CHƯƠNG 3: KĨ THUẬT THIẾT KẾ GIẢI THUẬT

3.1 TỔNG QUAN

3.1.1 Mục tiêu

Nắm vững các kĩ thuật thiết kế giải thuật: chia để trị, quy hoạch động, tham ăn, quay lui, cắt tỉa alpha-beta, nhánh cận và tìm kiếm địa phương. Với mỗi kĩ thuật cần nắm được:

- Nội dung kĩ thuật.
- Vận dụng kĩ thuật vào giải các bài toán thực tế.
- Đánh giá được giải thuật.

3.1.2 Kiến thức cơ bản cần thiết

Các cấu trúc dữ liệu, đặc biệt là cấu trúc cây và đồ thị.

3.1.3 Tài liệu tham khảo

A.V. Aho, J.E. Hopcroft, J.D. Ullman; *Data Structures and Algorithms*; Addison-Wesley; 1983. (Chapter 10).
Jeffrey H Kingston; *Algorithms and Data Structures*; Addison-Wesley; 1998. (Chapter 12).

Đinh Mạnh Tường; *Cấu trúc dữ liệu & Thuật toán*; Nhà xuất bản khoa học và kĩ thuật; Hà nội-2001. (Chương 8).

Nguyễn Đức Nghĩa, Tô Văn Thành; *Toán rời rạc*; 1997 (Chương 3, 5).

3.1.4 Nội dung cốt lõi

Nói chung khi thiết kế một giải thuật chúng ta thường dựa vào một số kĩ thuật nào đó. Chương này sẽ trình bày một số kĩ thuật quan trọng để thiết kế giải thuật như: Chia để trị (Divide-and-Conquer), quy hoạch động (dynamic programming), kĩ thuật tham ăn (greedy techniques), quay lui (backtracking) và tìm kiếm địa phương (local search). Các kĩ thuật này được áp dụng vào một lớp rộng các bài toán, trong đó có những bài toán cổ điển nổi tiếng như bài toán tìm đường đi ngắn nhất của người giao hàng, bài toán cây phủ tối tiểu...

3.2 KĨ THUẬT CHIA ĐỂ TRỊ

3.2.1 Nội dung kĩ thuật

Có thể nói rằng kĩ thuật quan trọng nhất, được áp dụng rộng rãi nhất để thiết kế các giải thuật có hiệu quả là kĩ thuật "chia để trị" (divide and conquer). Nội dung của nó là: Để giải một bài toán kích thước n, ta chia bài toán đã cho thành một số bài toán con có kích thước nhỏ hơn. Giải các bài toán con này rồi tổng hợp kết quả lại để được lời giải của bài toán ban đầu. Đối với các bài toán con, chúng ta lại sử dụng kĩ

thuật chia để trị để có được các bài toán kích thước nhỏ hơn nữa. Quá trình trên sẽ dẫn đến những bài toán mà lời giải chúng là hiển nhiên hoặc dễ dàng thực hiện, ta gọi các bài toán này là **bài toán cơ sở.**

Tóm lại kĩ thuật chia để trị bao gồm hai quá trình: Phân tích bài toán đã cho thành các bài toán cơ sở và tổng hợp kết quả từ bài toán cơ sở để có lời giải của bài toán ban đầu. Tuy nhiên đối với một số bài toán, thì quá trình phân tích đã chứa đựng việc tổng hợp kết quả do đó nếu chúng ta đã giải xong các bài toán cơ sở thì bài toán ban đầu cũng đã được giải quyết. Ngược lại có những bài toán mà quá trình phân tích thì đơn giản nhưng việc tổng hợp kết quả lại rất khó khăn. Trong các phần tiếp sau ta sẽ trình bày một số ví dụ để thấy rõ hơn điều này.

Kĩ thuật này sẽ cho chúng ta một giải thuật đệ quy mà việc xác định độ phức tạp của nó sẽ phải giải một phương trình đệ quy như trong chương I đã trình bày.

3.2.2 Nhìn nhận lại giải thuật MergeSort và QuickSort

Hai giải thuật sắp xếp đã được trình bày trong các chương trước (MergeSort trong chương I và QuickSort trong chương II) thực chất là đã sử dụng kĩ thuật chia để trị.

Với MergeSort, để sắp một danh sách L gồm n phần tử, chúng ta chia L thành hai danh sách con L1 và L2 mỗi danh sách có n/2 phần tử. Sắp xếp L1, L2 và trộn hai danh sách đã được sắp này để được một danh sách có thứ tự. Quá trình phân tích ở đây là quá trình chia đôi một danh sách, quá trình này sẽ dẫn đến bài toán sắp xếp một danh sách có độ dai bằng 1, đây chính là bài toán cơ sở vì việc sắp xếp danh sách này là "không làm gì cả". Việc tổng hợp các kết quả ở đây là "trộn 2 danh sách đã được sắp để được một danh sách có thứ tự".

Với QuickSort, để sắp xếp một danh sách gồm n phần tử, ta 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. Quá trình phân chia sẽ dẫn đến các bài toán sắp xếp một danh sách chỉ gồm một phần tử hoặc gồm nhiều phần tử có khoá bằng nhau, đó chính là các bài toán cơ sở, vì bản thân chúng đã có thứ tự rồi. Ở đây chúng ta cũng không có việc tổng hợp kết quả một cách tường minh, vì việc đó đã được thực hiện trong quá trình phân hoạch.

3.2.3 Bài toán nhân các số nguyên lớn

Trong các ngôn ngữ lập trình đều có kiểu dữ liệu số nguyên (chẳng hạn kiểu integer trong Pascal, Int trong C…), nhưng nhìn chung các kiểu này đều có miền giá trị hạn chế (chẳng hạn từ -32768 đến 32767) nên khi có một ứng dụng trên số nguyên lớn (hàng chục, hàng trăm chữ số) thì kiểu số nguyên định sẵn không đáp ứng được. Trong trường hợp đó, người lập trình phải tìm một cấu trúc dữ liệu thích hợp để biểu diễn cho một số nguyên, chẳng hạn ta có thể dùng một chuỗi kí tự để biểu diễn cho một số nguyên, trong đó mỗi kí tự lưu trữ một chữ số. Để thao tác được trên các số nguyên được biểu diễn bởi một cấu trúc mới, người lập trình phải xây dựng các phép toán cho số nguyên như phép cộng, phép trừ, phép nhân… Sau đây ta sẽ đề cập đến bài toán nhân hai số nguyên lớn.

Xét bài toán nhân hai số nguyên lớn X và Y, mỗi số có n chữ số.

Đầu tiên ta nghĩ đến giải thuật nhân hai số thông thường, nghĩa là nhân từng chữ số của X với số Y rồi cộng các kết quả lại. Việc nhân từng chữ số của X với sô Y đòi hỏi phải nhân từng chữ số của X với từng chữ số của Y, vì X và Y đều có n chữ số nên cần n^2 phép nhân hai chữ số, mỗi phép nhân hai chữ số này tốn O(1) thì phép nhân cũng tốn $O(n^2)$ thời gian.

Áp dụng kĩ thuật "chia để trị" vào phép nhân các số nguyên lớn, ta chia mỗi số nguyên lớn X và Y thành các số nguyên lớn có n/2 chữ số. Để đơn giản cho việc phân tích giải thuật ta giả sử **n là luỹ thừa của 2**, còn về khía cạnh lập trình, ta vẫn có thể viết chương trình với n bất kì.

$X = A10^{n/2} + B$ và $Y = C10^{n/2} + D$

Trong đó A, B, C, D là các số nguyên lớn có n/2 chữ số.

Chẳng hạn với X = 1234 thì A = 12 và B = 34 bởi vì $X = 12 * 10^2 + 34$.

Khi đó tích của X và Y là: $XY = AC10^n + (AD + BC)10^{n/2} + BD$ (III.1)

Với mỗi số có n/2 chữ số, chúng ta lại tiếp tục phân tích theo cách trên, quá trình phân tích sẽ dẫn đến bài toán cơ sở là nhân các số nguyên lớn chỉ gồm một chữ số mà ta dễ dàng thực hiện. Việc tổng hợp kết quả chính là thực hiện các phép toán theo công thức (III.1).

Theo (III.1) thì chúng ta phải thực hiện 4 phép nhân các số nguyên lớn n/2 chữ số (AC, AD, BC, BD), sau đó tổng hợp kết quả bằng 3 phép cộng các số nguyên lớn n chữ số và 2 phép nhân với 10^n và $10^{n/2}$.

Các phép cộng các số nguyên lớn n chữ số dĩ nhiên chỉ cần O(n). Phép nhân với 10^n có thể thực hiện một cách đơn giản bằng cách thêm vào n chữ số 0 và do đó cũng chỉ lấy O(n). Gọi T(n) là thời gian để nhân hai số nguyên lớn, mỗi số có n chữ số thì từ (III.1) ta có phương trình đệ quy:

$$T(1) = 1$$

$$T(n) = 4T(n/2) + cn \quad \text{(III.2)}$$

Giải (III.2) ta được $T(n) = O(n^2)$. Như vậy thì chẳng cải tiến được chút nào so với giải thuật nhân hai số bình thường. Để cải thiện tình hình, chúng ta có thể viết lại (III.1) thành dạng:

$$XY = AC10^n + [(A-B)(D-C) + AC + BD]\ 10^{n/2} + BD \quad \text{(III.3)}$$

Công thức (III.3) chỉ đòi hỏi 3 phép nhân của các số nguyên lớn n/2 chữ số là: AC, BD và (A-B)(D-C), 6 phép cộng trừ và 2 phép nhân với 10^n. Các phép toán này đều lấy O(n) thời gian. Từ (III.3) ta có phương trình đệ quy:

$$T(1) = 1$$

$$T(n) = 3T(n/2) + cn$$

Giải phương trình đệ quy này ta được nghiệm $T(n) = O(n^{\log 3}) = O(n^{1.59})$. Giải thuật này rõ ràng đã được cải thiện rất nhiều.

Giải thuật thô để nhân hai số nguyên lớn (dương hoặc âm) n chữ số là:

```
FUNCTION Mult(X, Y: Big_integer; n:integer) : Big_integer;
```

```
VAR
   m1,m2,m3,A,B,C,D: Big_integer;
   s: integer;{Lưu trữ dấu của tích xy}
BEGIN
   s := sign(X)*sign(Y);
   x := ABS(X);{Lấy trị tuyệt đối của x}
   y := ABS(Y);
   IF n = 1 THEN mult := X*Y*s
   ELSE BEGIN
        A := left(X, n DIV 2);
        B := right(X, n DIV 2);
        C := left(Y, n DIV 2);
        D := right(Y, n DIV 2);
        m1 := mult(A,C, n DIV 2);
        m2 := mult(A-B,D-C, n DIV 2);
        m3 := mult(B,D, n DIV 2);
        mult := (s * (m1 * 10^n + (m1+m2+m3)* 10 ^ n DIV 2 + m3));
    END
END;
```

Hàm Mult nhận vào ba tham số, trong đó X và Y là hai số nguyên lớn (kiểu Big_integer), n là số chữ số của X và Y và trả về một số nguyên lớn là tích XY.
A, B, C, D là các biến thuộc kiểu Big_integer, lưu trữ các số nguyên lớn trong việc chia đôi các số nguyên lớn X và Y. m1, m2 và m3 là các biến thuộc kiểu Big_integer lưu trữ các số nguyên lớn trung gian trong công thức (III.3), cụ thể là m1 = AC, m2 = (A-B)(D-C) và m3 = BD.
Hàm sign nhận vào một số nguyên lớn X và cho giá trị 1 nếu X dương và -1 nếu X âm.
Hàm ABS nhận vào một số nguyên lớn X và cho kết quả là giá trị tuyệt đối của X.
Hàm Left nhận vào một số nguyên lớn X và một số nguyên k, cho kết quả là một số nguyên lớn có k chữ số bên trái của X. Tương tự như thế cho hàm Right.

3.2.4 Xếp lịch thi đấu thể thao

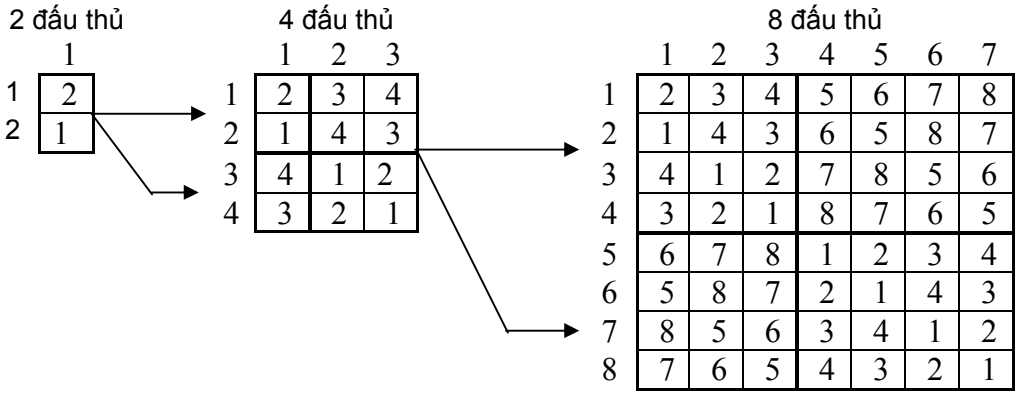
Kĩ thuật chia để trị không những chỉ có ứng dụng trong thiết kế giải thuật mà còn trong nhiều lĩnh vực khác của cuộc sống. Chẳng hạn xét việc xếp lịch thi đấu thể thao theo thể thức đấu vòng tròn 1 lượt cho n đấu thủ. Mỗi đấu thủ phải đấu với các đấu thủ khác, và mỗi đấu thủ chỉ đấu nhiều nhất một trận mỗi ngày. Yêu cầu là xếp một lịch thi đấu sao cho số ngày thi đấu là ít nhất. Ta dễ dàng thấy rằng tổng số trận đấu của toàn giải là $\frac{n(n-1)}{2}$. Như vậy nếu n là một số chẵn thì ta có thể sắp n/2 cặp thi đấu trong một ngày và do đó cần ít nhất n-1 ngày. Ngược lại nếu n là một số lẻ thì n-1 là một số chẵn nên ta có thể sắp (n-1)/2 cặp thi đấu trong một ngày và do đó ta cần n ngày. Giả sử $\mathbf{n = 2^k}$ thì n là một số chẵn và do đó cần tối thiểu n-1 ngày.

Lịch thi đấu là một bảng n dòng và n-1 cột. Các dòng được đánh số từ 1 đến n và các cột được đánh số từ 1 đến n-1, trong đó dòng i biểu diễn cho đấu thủ i, cột j biểu diễn cho ngày thi đấu j và ô(i,j) ghi đấu thủ phải thi đấu với đấu thủ i trong ngày j.

