lưu kết quả vào bảng quy hoạch**
 - Nếu KQ đã có trong bảng quy hoạch, sử dụng ngay kết quả này
 - Có thể sử dụng bảng băm để lưu trữ bảng quy hoạch

Phân tích Bài toán Dãy Fibonaci

- ◆ Dãy số Fibonaci được cho bởi công thức:

$$\begin{cases} F_n = F_{n-1} + F_{n-2}, \forall n \geq 2 \\ F_0 = 0 \\ F_1 = 1 \end{cases}$$

- ◆ Xây dựng thuật toán tính Fn với n là 1 số nguyên nhập từ bàn phím

Cách 1. giải bài toán Fibo bằng đệ quy

```
int fibonaci(int k)
{
    if(k>=2)
        return ( fibonaci(k-1) + fibonaci(k-2) );
    return 1;
}
```

Thuật toán trên hoàn toàn có thể cài đặt với 1 ngôn ngữ lập trình bất kỳ

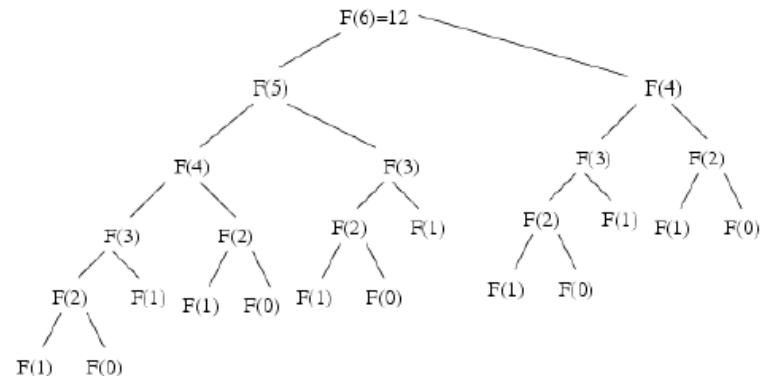
Cách 1. giải bài toán Fibo bằng đệ quy

Ta có $F_{n+1}/F_n \approx (1+\sqrt{5})/2 \approx 1.61803$ suy ra $F^n \approx 1.61803^n$.

- ◆ Do cây nhị phân tính Fn chỉ có 2 nút F0, F1 nên chúng ta sẽ có xấp xỉ 1.61803^n lần gọi đến hàm Fibonaci trên (thực tế n=6 là 13 lần)
- ◆ Có thể nói độ phức tạp của thuật toán là hàm mũ.

Cách 1. giải bài toán Fibo bằng đệ quy

- ◆ Tuy nhiên, đây là một thuật toán không hiệu quả, giả sử ta cần tính F6.



Cách 2. giải bài toán Fibo bằng quy hoạch động

- ◆ Sử dụng một thuật toán có độ phức tạp tuyến tính để tính Fn bằng cách sử dụng một mảng để lưu các giá trị dùng để tính Fn
 - $F_0 = 0$
 - $F_1 = 1$
 - For $i = 2$ to n
 - $F_i = F_{i-1} + F_{i-2}$
- ◆ Giảm được thời gian, mất thêm bộ nhớ để chứa các kết quả trung gian trong quá trình tính toán.

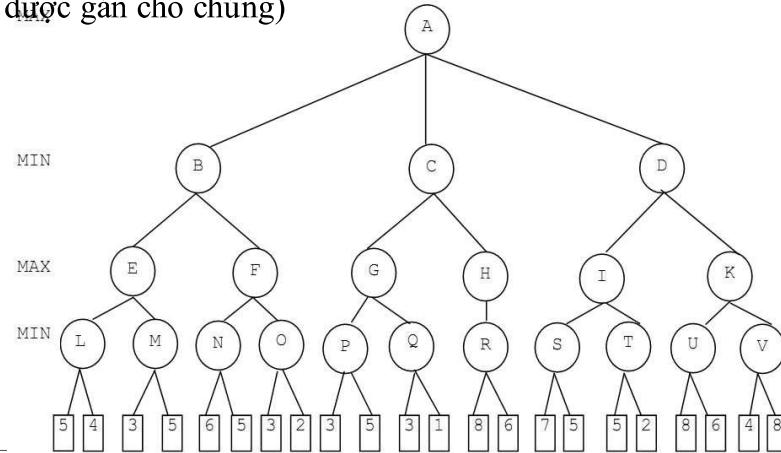
Cách 2. giải bài toán Fibo bằng quy hoạch động

◆ Qua ví dụ trên rút ra nhận xét.

- Quy hoạch động là một kỹ thuật tính toán đệ quy hiệu quả bằng cách lưu trữ các kết quả cục bộ
- Trong quy hoạch động kết quả của các bài toán con thường được lưu vào một mảng

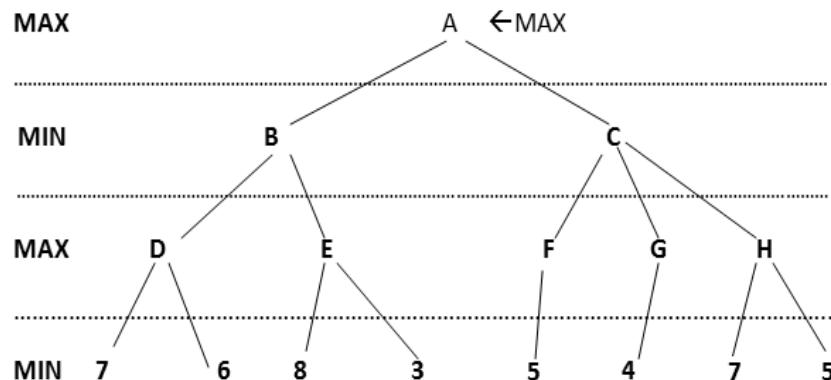
Bài tập Kỹ thuật Alpha-Beta

◆ **Bài 3:** Dùng kĩ thuật cắt tỉa alpha-beta để định trị cho nút gốc của cây trò chơi sau (các số trong các nút lá là các giá trị đã được gán cho chúng)



Bài tập Kỹ thuật Alpha-Beta

◆ **Bài 4:** Dùng kỹ thuật cắt tỉa alpha – beta để định trị cho nút gốc của cây trò chơi sau (các nút lá đã được gán trị):



Cảm ơn !