Chiến lược chia để trị xây dựng lịch thi đấu như sau: Để sắp lịch cho n đấu thủ, ta sẽ sắp lịch cho n/2 đấu thủ, để sắp lịch cho n/2 đấu thủ, ta sẽ sắp lịch cho n/4 đấu thủ... Quá trình này sẽ dẫn đến bài toán cơ sở là sắp lịch thi đấu cho 2 đấu thủ. Hai đấu thủ này sẽ thi đấu một trận trong một ngày, lịch thi đấu cho họ thật dễ sắp. Khó khăn chính là ở chỗ từ các lịch thi đấu cho hai đấu thủ, ta tổng hợp lại để được lịch thi đấu của 4 đấu thủ, 8 cấu thủ, ...

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 $\hat{O}(1,1)$ = "2" và $\hat{O}(2,1)$ = "1". Tương tự ta có lịch thi đấu cho 2 đấu thủ 3 và 4 trong ngày thứ 1. Nghĩa là $\hat{O}(3,1)$ ="4" và $\hat{O}(4,1)$ = "3". (Ta cố thể thấy rằng $\hat{O}(3,1) = \hat{O}(1,1) + 2$ và $\hat{O}(4,1) = \hat{O}(2,1) + 2$). Bây giờ để hoàn thành lịch thi đấu cho 4 đấu thủ, ta lấy góc trên bên trái của bảng lắp vào cho góc dưới bên phải và lấy góc dưới bên trái lắp cho góc trên bên phải.

Lịch thi đấu cho 8 đấu thủ là một bảng gồm 8 dòng, 7 cột. Góc trên bên trái chính là lịch thi đấu trong 3 ngày đầu của 4 đấu thủ từ 1 đến 4. Các ô của góc dưới bên trái sẽ bằng các ô tương ứng của góc trên bên trái cộng với 4. Đây chính là lịch thi đấu cho 4 đấu thủ 5, 6, 7 và 8 trong 3 ngày đầu. Bây giờ chúng ta hoàn thành việc sắp lịch bằng cách lấp đầy góc dưới bên phải bởi góc trên bên trái và góc trên bên phải bởi góc dưới bên trái.



Hình 3-1: Lịch thi đấu của 2, 4 và 8 đấu thủ

3.2.5 Bài toán con cân bằng (Balancing Subproblems)

Đối với kĩ thuật chia để trị, nói chung sẽ tốt hơn nếu ta chia bài toán cần giải thành các bài toán con có kích thước gần bằng nhau. Ví dụ, sắp xếp trộn (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(nlogn). 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 là chúng ta tìm cách chia bài toán thành các bài toán con có kích thước xấp xỉ bằng nhau thì hiệu suất sẽ cao hơn.

3.3 KĨ THUẬT "THAM ĂN"

3.3.1 Bài toán tối ưu tổ hợp

Là một dạng của bài toán tối ưu, nó có dạng tổng quát như sau:

- Cho hàm f(X) = xác định trên một tập hữu hạn các phần tử D. Hàm f(X) được gọi là hàm mục tiêu.
- Mỗi phần tử X $\in$ D có dạng X = (x1, x2, .. xn) được gọi là một phương án.
- Cần tìm một phương án X $\in$D sao cho hàm f(X) đạt min (max). Phương án X như thế được gọi là phương án tối ưu.

Ta có thể tìm thấy phương án tối ưu bằng phương pháp "vét cạn" nghĩa là xét tất cả các phương án trong tập D (hữu hạn) để xác đinh phương án tốt nhất. Mặc dù tập hợp D là hữu hạn nhưng để tìm phương án tối ưu cho một bài toán kích thước n bằng phương pháp "vét cạn" ta có thể cần một thời gian mũ.

Các phần tiếp theo của chương này sẽ trình bày một số kĩ thuật giải bài toán tối ưu tổ hợp mà thời gian có thể chấp nhận được.

3.3.2 Nội dung kĩ thuật tham ăn

Tham ăn hiểu một cách dân gian là: trong một mâm có nhiều món ăn, món nào ngon nhất ta sẽ ăn trước và ăn cho hết món đó thì chuyển sang món ngon thứ hai, lại ăn hết món ngon thứ hai này và chuyển sang món ngon thứ ba…

Kĩ thuật tham ăn thường được vận dụng để giải bài toán tối ưu tổ hợp bằng cách xây dựng một phương án X. Phương án X được xây dựng bằng cách lựa chọn từng thành phần Xi của X cho đến khi hoàn chỉnh (đủ n thành phần). Với mỗi Xi, ta sẽ chọn Xi 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.

Áp dụng kĩ thuật tham ăn sẽ cho một giải thuật thời gian đa thức, tuy nhiên nói chung chúng ta chỉ đạt được **một phương án tốt chứ chưa hẳn là tối ưu**.

Có rất nhiều bài toán mà ta có thể giải bằng kĩ thuật này, sau đây là một số ví dụ.

3.3.3 Bài toán trả tiền của máy rút tiền tự động ATM.

Trong máy rút tiền tự động ATM, ngân hàng đã chuẩn bị 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. Khi có một khách hàng cần rút một số tiền n đồng (tính chẵn đến 10.000 đồng, tức là n chia hết cho 10000). Hãy tìm một phương án trả tiền sao cho trả đủ n đồng và số tờ giấy bạc phải trả là ít nhất.

Gọi X = (X1, X2, X3, X4) là một phương án trả tiền, trong đó X1 là số tờ giấy bạc mệnh giá 100.000 đồng, X2 là số tờ giấy bạc mệnh giá 50.000 đồng, X3 là số tờ giấy bạc mệnh giá 20.000 đồng và X4 là số tờ giấy bạc mệnh giá 10.000 đồng. Theo yêu cầu ta phải có X1 + X2 + X3 + X4 nhỏ nhất và X1 * 100.000 + X2 * 50.000 + X3 * 20.000 + X4 * 10.000 = n.

Áp dụng kĩ thuật tham ăn để giải bài toán này là: để có số tờ giấy bạc phải trả (X1 + X2 + X3 + X4) 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 mệnh giá 100.000 đồng, nghĩa là X1 là số nguyên lớn nhất sao cho X1 * 100.000 ≤ n. Tức là X1 = n DIV 100.000.

Xác định số tiền cần rút còn lại là hiệu n – X1 * 100000 và chuyển sang chọn loại giấy bạc 50.000 đồng…

Ví dụ khách hàng cần rút 1.290.000 đồng (n = 1290000), phương án trả tiền như sau:

X1 = 1290000 DIV 100000 = 12.

Số tiền cần rút còn lại là 1290000 – 12 * 100000 = 90000.

X2 = 90000 DIV 50000 = 1.

Số tiền cần rút còn lại là 90000 – 1 * 50000 = 40000.

X3 = 40000 DIV 20000 = 2.

Số tiền cần rút còn lại là 40000 – 2 * 20000 = 0.

X4 = 0 DIV 10000 = 0.

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.

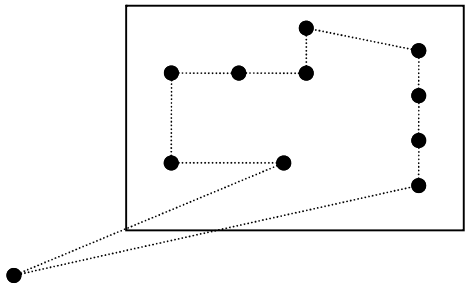
3.3.4 Bài toán đường đi của người giao hàng

Chúng ta sẽ xét một bài toán rất nổi tiếng có tên là bài toán tìm đường đi của người giao hàng (TSP - Traveling Salesman Problem): Có một người giao hàng cần đi giao hàng tại n thành phố. Xuất phát từ một thành phố nào đó, đi qua các thành phố khác để giao hàng và trở về thành phố ban đầu. Mỗi thành phố chỉ đến một lần, khoảng cách từ một thành phố đến các thành phố khác là xác định được. Giả thiết rằng mỗi thành phố đều có đường đi đến các thành phố còn lại. Khoảng cách giữa hai thành phố có thể là khoảng cách địa lý, có thể là cước phí di chuyển hoặc thời gian di chuyển. Ta gọi chung là độ dài. Hãy tìm một chu trình (một đường đi khép kín thỏa mãn điều kiện trên) sao cho tổng độ dài các cạnh là nhỏ nhất. Hay còn nói là tìm một phương án có giá nhỏ nhất. Bài toán này cũng được gọi là bài toán người du lịch.

Một cách tổng quát, có thể không tồn tại một đường đi giữa hai thành phố a và b nào đó. Trong trường hợp đó ta cho một đường đi ảo giữa a và b với độ dài bằng ∞.

Bài toán có thể biểu diễn bởi một đồ thị vô hướng có trọng số G = (V,E), trong đó mỗi thành phố được biểu diễn bởi một đỉnh, cạnh nối hai đỉnh biểu diễn cho đường đi giữa hai thành phố và trọng số của cạnh là khoảng cách giữa hai thành phố. Một chu trình đi qua tất cả các đỉnh của G, mỗi đỉnh một lần duy nhất, được gọi là chu trình Hamilton. Vấn đề là tìm một chu trình Hamilton mà tổng độ dài các cạnh là nhỏ nhất.

Bài toán này có những ứng dụng rất quan trọng. Thí dụ một máy hàn các điểm được điều khiển bởi máy tính. Nhiệm vụ của nó là hàn một số điểm dự định ở trên một tấm kim loại. Người thợ hàn bắt đầu từ một điểm bên ngoài tấm kim loại và kết thúc tại chính điểm này, do đó tấm kim loại phải được di chuyển để điểm cần hàn được đưa vào vị trí hàn (tương tự như ta đưa tấm vải vào đầu mũi kim của máy khâu). Cần phải tìm một phương án di chuyển tấm kim loại sao cho việc di chuyển ít nhất. Hình ảnh sau cho chúng ta hình dung về bài toán đặt ra.



Hình 3-2: Hàn các điểm trên một tấm kim loại

Dễ dàng thấy rằng, có thể áp dụng bài toán đường đi của người giao hàng để giải bài toán này.

Với phương pháp vét cạn ta xét tất cả các chu trình, mỗi chu trình tính tổng độ dài các cạnh của nó rồi chọn một chu trình có tổng độ dài nhỏ nhất. Tuy nhiên chúng ta cần xét tất cả là $\frac{(n-1)!}{2}$ chu trình. Thực vậy, do mỗi chu trình đều đi qua tất cả các đỉnh (thành phố) nên ta có thể cố định một đỉnh. Từ đỉnh này ta có n-1 cạnh tới n-1 đỉnh khác, nên ta có n-1 cách chọn cạnh đầu tiên của chu trình. Sau khi đã chọn được cạnh đầu tiên, chúng ta còn n-2 cách chọn cạnh thứ hai, do đó ta có (n-1)(n-2) cách chọn hai cạnh. Cứ lý luận như vậy ta sẽ thấy có (n-1)! cách chọn một chu trình. Tuy nhiên với mỗi chu trình ta chỉ quan tâm đến tổng độ dài các cạnh chứ không quan tâm đến hướïng đi theo chiều dương hay âm vì vậy có tất cả $\frac{(n-1)!}{2}$ phương án. Đó là một giải thuật thời gian mũ!.

Kĩ thuật tham ăn áp dụng vào đây là:

1. Sắp xếp các cạnh theo thứ tự tăng của độ dài.
2. Xét các cạnh có độ dài từ nhỏ đến lớn để đưa vào chu trình.
3. Một cạnh sẽ được đưa vào chu trình nếu cạnh đó thỏa mãn hai điều kiện sau:
   - Không tạo thành một chu trình thiếu (không đi qua đủ n đỉnh)
   - Không tạo thành một đỉnh có cấp ≥ 3 (tức là không được có nhiều hơn hai cạnh xuất phát từ một đỉnh, do yêu cầu của bài toán là mỗi thành phố chỉ được đến một lần: một lần đến và một lần đi)

4. Lặp lại bước 3 cho đến khi xây dựng được một chu trình.

Với kĩ thuật này ta chỉ cần n(n-1)/2 phép chọn nên ta có một giải thuật cần $O(n^2)$ thời gian.

Ví dụ 3-1: Cho bài toán TSP với 6 đỉnh được cho bởi các tọa độ như sau:

• c(1,7) • d(15,7)

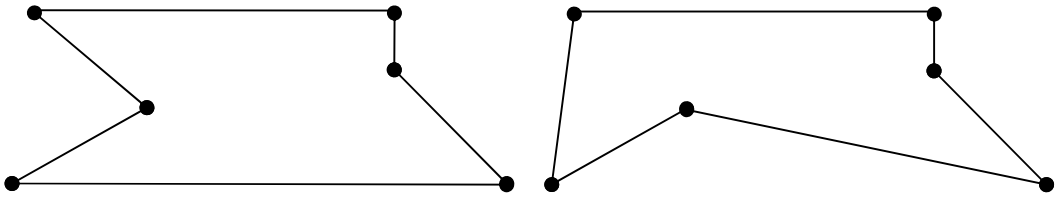
• b(4,3) • e(15,4)

• a(0,0) • f(18,0)

Hình 3-3: Sáu thành phố được cho bởi toạ độ

Do có 6 đỉnh nên có tất cả 15 cạnh. Đó là các cạnh: ab, ac, ad, ae, af, bc, bd, be, bf, cd, ce, cf, de, df và ef. Độ dài các cạnh ở đây là khoảng cách Euclide. Trong 15 cạnh này thì de = 3 là nhỏ nhất, nên de được chọn vào chu trình. Kế đến là 3 cạnh ab, bc và ef đều có độ dài là 5. Cả 3 cạnh đều thỏa mãn hai điều kiện nói trên, nên đều được chọn vào chu trình. Cạnh có độ dài nhỏ kế tiếp là ac = 7.08, nhưng không thể đưa cạnh này vào chu trình vì nó sẽ tạo ra chu trình thiếu (a-b-c-a). Cạnh df cũng bị loại vì lý do tương tự. Caûnh be được xem xét nhưng rồi cũng bị loại do tạo ra đỉnh b và đỉnh e có cấp 3. Tương tự chúng ta cũng loại bd. cd là cạnh tiếp theo được xét và được chọn. Cuối cùng ta có chu trình a-b-c-d-e-f-a với tổng độ dài là 50. Đây chỉ là một phương án tốt.

Phương án tối ưu là chu trình a-c-d-e-f-b-a với tổng độ dài là 48.39.



Hình3-4: Phương án Greedy và phương án tối ưu

Giải thuật sơ bộ như sau:

```
PROCEDURE TSP;
BEGIN
     {E là tập hợp các cạnh, Chu_trinh là tập hợp các cạnh
     được chọn để đưa vào chu trình, mở đầu Chu_trinh rỗng}
     {Sắp xếp các cạnh trong E theo thứ tự tăng của độ dài}
     Chu_Trinh := Φ;
     Gia := 0.0;
     WHILE E <> Φ DO BEGIN
          IF cạnh e có thể chọn THEN BEGIN
               Chu_Trinh := Chu_Trinh + [e]  ;
               Gia := Gia + độ dài của e;
```

```
            END;
            E := E-[e];
      END;
END;
```

Một cách tiếp cận khác của kĩ thuật tham ăn vào bài toán này là:

1. Xuất phát từ một đỉnh bất kỳ, chọn một cạnh có độ dài nhỏ nhất trong tất cả các cạnh đi ra từ đỉnh đó để đến đỉnh kế tiếp.

2. Từ đỉnh kế tiếp ta lại chọn một cạnh có độ dài nhỏ nhất đi ra từ đỉnh này thoả mãn hai điều kiện nói trên để đi đến dỉnh kế tiếp.

3. Lặp lại bước 2 cho đến khi đi tới đỉnh n thì quay trở về đỉnh xuất phát.

3.3.5 Bài toán cái ba lô

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 gi và một giá trị vi. 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.

Theo yêu cầu của bài toán thì ta cần những đồ vật có giá trị cao mà trọng lượng lại nhỏ để sao cho có thể mang được nhiều "đồ quý", sẽ là hợp lý khi ta quan tâm đến yếu tố "đơn giá" của từng loại đồ vật tức là tỷ lệ giá trị/trọng lượng. Đơn giá càng cao thì đồ càng quý. Từ đó ta có kĩ thuật greedy áp dụng cho bài toán này là:

1. Tính đơn giá cho các loại đồ vật.

2. Xét các loại đồ vật theo thứ tự đơn giá từ lớn đến nhỏ.

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.

Ví dụ 3-2: Ta có một ba lô có trọng lượng lài 37 và 4 loại đồ vật với trọng lượng và giá trị tương ứng được cho trong bảng bên.

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

Từ bảng đã cho ta tính đơn giá cho các loại đồ vật và sắp xếp các loại đồ vật này theo thứ tự đơn giá giảm dần ta có bảng sau.

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 đó thì thứ tự ưu tiên để chọn đồ vật là là B, A, D và cuối cùng là C.

Vật B được xét đầu tiên và ta chọn tối đa 3 cái vì mỗi cái vì trọng lượng mỗi cái là 10 và ba lô có trọng lượng 37. Sau khi đã chọn 3 vât loại B, trọng lượng còn lại trong ba lô là 37 - 3*10 = 7. Ta xét đến vật A, vì A có trọng lượng 15 mà trọng lượng còn lại của balô chỉ còn 7 nên không thể chọn vật A. Xét vật D và ta thấy có thể chọn 1 vật D, khi đó trọng lượng còn lại của ba lô là 7-4 = 3. Cuối cùng ta chọn được một vật C.

Như vậy chúng ta đã chọn 3 cái loại B, một cái loại D và 1 cái loại C. Tổng trọng lương là 3*10 + 1*4 + 1*2 = 36 và tổng giá trị là 3*25+1*6+1*2 = 83.

Giải thuật thô giải bài toán cái ba lô bằng kĩ thuật tham ăn như sau:

Tổ chức dữ liệu:

- Mỗi đồ vật được biểu diễn bởi một mẩu tin có các trường:
  - Ten: Lưu trữ tên đồ vật.
  - Trong_luong: Lưu trữ trọng lượng của đồ vật.
  - Gia_tri: Lưu trữ giá trị của đồ vật
  - Don_gia: Lưu trữ đơn giá của đồ vật
  - Phuong_an: Lưu trữ số lượng đồ vật được chọn theo phương án.
- Danh sách các đồ vật được biểu diễn bởi một mảng các đồ vật.

Khai báo bằng pascal:

```
Type
Do_vat = Record
          Ten: String[20]
          Trong_luong, Gia_tri, Don_gia : Real;
          Phuong_an : Integer;
End;
Danh_sach_do_vat = ARRAY[1..n] OF do_vat;

Procedure Greedy (VAR dsdv : Danh_sach_do_vat; W: real);
VAR i: integer;
BEGIN
     {Sắp xếp mảng dsdv theo thứ tự giảm của don_gia}
     FOR i:=1 TO n DO BEGIN
          Dsdv[i].Phuong_an:= Chon(dsdv[i].Trong_luong, W);
          W := W – dsdv[i].phuong_an * dsdv[i].Trong_luong;
     END;
END;
```

Trong đó hàm Chon(trong_luong, W) nhận vào trọng lượng trong_luong của một vật và trọng lượng còn lại W của ba lô, trả về số lượng đồ vật được chọn, sao cho tổng trọng lượng của các vật được chọn không lớn hơn W. Nói riêng, trong trường hợp trong_luong và W là hai sô nguyên thì Chon(Trong_luong, W) chính là W DIV Trong_luong.

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

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

3.4 QUY HOẠCH ĐỘNG

3.4.1 Nội dung kĩ thuật

Như trong 3.1 đã nói, kĩ thuật chia để trị thường dẫn chúng ta tới một giải thuật đệ quy. Trong các giải thuật đó, có thể có một số giải thuật có độ phức tạp thời gian mũ. Tuy nhiên, thường chỉ có một số đa thức các bài toán con, điều đó có nghĩa là chúng ta đã phải giải một số bài toán con nào đó nhiều lần. Để tránh việc giải dư thừa một số bài toán con, chúng ta tạo ra một bảng để lưu trữ kết quả của các bài toán con và khi cần chúng ta sẽ sử dụng kết quả đã được lưu trong bảng mà không cần phải giải lại bài toán đó. Lấp đầy bảng kết quả các bài toán con theo một quy luật nào đó để nhận được kết quả của bài toán ban đầu (cũng đã được lưu trong một số ô nào đó của bảng) được gọi là quy hoạch động (dynamic programming). Trong một số trường hợp, để tiết kiệm ô nhớ, thay vì dùng một bảng, ta chỉ dùng một véctơ.

Có thể tóm tắt giải thuật quy hoạch động như sau:

1. Tạo bảng bằng cách:
    a. Gán giá trị cho một số ô nào đó.
    b. Gán trị cho các ô khác nhờ vào giá trị của các ô trước đó.
2. Tra bảng và xác định kết quả của bài toán ban đầu.

Ưu điểm của phương pháp quy hoạch động là chương trình thực hiện nhanh do không phải tốn thời gian giải lại một bài toán con đã được giải.

Kĩ thuật quy hoạch động có thể vận dụng để giải các bài toán tối ưu, các bài toán có công thức truy hồi.

Phương pháp quy hoạch động sẽ không đem lại hiệu quả trong các trường hợp sau:

- Không tìm được công thức truy hồi.
- Số lượng các bài toán con cần giải quyết và lưu giữ kết quả là rất lớn.
- Sự kết hợp lời giải của các bài toán con chưa chắc cho ta lời giải của bài toán ban đầu.

Sau đây chúng ta sẽ trình bày một số bài toán có thể giải bằng kĩ thuật quy hoạch động.

3.4.2 Bài toán tính số tổ hợp

Một bài toán khá quen thuộc là tính số tổ hợp chập k của n theo công thức truy hồi:

$$C_n^k = \begin{cases} 1 \text{ nêu } k = 0 \text{ hoac } k = n \\ C_{n-1}^{k-1} + C_{n-1}^{k} \text{ nêu } 0 < k < n \end{cases}$$

Công thức trên đã gợi ý cho chúng ta một giải thuật đệ quy như sau:

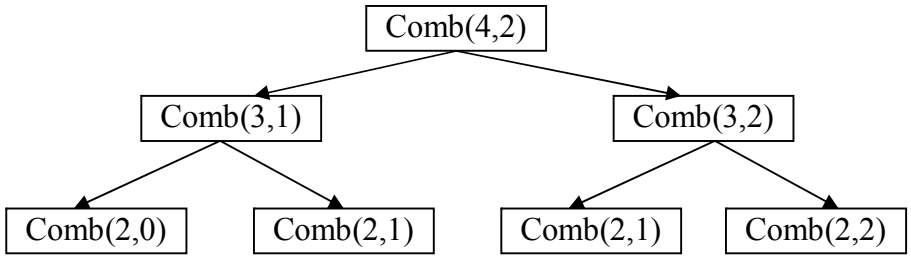
```
FUNCTION Comb(n,k : integer) : Integer;
BEGIN
     IF (k=0) OR (k=n) THEN Comb := 1
     ELSE  Comb := Comb(n-1, k-1) + Comb(n-1,k);
END;
```

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

T(1) = C1 và T(n) = 2T(n-1) + C2

Giải phương trình này ta được $T(n) = O(2^n)$, như vậy là một giải thuật thời gian mũ, trong khi chỉ có một đa thức các bài toán con. Điều đó chứng tỏ rằng có những bài toán con được giải nhiều lần.

Chẳng hạn để tính Comb(4,2) ta phải tính Comb(3,1) và Comb(3,2). Để tính Comb(3,1) ta phải tính Comb(2,0) và Comb(2,1). Để tính Comb(3,2) ta phải tính Comb(2,1) và Comb(2,2). Như vậy để tính Comb(4,2) ta phải tính Comb(2,1) hai lần. Hình sau minh hoạ rõ điều đó.



Hình 3-5 : Sơ đồ gọi thực hiện Com(4,2)

Áp dụng kĩ thuật quy hoạch động để khắc phục tình trạng trên, ta xây dựng một bảng gồm n+1 dòng (từ 0 đến n) và n+1 cột (từ 0 đến n) và điền giá trị cho O(i,j) theo quy tắc sau: (Quy tắc tam giác Pascal):

O(0,0) = 1;

O(i,0) =1;

O(i,i) = 1 với 0 < i (n;

O(i,j) = O(i-1,j-1) + O(i-1,j) với 0 < j < i (n.

Chẳng hạn với n = 4 ta có bảng bên.

i \ j	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

Tam giác Pascal

O(n,k) chính là Comb(n,k) và ta có giải thuật như sau:

```
FUNCTION Comb(n, k : Integer) : Integer
VAR  C: array[0..n, 0..n] of integer;
     i,j : integer;
BEGIN
```

```
{1}  C[0,0] := 1;
{2}  FOR i := 1 TO n DO BEGIN
{3}       C[i,0] := 1;
{4}       C[i,i] := 1;
{5}       FOR j := 1 TO i-1 DO C[i,j]:=C[i-1,j-1]+C[i-1,j];
     END;
{6}  Comb := C[n,k];
END;
```

Vòng lặp {5} thực hiện i-1 lần, mỗi lần O(1). Vòng lặp {2} có i chạy từ 1 đến n, nên nếu gọi T(n) là thời gian thực hiện giải thuật thì ta có:

$$T(n) = \sum_{i=1}^{n}(i-1) = \frac{n(n-1)}{2} = O(n^2)$$

Nhận xét: Thông qua việc xác định độ phức tạp, ta thấy rõ ràng giải thuật quy hoạch động hiệu quả hơn nhiều so với giải thuật đệ qui ($n^2 < 2^n$). Tuy nhiên việc sử dụng bảng (mảng hai chiều) như trên còn lãng phí ô nhớ, do đó ta sẽ cải tiến thêm một bước bằng cách sử dụng véctơ (mảng một chiều) để lưu trữ kết quả trung gian. Cách làm cụ thể như sau:

Ta sẽ dùng một véctơ V có n+1 phần tử từ V[0] đến V[n]. Véctơ V sẽ lưu trữ các giá trị tương ứng với dòng i trong tam giác Pascal ở trên. Trong đó V[j] lưu trữ giá trị số tổ hợp chập j của i (C^j_i) (j = 0 đến i). Dĩ nhiên do chỉ có một véctơ V mà phải lưu trữ nhiều dòng i do đó tại mỗi bước, V chỉ lưu trữ được một dòng và ở bước cuối cùng, V lưu trữ các giá trị ứng với i = n, trong đó V[k] chính là C^k_n.

Khởi đầu, ứng với i =1, ta cho V[0] = 1 và V[1] = 1. Tức là $C^0_1 = 1$ và $C^1_1 = 1$. Với các giá trị i từ 2 đến n, ta thực hiện như sau:

- V[0] được gán giá trị 1 tức là $C^0_i = 1$. Tuy nhiên giá trị V[0] = 1 đã được gán ở trên, không cần phải gán lại.
- Với j từ 1 đến i-1, ta vẫn áp dụng công thức $C^j_i = C^{j-1}_{i-1} + C^j_{i-1}$. Nghĩa là để tính các giá trị trong dòng i ta phải dựa vào dòng i-1. Tuy nhiên do chỉ có một véctơ V và lúc này nó sẽ lưu trữ các giá trị của dòng i, tức là dòng i-1 sẽ không còn. Để khắc phục điều này ta dùng thêm hai biến trung gian p1 và p2. Trong đó p1 dùng để lưu trữ C^{j-1}_{i-1} và p2 dùng để lưu trữ C^j_{i-1}. Khởi đầu p1 được gán V[0] tức là C^0_{i-1} và p2 được gán V[j] tức là C^j_{i-1}, V[j] lưu trữ giá trị C^j_i sẽ được gán bởi p1+p2, sau đó p1 được gán bởi p2, nghĩa là khi j tăng lên 1 đơn vị thành j+1 thì p1 là C^j_{i-1} và nó được dùng để tính C^{j+1}_i.
- Cuối cùng với j = i ta gán V[i] giá trị 1 tức là $C^i_i = 1$.

Giải thuật cụ thể như sau:

```
FUNCTION Comb(n, k : Integer) : Integer
VAR  V: array[0..n] of integer;
     i,j : integer;
     p1,p2: integer;
BEGIN
{1}  V[0] := 1;
```

```
{2}  V[1] := 1;
{3}  FOR i := 2 TO n DO BEGIN
{4}       p1 := V[0];
{5}       FOR j := 1 TO i-1 DO BEGIN
{6}            p2 := V[j];
{7}            V[j]:= p1+p2;
{8}            P1:= p2;
           END;
{9}       V[i] := 1;
     END;
{10} Comb := V[k];
END;
```

Dễ dàng tính được độ phức tạp của giải thuật vẫn là $O(n^2)$.

3.4.3 Bài toán cái ba lô

Sử dụng kĩ thuật quy hoạch động để giải bài toán cái ba lô đã trình bày trong mục 3.2.5 với một lưu ý là các số liệu đều cho dưới dạng **số nguyên**.

Giả sử X[k,V] là số lượng đồ vật k được chọn, F[k,V] là tổng giá trị của k đồ vật đã được chọn và V là trọng lượng còn lại của ba lô, k = 1..n, V = 1..W.

Trong trường hợp đơn giản nhất, khi chỉ có một đồ vật, ta tính được X[1,V] và F[1,V] với mọi V từ 1 đến W như sau:

X[1,V] = V DIV g1 và F[1,V] = X[1,V] * v1.

Giả sử ta đã tính được F[k-1,V], khi có thêm đồ vật thứ k, ta sẽ tính được F[k,V], với mọi V từ 1 đến W. Cách tính như sau: Nếu ta chọn xk đồ vật loại k, thì trọng lượng còn lại của ba lô dành cho k-1 đồ vật từ 1 đến k-1 là U = V-xk*gk và tổng giá trị của k loại đồ vật đã được chọn F[k,V] = F[k-1,U] + xk*vk, với xk thay đổi từ 0 đến yk= V DIV gk và ta sẽ chọn xk sao cho F[k,V] lớn nhất.

Ta có công thức truy hồi như sau:

X[1,V] = V DIV g1 và F[1,V] = X[1,V] * v1.

F[k,V] = Max(F[k-1,V-xk*gk] + xk*vk) với xk chạy từ 0 đến V DIV gk.

Sau khi xác định được F[k,V] thì X[k,V] là xk ứng với giá trị F[k,V] được chọn trong công thức trên.

Để lưu các giá trị trung gian trong quá trình tính F[k,V] theo công thức truy hồi trên, ta sử dụng một bảng gồm n dòng từ 1 đến n, dòng thứ k ứng với đồ vật loại k và W+1 cột từ 0 đến W, cột thứ V ứng với trọng lượng V. Mỗi cột V bao gồm hai cột nhỏ, cột bên trái lưu F[k,V], cột bên phải lưu X[k,V]. Trong lập trình ta sẽ tổ chức hai bảng tách rời là F và X.

Ví dụ bài toán cái ba lô với trọng lượng W=9, và 5 loại đồ vật được cho trong bảng sau

Đồ vật	Trọng lượng (gi)	Giá trị (vi)

1	3	4
2	4	5
3	5	6
4	2	3
5	1	1

Ta có bảng F[k,V] và X[k,V] như sau, trong đó mỗi cột V có hai cột con, cột bên trái ghi F[k,V] và cột bên phải ghi X[k,V].

v / k	0		1		2		3		4		5		6		7		8		9	
1	0	0	0	0	0	0	4	1	4	1	4	1	8	2	8	2	8	2	12	3
2	0	0	0	0	0	0	4	0	5	1	5	1	8	0	9	1	10	2	12	0
3	0	0	0	0	0	0	4	0	5	0	6	1	8	0	9	0	10	0	12	0
4	0	0	0	0	3	1	4	0	6	2	7	1	9	3	10	2	12	4	13	3
5	0	0	1	1	3	0	4	0	6	0	7	0	9	0	10	0	12	0	13	0

Trong bảng trên, việc điền giá trị cho dòng 1 rất đơn giản bằng cách sử dụng công thức: X[1,V] = V DIV g1 và F[1,V] = X[1,V] * v1.

Từ dòng 2 đến dòng 5, phải sử dụng công thức truy hồi:

F[k,V] = Max(F[k-1,V-xk*gk] + xk*vk) với xk chạy từ 0 đến V DIV gk.

Ví dụ để tính F[2,7], ta có xk chạy từ 0 đến V DIV gk, trong trường hợp này là xk chạy từ 0 đến 7 DIV 4, tức xk có hai giá trị 0 và 1.

Khi đó F[2,7] = Max (F[2-1, 7-0*4] + 0*5, F[2-1,7-1*4] + 1*5)

= Max(F[1,7], F[1,3] + 5) = Max(8, 4+5) = 9.

F[2,7] = 9 ứng với xk = 1 do đó X[2,7] = 1.

Vấn đề bây giờ là cần phải tra trong bảng trên để xác định phương án.

Khởi đầu, trọng lượng còn lại của ba lô V = W.

Xét các đồ vật từ n đến 1, với mỗi đồ vật k, ứng với trọng lượng còn lại V của ba lô, nếu X[k,V] > 0 thì chọn X[k,V] đồ vật loại k. Tính lại V = V - X[k,V] * gk.

Ví dụ, trong bảng trên, ta sẽ xét các đồ vật từ 5 đến 1. Khởi đầu V = W = 9.

Với k = 5, vì X[5,9] = 0 nên ta không chọn đồ vật loại 5.

Với k = 4, vì X[4,9] = 3 nên ta chọn 3 đồ vật loại 4. Tính lại V = 9 – 3 * 2 = 3.

Với k = 3, vì X[3,3] = 0 nên ta không chọn đồ vật loại 3.

Với k = 2, vì X[2,3] = 0 nên ta không chọn đồ vật loại 2.

Với k = 1, vì X[1,3] = 1 nên ta chọn 1 đồ vật loại 1. Tính lại V = 3 – 1 * 3 = 0.

Vậy tổng trọng lương các vật được chọn là 3 * 2 + 1 * 3 = 9. Tổng giá trị các vật được chọn là 3 * 3 + 1 * 4 = 13.

Giải thuật thô theo kĩ thuật quy hoạch động như sau:

Tổ chức dữ liệu:

- Mỗi đồ vật được biểu diễn bởi một mẩu tin có các trường:
  - Ten: Lưu trữ tên đồ vật.
  - Trong_luong: Lưu trữ trọng lượng của đồ vật.
  - Gia_tri: Lưu trữ giá trị của đồ vật
  - Phuong_an: Lưu trữ số lượng đồ vật được chọn theo phương án.
- Danh sách các đồ vật được biểu diễn bởi một mảng các đồ vật.
- Bảng được biểu diễn bởi một mảng hai chiều các số nguyên để lưu trữ các giá trị F[k,v] và X[k,v].

Khai báo bằng pascal:

```
Type
Do_vat = Record
          Ten: String[20]
          Trong_luong, Gia_tri : integer;
          Phuong_an : Integer;
End;
Danh_sach_vat = ARRAY[1..MAX] OF do_vat;
BANG =  ARRAY[1..10, 0..100] of integer;
```

Thủ tục tạo bảng nhận vào ds_vat là danh sách các vật, n là số lượng các loại vật, W là trọng lượng của ba lô. F và X là hai tham số thuộc kiểu Bang và được truyền bằng tham chiếu để nhận lại hai bảng F và X do thủ tục tạo ra.

```
PROCEDURE Tao_Bang (ds_vat:Danh_sach_vat;n,W: integer; VAR F,X: Bang);
VAR   xk, yk, k: integer;
FMax, XMax, v : integer;
BEGIN
FOR v:= 0 To W Do BEGIN {Hàng đầu tiên của hai bảng}
   X[1, v] := v div ds_vat[1].trong_luong;
   F[1, v] := X[1, v] * ds_vat[1].gia_tri;
END;
FOR k:= 2 TO N DO BEGIN
   X[k, 0] := 0;
   F[1, 0] := 0;
   For v:= 1 TO W DO BEGIN
     FMax := F[k-1, v] ;
     XMax := 0;
     yk := v DIV ds_vat[k].trong_luong;
     FOR xk:= 1 TO yk DO
        If(F[k-1,v-xk*ds_vat[k].trong_luong]+xk*ds_vat[k].gia_tri>FMax)
        THEN BEGIN
            FMax:=F[k-1,v-k*ds_vat[k].trong_luong]+xk*ds_vat[k].gia_tri;
            XMax:= xk;
        END ;
      F[k, v] := FMax;
      X[k, v] := XMax;
    END;
  END;
END;
```

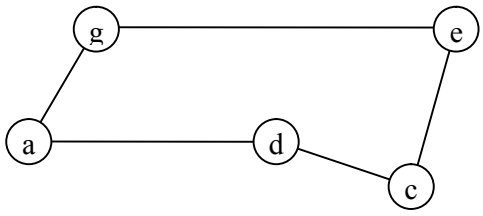
Thủ tục Tra_bang nhận vào hai bảng F và X; n là số lượng các loại đồ vật, W là trọng lượng của ba lô và trả ra ds_vat là một danh sách đồ vật đã được xác định phương án. Tham số ds_vat được truyền bằng tham chiếu.

```
PROCEDURE Tra_Bang(VAR ds_vat:Danh_sach_vat;n,W:integer;F,X:
Bang);
VAR k, v: integer;
BEGIN
   v := W;
   FOR k:= n DOWNTO 1 DO
     IF X[k,v] > 0 THEN BEGIN
          ds_vat[k].Phuong_an := X[k,v];
          v := v - X[k, v] * ds_vat[k].trong_luong;
     END;
END;
```

3.4.4 Bài toán đường đi của người giao hàng

Chúng ta có thể áp dụng kĩ thuật quy hoạch động để giải bài toán TSP đã trình bày trong mục 3.2.4.

Đặt S = $\{x_1, x_2, \ldots, x_k\}$ là tập hợp con các cạnh của đồ thị G = (V,E). Ta nói rằng một đường đi P từ v đến w *phủ lên* S nếu P = $\{v, x_1, x_2, \ldots, x_k, w\}$, trong đó xi có thể xuất hiện ở một thứ tự bất kì, nhưng chỉ xuất hiện duy nhất một lần. Ví dụ đường cho trong hình sau, đi từ a đến a, phủ lên {c, d, e, g}.



Hình 3-6: Đường đi từ a đến a phủ lên {c, d, e, g}

Ta định nghĩa d(v, w, S) là tổng độ dài của đường đi ngắn nhất từ v đến w, phủ lên S. Nếu không có một đường đi như vậy thì đặt d(v, w, S) = ∞. Một chu trình Hamilton nhỏ nhất C_{min} của G phải có tổng độ dài là $c(C_{min}) = d(v,v, V - \{v\})$. Trong đó v là một đỉnh nào đó của V. Ta xác định C_{min} như sau:

Nếu |V| = 1 (G chỉ có một đỉnh) thì $c(C_{min}) = 0$, ngược lại ta có công thức đệ qui để tính d(v, w, S) là:

$d(v, w, \{\}) = c(v,w)$

$d(v, w, S) = \min [c(v, x) + d(x, w, S - \{x\}]$, lấy với mọi $x \in S$.

Trong đó c(v, w) là độ dài của cạnh nối hai đỉnh v và w nếu nó tồn tại hoặc là ∞ nếu ngược lại. Dòng thứ hai trong công thức đệ qui trên ứng với tập S không rỗng, nó chỉ ra rằng đường đi ngắn nhất từ v đến w phủ lên S, trước hết phải đi đến một đỉnh x nào đó trong S và sau đó là đường đi ngắn nhất từ x đến w, phủ lên tập S – {x}.

Bằng cách lưu trữ các đỉnh x trong công thức đệ qui nói trên, chúng ta sẽ thu được một chu trinh Hamilton tối tiểu.

3.5 KĨ THUẬT QUAY LUI

Kĩ thuật quay lui (backtracking) như tên gọi của nó, 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 cứ 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.

Ở đây chúng ta sẽ xét 3 kĩ thuật quay lui: "vét cạn" là kĩ thuật phải đi tới tất cả các điểm dừng rồi mới quay lui. "Cắt tỉa Alpha-Beta" và "Nhánh-Cận" là hai kĩ thuật cho phép chúng ta 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. Các kĩ thuật này sẽ được trình bày thông qua một số bài toán cụ thể sau.

3.5.1 Định trị cây biểu thức số học

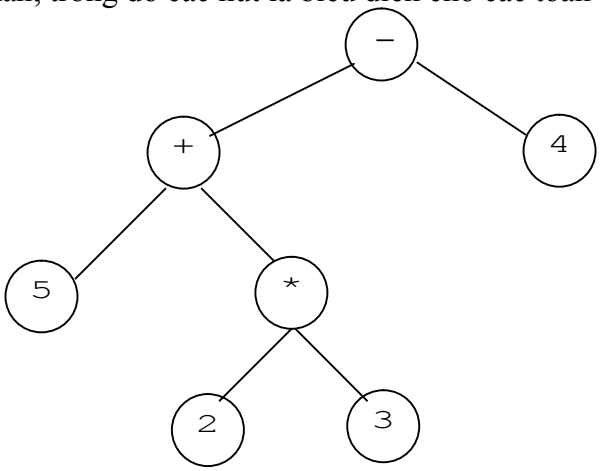
Trong các ngôn ngữ lập trình đều có các biểu thức số học, việc dịch các biểu thức này đòi hỏi phải đánh giá (định trị) chúng. Để làm được điều đó cần phải có một biểu diễn trung gian cho biểu thức. Một trong các biểu diễn trung gian cho biểu thức là cây biểu thức.

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.

Ví dụ 3-3: Biểu thức 5 + 2 * 3 - 4 sẽ được biểu diễn bởi cây trong hình 3-8

Trị của một 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.



Hình 3-7: Một cây biểu thức số học

Như vậy để định trị cho nút gốc, chúng ta phải 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 kĩ thuật quay lui vét cạn, vì chúng ta phải lần đến tất cả các nút lá mới định trị được cho các nút trong và do thế mới định trị được cho nút gốc.

Ví dụ 3-4: Với cây biểu thức trong ví dụ 3-3. Để định trị cho nút - chúng ta phải định trị cho nút + và nút 4. Nút 4 là nút lá nên giá trị của nó là 4. Để định trị cho nút + ta phải định trị cho nút 5 và nút *. Nút 5 là nút lá nên giá trị của nó là 5. Để định trị cho nút *, ta phải định trị cho nút 2 và nút 3. Cả hai nút này đều là lá nên giá trị của chúng tương ứng là 2 và 3. Quay lui lại nút *, lấy toán tử * áp dụng cho hai con của nó là 2 và 3 ta được trị của nút * là 6. Quay lui về nút +, lại áp dụng toán tử + vào hai con của nó là 5 và 6 được trị của nút + là 11. Cuối cùng quay về nút -, áp dụng toán tử - vào hai con của nó là 11 và 4 ta được trị của nút - (nút gốc) là 7. Đó chính là trị của biểu thức. Trong hình 3-9î, mũi tên nét đứt minh họa quá trình đi tìm nút lá và mũi tên nét liền minh họa quá trình quay lui để định trị cho các nút, các số bên phải mỗi nút là trị của nút đó.

Giải thuật sơ bộ để định trị một nút bất kỳ như sau:

```
FUNCTION Eval(n : node): real;
BEGIN
     IF n là lá THEN RETURN (trị của toán hạng trong n)
     ELSE RETURN (Toán tử trong n (Eval (Con trái của n),
     Eval (Con phải của n)) );
END;
```

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

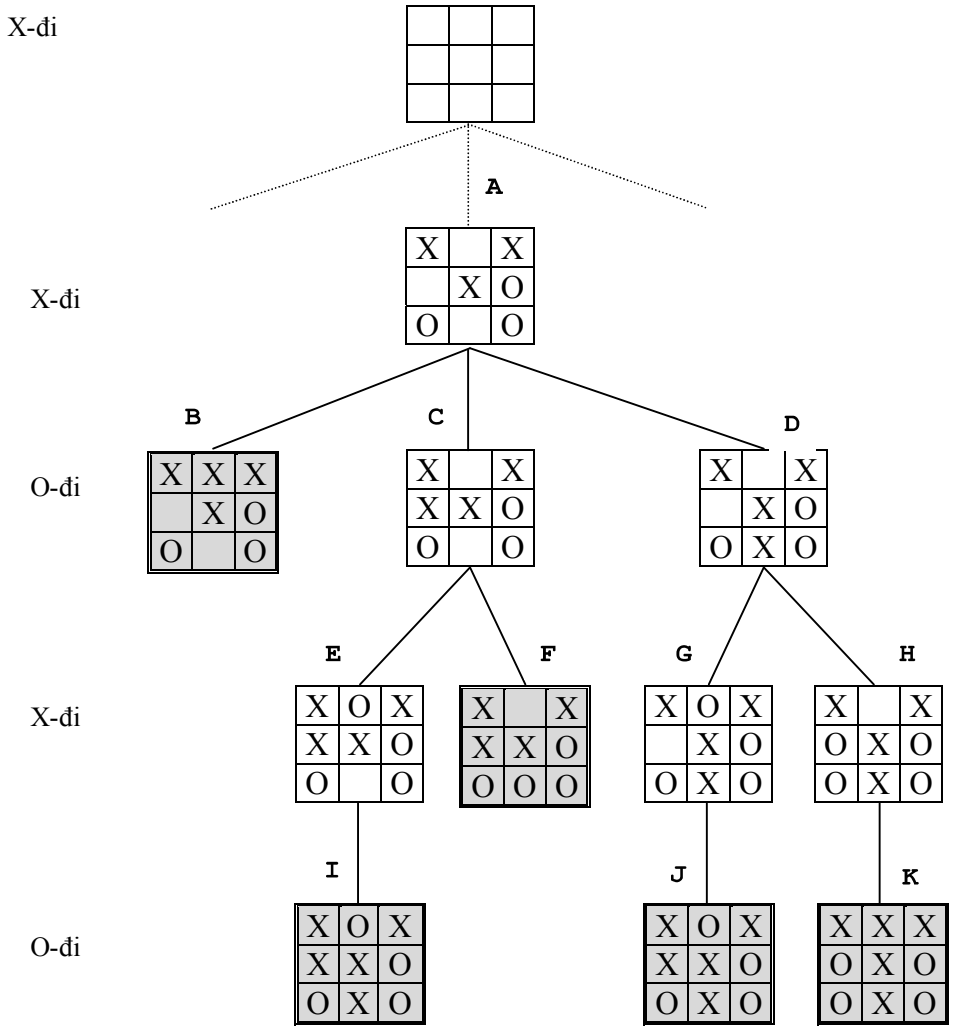
3.5.2 Kĩ thuật cắt tỉa Alpha-Beta

3.5.2.1 Cây trò chơi

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.

Một trò chơi như vậy có thể được biểu diễn bởi một cây, gọi là 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.

Ví dụ 3-5: 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 sau:



Hình 3-8: Một phần của cây trò chơi carô 9 ô

Trong cây trò chơi trên, 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.

Như vậy từ một trạng thái bất kỳ, đến lượt mình, 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. 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.

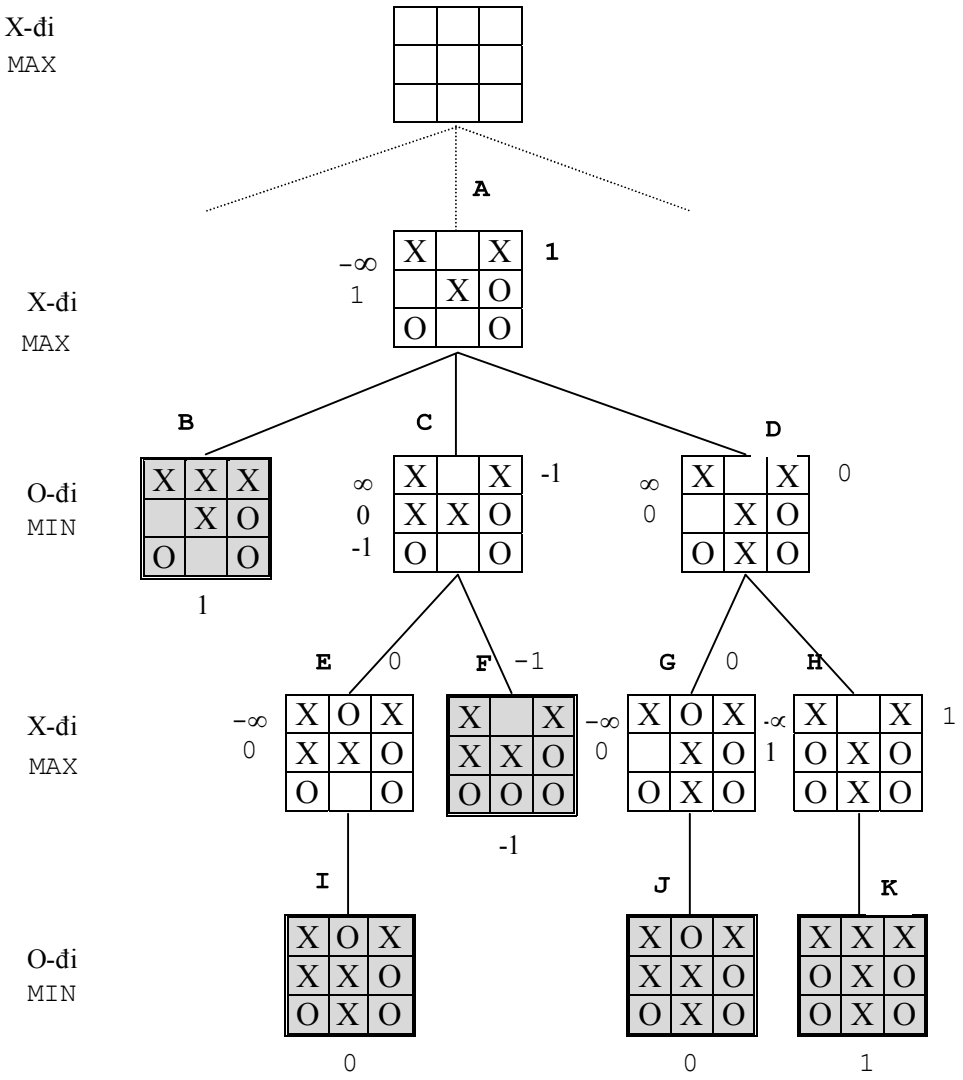
Quy tắc định trị này cũng gần giống với quy tắc định trị cho cây biểu thức số học, điểm khác biệt ở đây là các toán tử là các hàm lấy max hoặc min và mỗi nút có thể có nhiều con. Do vậy ta có thể dùng kĩ thuật quay lui để định trị cho các nút của cây trò chơi.

Ví dụ 3-6: Vận dụng quy tắc quay lui vét cạn để định trị cho nút A trong cây trò chơi trong ví dụ 3-5.

Trước hết ta gán trị cho các nút lá, theo qui định trên thì nút lá B được gán giá trị 1, vì tại đó người đánh X đã thắng. Nút F được gán giá trị -1 vì tại đó người đánh X đã thua (người đánh O đã thắng). Nút I được gán giá trị 0 vì tại đó hai người hòa nhau. Tương tự nút J được gán giá trị 0 và nút K được gán giá trị 1.

Vì người đánh X được gán giá trị 1 tại nút lá mà anh ta đã thắng (giá trị lớn nhất) nên ta nói X chọn nước đi MAX, ngược lại người đánh O sẽ chọn nước đi MIN.

Để định trị cho nút A, ta thấy A là nút MAX và không phải là nút lá nên ta gán giá trị tạm là -∞, xét B là con của A, B là nút lá nên giá trị của nó là giá trị đã được gán 1, giá trị tạm của A bây giờ là max(-∞,1) = 1. Xét con C của A, C là nút MIN, giá trị tạm lúc đầu của C là ∞. Xét con E của C, E là nút MAX, giá trị tạm của E là -∞. Xét con I của E, I là nút lá nên giá trị của nó là 0. Quay lui lại E, giá trị tạm của E bây giờ là max(-∞,0) = 0. Vì E chỉ có một con là I đã xét nên giá trị tạm 0 trở thành giá trị của E. Quay lui lại C, giá trị tạm mới của C là min(∞,0) = 0. Lại xét con F của C, vì F là nút lá, nên giá trị của F đã được gán là –1. Quay lui lại C, giá trị tạm mới của C là min(0,-1) = -1. Nút C có hai con là E và F, cả hai con này đều đã được xét, vậy giá trị tạm -1 của C trở thành giá trị của nó. Sau khi có giá trị của C, ta phải quay lại A và đặt lại giá trị tạm của A là max(1,-1) = 1. Tiếp tục xét nút D, D là nút MIN nên giá trị tạm là ∞, xét nút con G của D, G là nút MAX nên giá trị tạm của nó là -∞, xét nút con J của G. Vì J là nút lá nên có giá trị 0. Quay lui lại G, giá trị tạm của G bây giờ là max(-∞,0) = 0 và giá trị tạm này trở thành giá trị của G vì G chỉ có một con J đã xét. Quay lui về D, giá trị tạm của D bây giờ là min(∞,0) = 0. Lại xét con H của D, H là nút MAX nên gán giá trị tạm ban đầu là -∞. Xét con K của H, nút K là nút lá nên giá trị của K đã được gán là 1. Quay lui về H và đặt lại giá trị tạm của H là max(-∞,1) = 1. Giá trị tạm này chính là giá trị của H vì H chỉ có một con K đã được xét. Quay lui về D và đặt lại giá trị tạm của D là min(0, 1) = 0. Cả hai con G và H của D đều đã được xét nên giá trị tạm 0 của D trở thành giá trị của nó. Quay lui về A, giá trị tạm của nó là max(1,0) = 1vẫn không thay đổi, nhưng lúc này cả 3 con của A đều đã được xét nên giá trị tạm 1 trở thành giá trị của A. Kết quả được minh họa trong hình sau:



Hình 3-9: Định trị cây trò chơi bằng kĩ thuật quay lui vét cạn

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

3.5.2.2 Giải thuật vét cạn định trị cây trò chơi

Để cài đặt ta có một số giả thiết sau:

- Ta có một hàm Payoff nhận vào một nút lá và cho ta giá trị của nút lá đó.
- Các hằng ∞ và -∞ tương ứng là các trị Payoff lớn nhất và nhỏ nhất.
- Khai báo kiểu ModeType = (MIN, MAX) để xác định định trị cho nút là MIN hay MAX.

- Một kiểu NodeType được khai báo một cách thích hợp để biểu diễn cho một nút trên cây phản ánh một trạng thái của cuộc chơi.
- Ta có một hàm is_leaf để xác định xem một nút có phải là nút lá hay không?
- Hàm max và min tương ứng lấy giá trị lớn nhất và giá trị nhỏ nhất của hai giá trị.

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 là -∞ hoặc ∞ tùy thuộc n là nút MAX hay MIN và xét con của n. Sau khi một con của n có giá trị V thì đặt lại value = max(value,V) nếu n là nút MAX và value = min(value,V) nếu n là nút MIN. Khi tất cả các con của n đã được xét thì giá trị tạm value của n trở thành giá trị của nó.

```
FUNCTION Search(n : NodeType; mode: ModeType): real;
VAR  C : NodeType ;  { C là một nút con của nút n}
     Value : real;
{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 }
BEGIN
     IF is_leaf(n) THEN RETURN ( Payoff(n) )
     ELSE BEGIN
          {Khởi tạo giá trị tạm cho n }
          IF mode = MAX THEN value := -∞ ELSE value := ∞;

{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}

          FOR với mỗi con C của n DO
               IF mode = MAX THEN
                    Value := max(Value, Search(C, MIN) )
               ELSE Value := min(Value, Search(C, MAX) );
          RETURN (value);
     END;
END;
```

3.5.2.3 Kĩ thuật cắt tỉa Alpha-Beta (Alpha-Beta Pruning)

Trong giải thuật vét cạn ở trên, ta thấy 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.

Chúng ta cố gắng tìm một cách sao cho khi định trị một nút thì không nhất thiết phải định trị cho tất cả các nút con cháu của nó. Trước hết ta có nhận xét như sau:

Nếu P là một nút MAX và ta đang xét một nút con Q của nó (dĩ nhiên Q là nút MIN). Giả sử Vp là một giá trị tạm của P, Vq là một giá trị tạm của Q và nếu ta có Vp ≥ Vq thì ta không cần xét các con chưa xét của Q nữa. Vì nếu có xét thì giá trị của Q cũng sẽ nhỏ hơn hoặc bằng Vq và do đó không ảnh hưởng gì đến Vp. Tương tự nếu P là nút MIN (tất nhiên Q là nút MAX) và Vp ≤ Vq thì ta cũng không cần xét đến các con chưa xét của Q nữa. Việc không xét tiếp các con chưa được xét của nút Q gọi là việc cắt tỉa Alpha-Beta các con của nút Q.

Trên cơ sở nhận xét đó, ta nêu ra quy tắc định trị cho một nút không phải là nút lá trên cây như sau:

1. Khởi đầu nút MAX có giá trị tạm là -∞ và nút MIN có giá trị tạm là ∞.

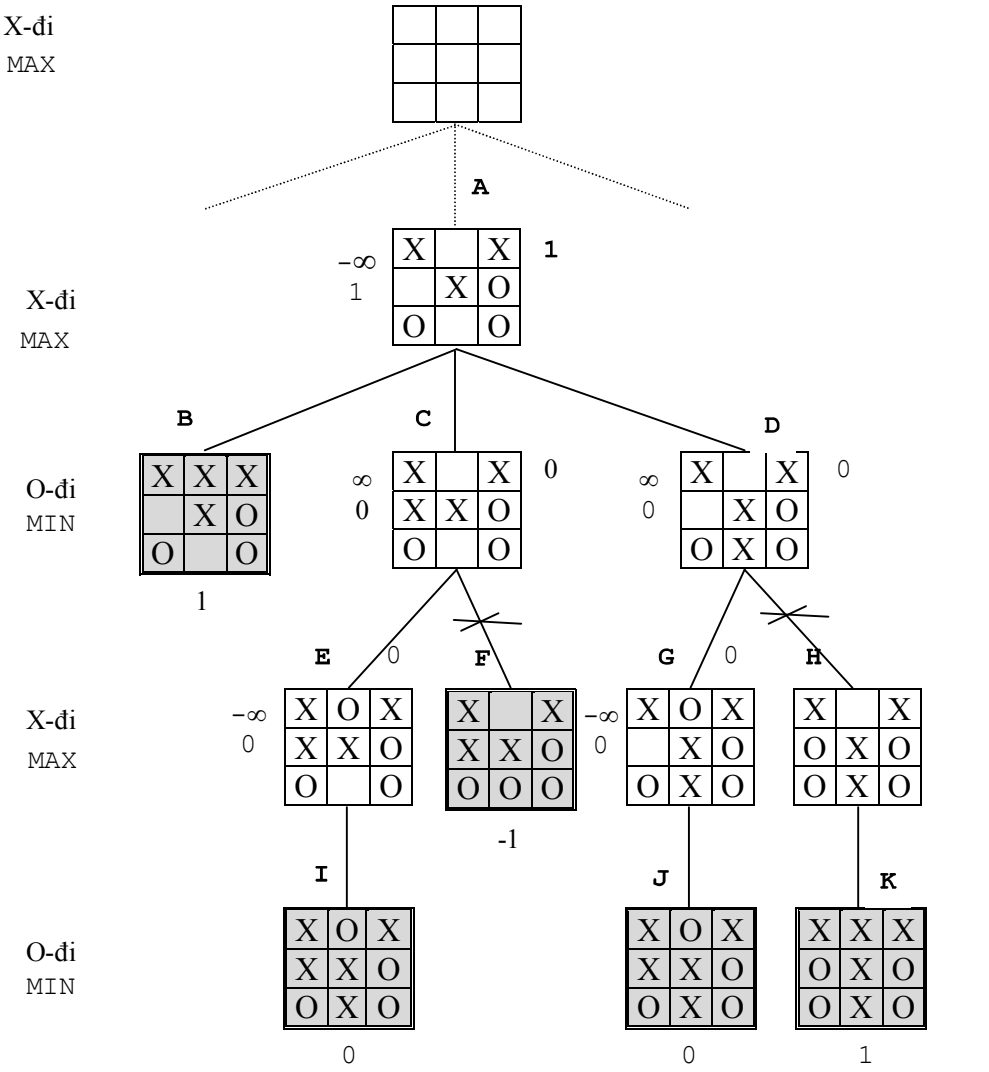
2. Nếu tất cả các nút con của một nút đã được xét hoặc bị cắt tỉa thì giá trị tạm của nút đó trở thành giá trị của nó.

3. Nếu một nút MAX n có giá trị tạm là V1 và một nút con của nó có giá trị là V2 thì đặt giá trị tạm mới của n là max(V1,V2). Nếu n là nút MIN thì đặt giá trị tạm mới của n là min(V1,V2).

4. Vận dụng quy tắc cắt tỉa Alpha-Beta nói trên để hạn chế số lượng nút phải xét.

Ví dụ 3-7: Vận dụng quy tắc trên để định trị cho nút A của cây trò chơi trong ví dụ 3-5.

A là nút MAX, vì A không phải là nút lá nên ta gán giá trị tạm là -∞, xét B là con của A, B là nút lá nên giá trị của nó là giá trị đã được gán 1, giá trị tạm của A bây giờ là max(-∞,1) = 1. Xét con C của A, C là nút MIN, giá trị tạm lúc đầu của C là ∞. Xét con E của C, E là nút MAX, giá trị tạm của E là -∞. Xét con I của E, I là nút lá nên giá trị của nó là 0. Quay lui lại E, giá trị tạm của E bây giờ là max(-∞,0) = 0. Vì E chỉ có một con là I đã xét nên giá trị tạm 0 trở thành giá trị của E. Quay lui lại C, giá trị tạm mới của C là min(∞,0) = 0. A là nút MAX có giá trị tạm là 1, C là con của A, có giá trị tạm là 0, 1>0 nên ta không cần xét con F của C nữa. Nút C có hai con là E và F, trong đó E đã được xét, F đã bị cắt, vậy giá trị tạm 0 của C trở thành giá trị của nó. Sau khi có giá trị của C, ta phải đặt lại giá trị tạm của A, nhưng giá trị tạm này không thay đổi vì max(1,0) = 1. Tiếp tục xét nút D, D là nút MIN nên giá trị tạm là ∞, xét nút con G của D, G là nút MAX nên giá trị tạm của nó là -∞, xét nút con J của G. Vì J là nút lá nên có giá trị 0. Quay lui lại G, giá trị tạm của G bây giờ là max(-∞,0) = 0 và giá trị tạm này trở thành giá trị của G vì G chỉ có một con J đã xét. Quay lui về D, giá trị tạm của D bây giờ là min(∞,0) = 0. Giá trị tạm này của D nhỏ hơn giá trị tạm của nút A MAX là cha của nó nên ta cắt tỉa con H chưa được xét của D và lúc này D có giá trị là 0. Quay lui về A, giá trị tạm của nó vẫn không thay đổi, nhưng lúc này cả 3 con của A đều đã được xét nên giá trị tạm 1 trở thành giá trị của A. Kết quả được minh họa trong hình sau:



Hình 3-10: Định trị cây trò chơi bằng kĩ thuật cắt tỉa alpha-beta

Hàm cat_tia sau trình bày giải thuật thô để định trị một nút, áp dụng kĩ thuật cắt tỉa alpha-beta

```
FUNCTION cat_tia(Q:NodeType; mode:ModeType; Vp: real): real;
var  C : NodeType ;  { C là một nút con của nút Q}
     Vq : real;
{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}
BEGIN
     IF is_leaf(Q) THEN RETURN ( Payoff(Q) )
     ELSE BEGIN
          { Khởi tạo giá trị tạm cho Q }
          IF mode = MAX THEN 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 DO
               IF mode = MAX THEN BEGIN
                    Vq:= max(Vq, Cat_tia(C, MIN, Vq));
                    IF Vp<=Vq THEN RETURN(Vq);
               END
               ELSE BEGIN
                    Vq := min(Vq, Cat_tia(C, MAX, Vq));
                    IF Vp >= Vq THEN RETURN(Vq);
               END;
          RETURN (Vq);
     END;
END;
```

3.5.3 Kĩ thuật nhánh cận

Với các bài toán tìm phương án tối ưu, nếu chúng ta xét hết tất cả các phương án thì mất rất nhiều thời gian, nhưng nếu sử dụng phương pháp tham ăn thì phương án tìm được chưa hẳn đã là phương án tối ưu. Nhánh cận là kĩ 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.

Cây tìm kiếm phương án có nút gốc biểu diễn cho tập tất cả các phương án có thể có, mỗi nút lá biểu diễn cho một phương án nào đó. Nút n có các nút con tương ứng với các khả năng có thể lựa chọn tập phương án xuất phát từ n. Kĩ thuật này gọi là phân nhánh.

Vói 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.

Để dễ hình dung ta sẽ xét hai bài toán quen thuộc là bài toán TSP và bài toán cái ba lô.

3.5.3.1 Bài toán đường đi của người giao hàng

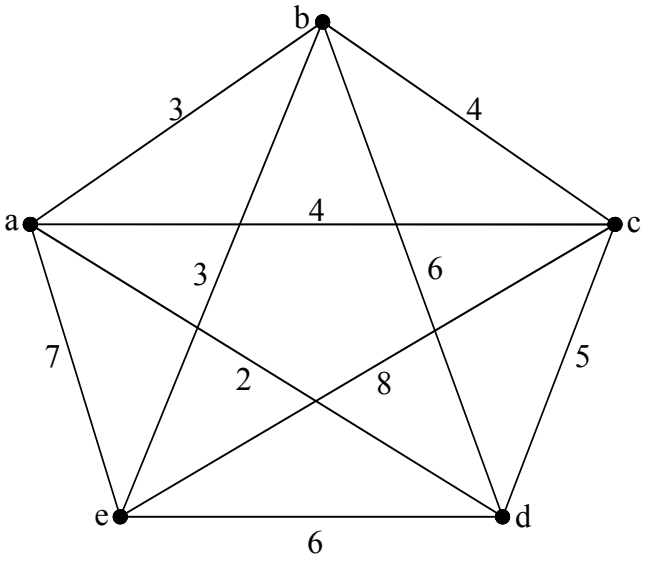
3.5.3.1.1 Phân nhánh

Cây tìm kiếm phương án là cây nhị phân trong đó:

- Nút gốc là nút biểu diễn cho cấu hình bao gồm tất cả các phương án.
- Mỗi nút sẽ có hai con, con trái biểu diễn cho cấu hình bao gồm tất cả các phương án chứa một cạnh nào đó, con phải biểu diễn cho cấu hình bao gồm tất cả các phương án không chứa cạnh đó (các cạnh để xét phân nhánh được thành lập tuân theo một thứ tự nào đó, chẳng hạn thứ tự từ điển).
- Mỗi nút sẽ kế thừa các thuộc tính của tổ tiên của nó và có thêm một thuộc tính mới (chứa hay không chứa một cạnh nào đó).

- Nút lá biểu diễn cho một cấu hình chỉ bao gồm một phương án.
- Để quá trình phân nhánh mau chóng tới nút lá, tại mỗi nút ta cần có một quyết định bổ sung dựa trên nguyên tắc là mọi đỉnh trong chu trình đều có cấp 2 và không tạo ra một chu trình thiếu.

Ví dụ 3-7: Xét bài toán TSP có 5 đỉnh với độ dài các cạnh được cho trong hình 3-11.



Hình 3-11: Bài toán TSP có 5 đỉnh

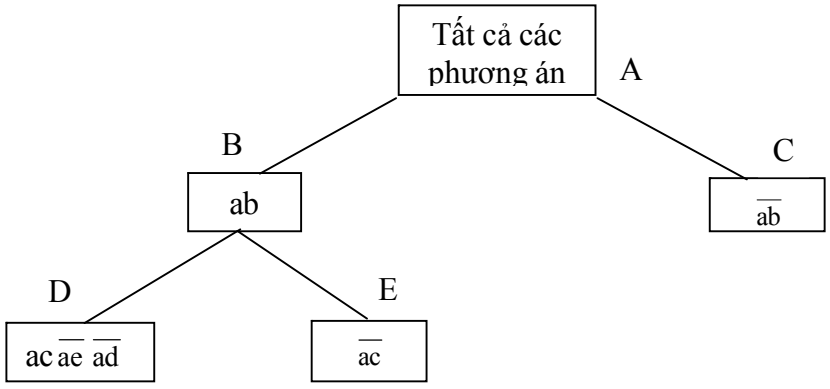
Các cạnh theo thứ tự từ điển để xét là:

ab, ac, ad, ae, bc, bd, be, cd, ce và de.

Nút gốc A của cây bao gồm tất cả các phương án.
Hai con của A là B và C, trong đó B bao gồm tất cả các phương án chứa cạnh ab, C bao gồm tất cả các phương án không chứa ab, kí hiệu là $\overline{ab}$

Hai con của B là D và E. Nút D bao gồm tất cả các phương án chứa ac. Vì các phương án này vừa chứa ab (kế thừa của B) vừa chứa ac nên đỉnh a đã đủ cấp hai nên D không thể chứa ad và ae. Nút E bao gồm tất cả các phương án không chứa ac…

Ta được cây (chưa đầy đủ) trong hình 3-12.



Hình 3-12: Phân nhánh

3.5.3.1.2 Tính cận dưới

Đây là bài toán tìm min nên ta sử dụng cận dưới. Cận dưới tại mỗi nút là một số nhỏ hơn hoặc bằng giá của tất cả các phương án được biểu diễn bởi nút đó. Giá của một phương án ở đây là tổng độ dài của một chu trình.

Để tính cận dưới của nút gốc, mỗi đỉnh ta chọn hai cạnh có độ dài nhỏ nhất. Cận dưới của nút gốc bằng tổng độ dài tất cả các cạnh được chọn chia cho 2.

Ví dụ 3-8: Với số liệu cho trong ví dụ 3-7 nói trên, ta tính cận dưới của nút gốc A (hình 3-12) như sau:

- Đỉnh a chọn ad = 2, ab = 3
- Đỉnh b chọn ba = 3, be = 3
- Đỉnh c chọn ca = 4, cb = 4
- Đỉnh d chọn da = 2, dc = 5
- Đỉnh e chọn eb = 3, ed = 6

Tổng độ dài các cạnh được chọn là 35, cận dưới của nút gốc A là 35/2 = 17.5

Đối với các nút khác, chúng ta phải lựa chọn hai cạnh có độ dài nhỏ nhất thỏa điều kiện ràng buộc (phải chứa cạnh này, không chứa cạnh kia).

Ví dụ 3-9: Tính cận dưới cho nút D trong hình 3-13. Điều kiện ràng buộc là phải chứa ab, ac và không chứa ad, ae.

- Đỉnh a chọn ab = 3, ac = 4, do hai cạnh này buộc phải chọn.
- Đỉnh b chọn ba = 3, be = 3
- Đỉnh c chọn ca = 4, cb = 4
- Đỉnh d chọn de = 6, dc = 5, do không được chọn da nên ta phải chọn de.
- Đỉnh e chọn eb = 3, ed = 6

Tổng độ dài các cạnh được chọn là 41, cận dưới của nút D là 41/2 = 20.5

3.5.3.1.3 Kĩ thuật nhánh cận

Bây giờ ta sẽ kết hợp hai kĩ thuật trên để xây dựng cây tìm kiếm phương án. Quy tắc như sau:

- Xây dựng nút gốc, bao gồm tất cả các phương án, tính cận dưới cho nút gốc.
- Sau khi phân nhánh cho mỗi nút, ta tính cận dưới cho cả hai con.
- Nếu cận dưới của một nút con lớn hơn hoặc bằng giá nhỏ nhất tạm thời của một phương án đã được tìm thấy thì ta không cần xây dựng các cây con cho nút này nữa (Ta gọi là cắt tỉa các cây con của nút đó).
- Nếu cả hai con đều có cận dưới nhỏ hơn giá nhỏ nhất tạm thời của một phương án đã được tìm thấy thì nút con nào có cận dưới nhỏ hơn sẽ được ưu tiên phân nhánh trước.
- Mỗi lần quay lui để xét nút con chưa được xét của một nút ta phải xem xét lại nút con đó để có thể cắt tỉa các cây của nó hay không vì có thể một phương án có giá nhỏ nhất tạm thời vừa được tìm thấy.

- Sau khi tất cả các con đã được phân nhánh hoặc bị cắt tỉa thì phương án có giá nhỏ nhất trong các phương án tìm được là phương án cần tìm.

Trong quá trình xây dựng cây có thể ta đã xây dựng được một số nút lá, như ta biết mỗi nút lá biểu diễn cho một phương án. Giá nhỏ nhất trong số các giá của các phương án này được gọi là giá nhỏ nhất tạm thời.

Ví dụ 3-10: Xét bài toán TSP trong ví dụ 3-7 nói trên.

Tập hợp các cạnh để xét phân nhánh là ab, ac, ad, ae, bc, bd, be, cd, ce và de. Điều kiện bổ sung ở đây là mỗi đỉnh phải được chọn hai cạnh, bị loại hai cạnh và không được tạo ra chu trình thiếu.

Nút gốc A bao gồm tất cả các phương án, có cận dưới là 17.5. Phân nhánh cho A, xây dựng hai con là B và C. Tính cận dưới cho hai nút này được cận dưới của B là 17.5 và C là 18.5. Nút B có cận dưới nhỏ hơn nên được phân nhánh trước. Hai con của B là D và E. Các ràng buộc của D và E giống nh-ư ta đã nói trong ví dụ của phần phân nhánh. Tính cận cho D và E, được cận dưới của D là 20.5 và của E là 18. Nút E được xét trước. Phân nhánh cho nút E theo cạnh ad, hai con của E là F và G. F chứa ad và G không chứa ad. Do F kế thừa các thuộc tính của E và B, nên F là tập hợp các phương án chứa ab, ad và không chứa ac, đỉnh a đã đủ cấp 2 vậy F không chứa ae. Tương tự G chứa ab, không chứa ac, không chứa ad nên phải chứa ae. Tính cận dưới cho F và G được cận dưới của F là 18 và của G là 23. Tiếp tục xây dựng hai con cho F theo cạnh bc là H và I. H chứa bc và I không chứa bc. Do H kế thừa các thuộc tính của B, E và F nên H là các phương án chứa ab, ad, không chứa ac và chứa bc. Như vậy đỉnh a đã thỏa điều kiện là được chọn hai cạnh (ab và ad) và bị loại hai cạnh (ac và ae), Đỉnh b đã được chọn 2 cạnh (ba và bc) nên bd và be bị loại. Đỉnh c đã được chọn cb, bị loại ca, ta có thể chọn cd hoặc ce. Nếu chọn cd thì sẽ có một chu trinh thiếu a b c d a, như vậy cd bị loại nên phải chọn ce. Đỉnh d có db và dc đã bị loại, da đã được chọn nên phải chọn thêm de. Lúc đó đỉnh e cũng đã có hai cạnh được chọn là ec và ed, hai cạnh bị loại là eb và ea. Tóm lại H là tập chỉ bao gồm một phương án a b c e d a có giá là 23. Đối với I ta đã có I chứa ab, không chứa ac, chứa ad, không chứa ae và không chứa bc. Bằng lý luận tương tự ta có I không chứa bd, chứa be, cd, ce và không chứa de. Một phương án mới là a b e c d a với giá 21. Đây là giá nhỏ nhất tạm thời mới được tìm thấy.

Bây giờ ta quay lui về E và xét nút con của nó là G. Vì G có cận dưới là 23 lớn hơn giá thấp nhất tạm thời 21 nên cắt tỉa các con của G.

Quay lui về B và xét nút con D của nó. Cận dưới của D là 20.5 không lớn hơn 21. Nhưng vì độ dài các cạnh trong bài toán đã cho là số nguyên nên nếu ta triển khai các con của D tới nút lá gồm một phương án. Giá của phương án này phải là một số nguyên lớn hơn 20.5 hay lớn hơn hoặc bằng 21. Vậy ta cũng không cần xây dựng các con của D nữa.

Tiếp tục quay lui đến A và xét con C của nó. Phân nhánh C theo cạnh ac thành hai con J và K. J chứa ac có cận dưới là 18.5. K không chứa ac nên phải chứa ad và ae, cận dưới của K là 21 bằng giá nhỏ nhất tạm thời nên cắt tỉa các con của K.

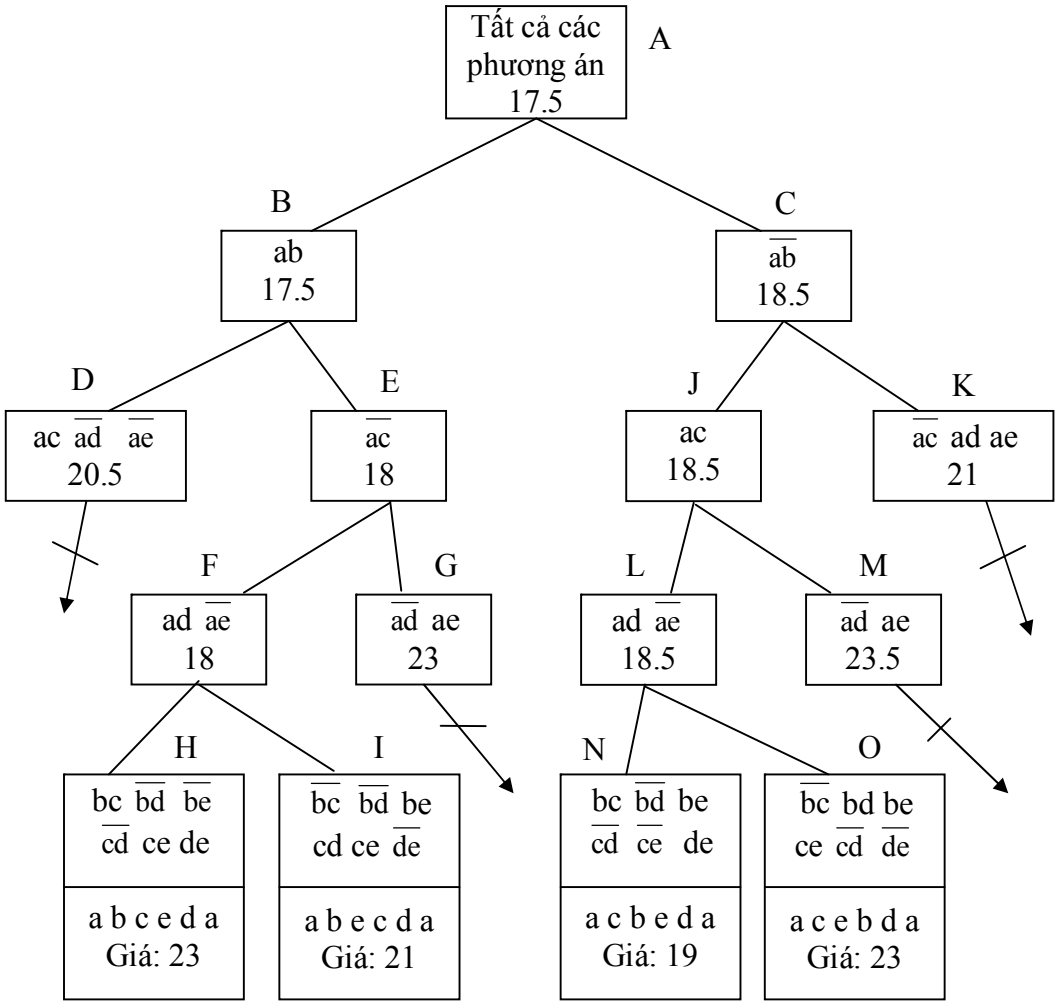
Hai con của J là L và M. M không chứa ad, ab, chứa ac và ae có cận dưới 23.5 nên bị cắt tỉa các con. Hai con của L là N và O, N chứa bc và O không chứa bc.

Xét nút N ta có: Đỉnh a được chọn hai cạnh ac và ad, bị loại hai cạnh ab và ae. Đỉnh b đã được chọn bc, bị loại ba, ta có thể chọn bd hoặc be. Nếu chọn bd thì sẽ có một chu trình thiếu là a c b d a, vậy phải loại bd và chọn be. Đỉnh c đã được chọn ca, cb nên phải loại cd và ce. Đỉnh d đã được chọn da, bị loại db và dc nên phải chọn de. Khi đó đỉnh e có đủ hai cạnh được chọn là eb, ed và hai cạnh bị loại là ea và ec. Vậy N bao gồm chỉ một phương án là a c b e d a với giá 19.

Tương tự nút O bao gồm chỉ một phương án a c e b d a có giá là 23.

Tất cả các nút con của cây đã được xét hoặc bị cắt tỉa nên phương án cần tìm là a c b e d a với giá 19.

Hình 3-13 minh họa cho những điều ta vừa nói.



Hình 3-13: Kĩ thuật nhánh cận giải bài toán TSP

3.5.3.2 Bài toán cái ba lô

Ta thấy đây là một bài toán tìm max. Danh sách các đồ vật được sắp xếp theo thứ tự giảm của đơn giá để xét phân nhánh.

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 * Đơn giá lớn nhất.

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 (của nút cha) + số đồ vật được chọn * giá trị mỗi vật.
   - W = W (của nút cha) - số đồ vật được chọn * trọng lượng mỗi vật.
   - CT = TGT + W * 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.

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.

Ví dụ 3-11: Với bài toán cái ba lô đã cho trong ví dụ 3-2 , sau khi tính đơn giá cho các đồ vật và sắp xếp các đồ vật theo thứ tự giảm dần của đơn giá ta được bảng sau.

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

Gọi X_A, X_B, X_C, X_D là số lượng cần chọn tương ứng của các đồ vật a, b, c d.

Nút gốc A biểu diễn cho trạng thái ta chưa chọn bất cứ một đồ vật nào. Khi đó tổng giá trị TGT =0, trọng lượng của ba lô W=37 (theo đề ra) và cận trên CT = 37*2.5 = 92.5, trong đó 37 là W, 2.5 là đơn giá của đồ vật b.

Với đồ vật b, ta có 4 khả năng: chọn 3 đồ vật b (X_B=3), chọn 2 đồ vật b (X_B=2), chọn 1 đồ vật b (X_B=1) và không chọn đồ vật b (X_B=0). Ứng với 4 khả năng này, ta phân nhánh cho nút gốc A thành 4 con B, C, D và E.

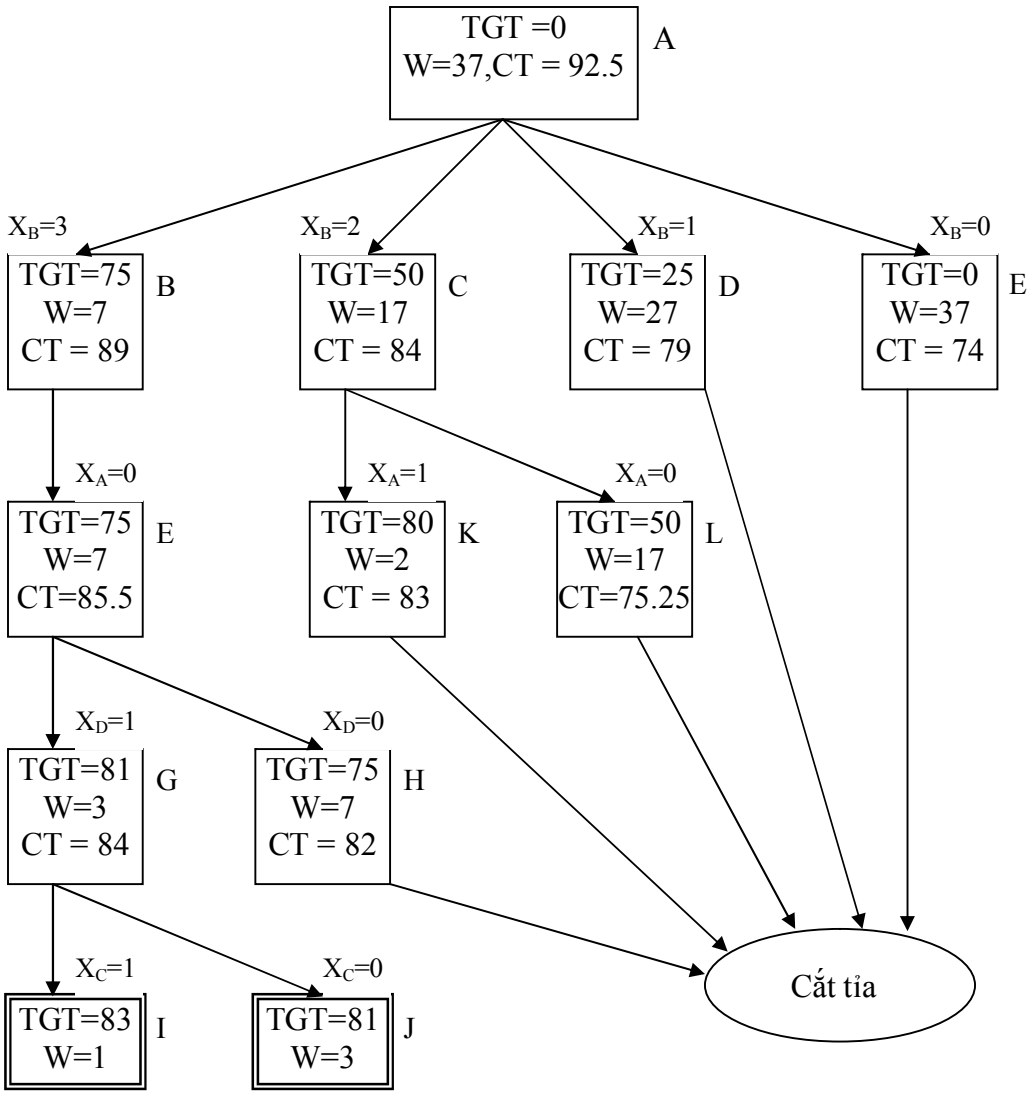
Với nút con B, ta có TGT = 0+ 3*25 = 75, trong đó 3 là số vật b được chọn, 25 là giá trị của mỗi đồ vật b. W = 37- 3*10 = 7, trong đó 37 là trọnh lượng ban đầu của ba lô, 3 là số vật b được, 10 là trọng lượng mõi đồ vật b. CT = 75 + 7*2 = 89, trong đó 75 là TGT, 7 là trọng lượng còn lại của ba lô và 2 là đơn giá của đồ vật a. Tương

tự ta tính được các thông số cho các nút C, D và E, trong đó cận trên tương ứng là 84, 79 và 74.

Trong các nút B, C, D và E thì nút B có cận trên lớn nhất nên ta sẽ phân nhánh cho nút B trước với hy vọng sẽ có được phương án tốt từ hướng này. Từ nút B ta chỉ có một nút con F duy nhất ứng với X_A=0 (do trọng lượng còn lại của ba lô là 7, trong khi trọng lượng của mỗi đồ vật a là 15). Sau khi xác định các thông số cho nút F ta có cận trên của F là 85.5. Ta tiếp tục phân nhánh cho nút F. Nút F có 2 con G và H tương ứng với X_D=1 và X_D=0. Sau khi xác định các thông số cho hai nút này ta thấy cận trên của G là 84 và của H là 82 nên ta tiếp tục phân nhánh cho nút G. Nút G có hai con là I và J tương ứng với X_C=1 và X_C=0. Đây là hai nút lá (biểu diễn cho phương án) vì với mỗi nút thì số các đồ vật đã được chọn xong. Trong đó nút I biểu diễn cho phương án chọn X_B=3, X_A=0, X_D=1 và X_C=1 với giá 83, trong khi nút J biểu diễn cho phương án chọn X_B=3, X_A=0, X_D=1 và X_C=0 với giá 81. Như vậy giá lớn nhất tạm thời ở đây 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, ta thấy cận trên của C là 84>83 nên tiếp tục phân nhánh cho nút C. Nút C có hai con là K và L ứng với X_A=1 và X_A=0. Sau khi tính các thông số cho K và L ta thấy cận trên của K là 83 và của L là 75.25. Cả hai giá trị này đều không lớn hơn 83 nên cả hai nút này đều bị cắt tỉa. Cuối cùng 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 nên phương án tốt nhất tạm thời là phương án cần tìm. Theo đó ta cần chọn 3 đồ vật loại b, 1 đồ vật loạ d và một đồ vật loại c với tổng giá trị là 83, tổng trọng lượng là 36. Xem minh hoạ trong hình 3-14.



Hình 3-14: Kĩ thuật nhánh cận áp dụng cho bài toán cái ba lô

3.6 KĨ THUẬT TÌM KIẾM ĐỊA PHƯƠNG

3.6.1 Nội dung kĩ thuật

Kĩ thuật tìm kiếm địa phương (local search) thường được áp dụng để giải các bài toán tìm lời giải tối ưu. Phương pháp như sau:

- Xuất phát từ một phương án nào đó.
- Áp dụng một phép biến đổi lên phương án hiện hành để được một phương án mới tốt hơn phương án đã có.
- Lặp lại việc áp dụng phép biến đổi lên phương án hiện hành cho đến khi không còn có thể cải thiện được phương án nữa.

Thông thường một phép biến đổi chỉ thay đổi một bộ phận nào đó của phương án hiện hành để được một phương án mới nên phép biến đổi được gọi là phép biến đổi địa phương và do đó ta có tên kĩ thuật tìm kiếm địa phương. Sau đây ta sẽ trình bày một số ví dụ áp dụng kĩ thuật tìm kiếm địa phương.

3.6.2 Bài toán cây phủ tối thiểu

Cho G = (V,E) là một đồ thị vô hướng liên thông, trong đó V là tập các đỉnh và E là tập các cạnh. Các cạnh của đồ thị G đều có trọng số. Cây T có tập hợp các nút là V được gọi là cây phủ (spaning tree) của đồ thị G.

Cây phủ tối thiểu là một cây phủ của G mà tổng độ dài (trọng số) các cạnh nhỏ nhất.

Bài toán cây phủ tối thiểu thường được áp dụng trong việc thiết kế một mạng lưới giao thông giữa các thành phố hay thiết kế một mạng máy tính.

Kĩ thuật tìm kiếm địa phương áp dụng vào bài toán này như sau:

- Phương án ban đầu là một cây phủ nào đó.
- Thành lập tập tất cả các cạnh theo thứ tăng dần của độ dài (có $\frac{n(n-1)}{2}$ cạnh đối với đồ thị có n đỉnh).
- Phép biến đổi địa phương ở đây là: Chọn một cạnh có độ dài nhỏ nhất trong tập các cạnh chưa sử dụng để thêm vào cây. Trong cây sẽ có một chu trình, loại khỏi chu trình cạnh có độ dài lớn nhất trong chu trình đó. Ta được một cây phủ mới. Lặp lại bước này cho đến khi không còn cải thiện được phương án nữa.

Ví dụ 3-12: Cho đồ thị G bao gồm 5 đỉnh a, b, c, d,e và độ dài các cạnh được cho trong hình 3-15.

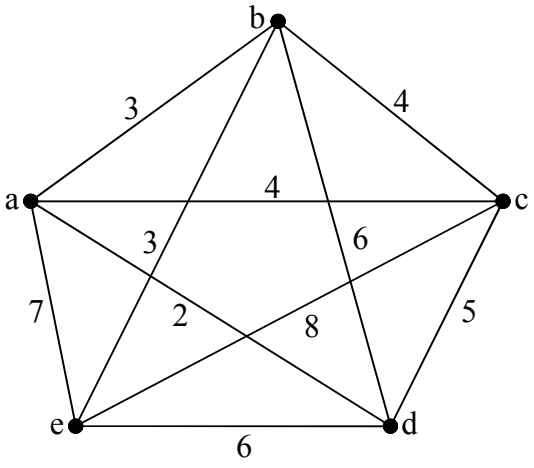
Tập hợp các cạnh để xét được thành lập theo thứ tự từ nhỏ đến lớn là ad, ab, be, bc, ac, cd, bd, de, ae và ce.

Cây xuất phát với giá là 20 (Hình 3-16). Thêm cạnh ad = 2, bỏ cạnh cd = 5 ta được cây mới có giá là 17 (Hình 3-17).

Lại thêm cạnh ab = 3, bỏ cạnh bc = 4 ta được cây có giá là16 (Hình 3-18).

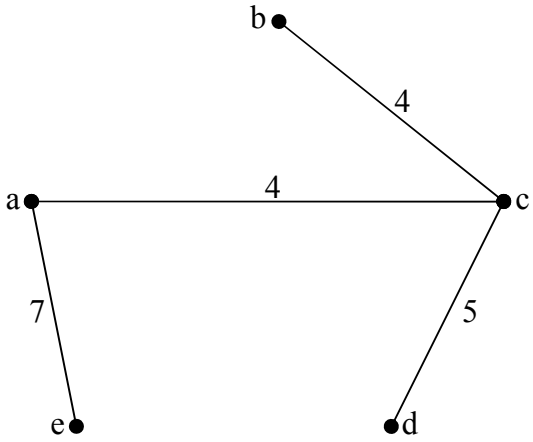
Thêm cạnh be = 3, bỏ cạnh ae = 7 ta được cây có giá là 12. (Hình 3-19).

Việc áp dụng các phép biến đổi đến đây dừng lại vì nếu tiếp tục nữa thì cũng không cải thiện được phương án.

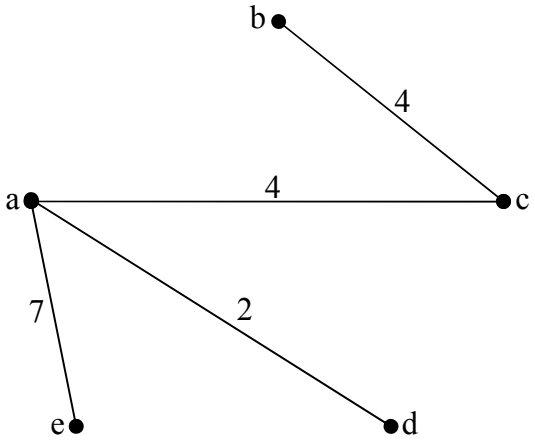


Hình 3-15: Bài toán cây phủ tối thiểu

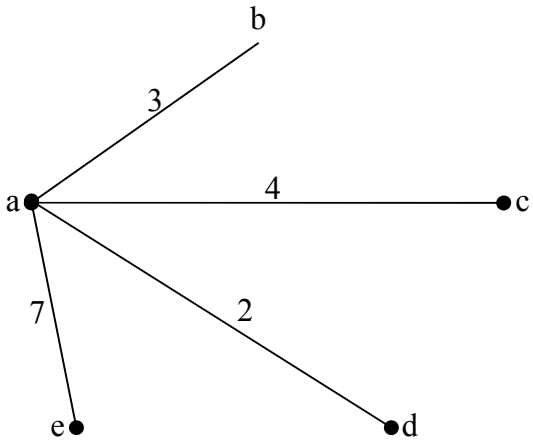
Vậy cây phủ tối thiểu cần tìm là cây trong hình 3-19



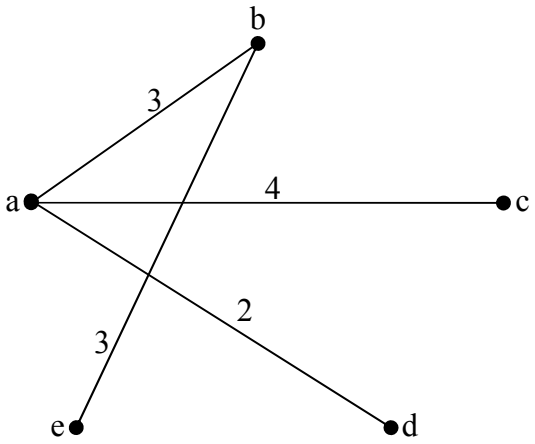
Hình 3-16: Cây xuất phát, giá 20



Hình 3-17: Giá 17



Hình 3-18: Giá 16



Hình 3-19: Giá 12

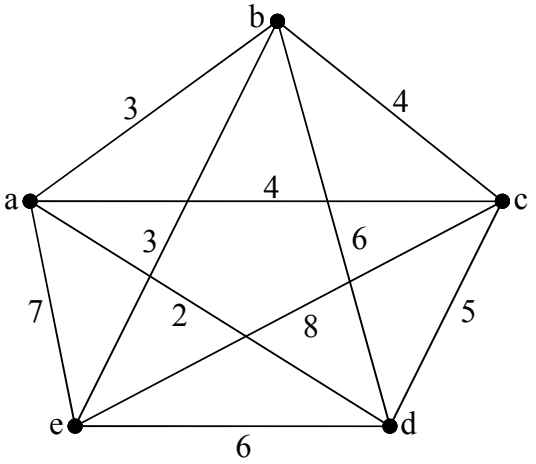
3.6.3 Bài toán đường đi của người giao hàng.

Ta có thể vận dụng kĩ thuật tìm kiếm địa phương để giải bài toán tìm đường đi ngắn nhất của người giao hàng (TSP).

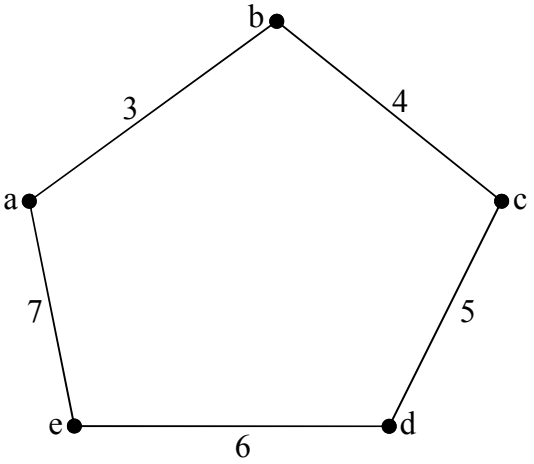
- Xuất phát từ một chu trình nào đó.
- Bỏ đi hai cạnh có độ dài lớn nhất không kề nhau, nối các đỉnh lại với nhau sao cho vẫn tạo ra một chu trình đủ.
- Tiếp tục quá trình biến đổi trên cho đến khi nào không còn cải thiện được phương án nữa.

Ví dụ 3-13: Bài toán TSP có 5 đỉnh và các cạnh có độ dài được cho trong hình 3-20

Phương án ban đầu là chu trình (a b c d e a) có giá (tổng độ dài) là 25. (Hình 3-21).

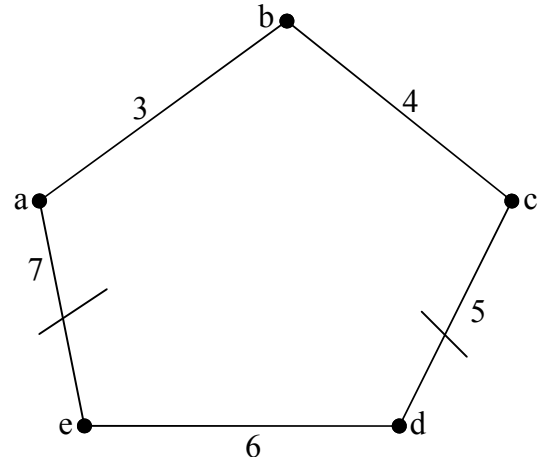


Hình 3-20: Bài toán TSP với 5 đỉnh

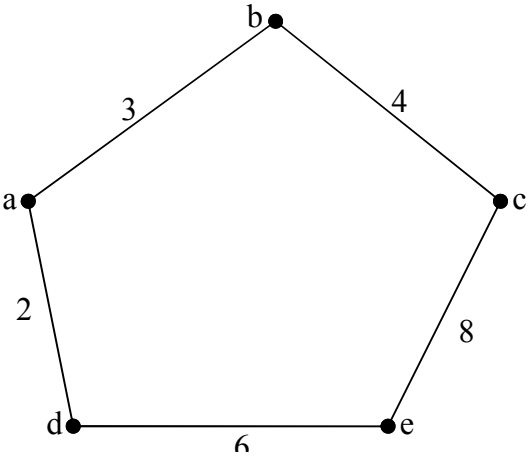


Hình 3-21: Phương án ban đầu, giá 25

Bỏ hai cạnh có độ dài lớn nhất không kề nhau là ae và cd (hình 3-22a), nối a với d và e với c. ta được chu trình mới (a b c e d a) với giá = 23 (Hình 3-22b).

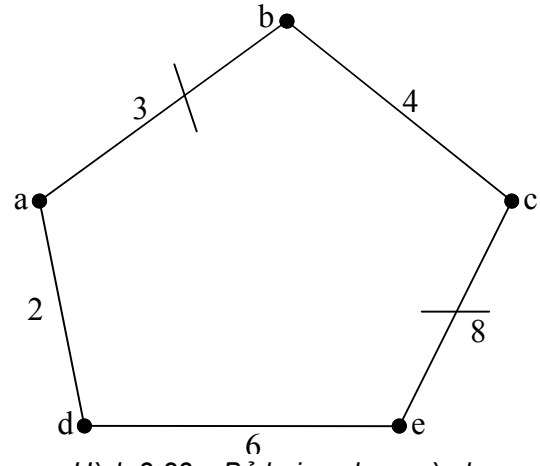


Hình 3-22a: Bỏ hai cạnh ae và cd

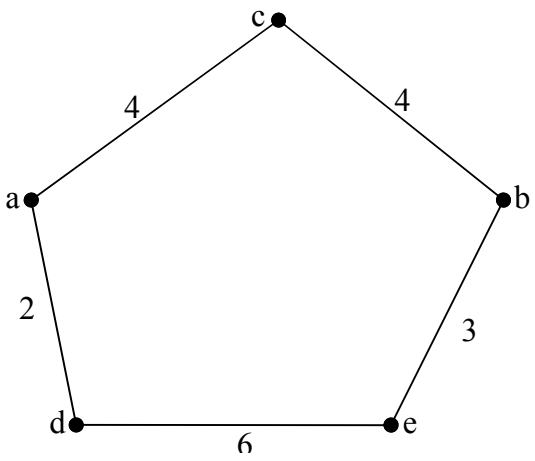


Hình 3-22b: Phương án mới, giá 23.

Bỏ hai cạnh có độ dài lớn nhất, không kề nhau là ce và ab (hình 3-23a), nối a với c và b với e, ta được chu trình mới (a c b e d a) có giá = 19. (Hình 3-23b). Quá trình kết thúc vì nếu tiếp tục thì giá sẽ tăng lên.



Hình 3-23a: Bỏ hai cạnh ce và ab.



Hình 3-23b: Phương án mới, giá 19

3.7 TỔNG KẾT CHƯƠNG 3

Trong các kĩ thuật được trình bày trong chương, kĩ thuật chia để trị là kĩ thuật cơ bản nhất. Hãy chia nhỏ các bài toán để giải quyết nó!

Với các bài toán tìm phương án tối ưu, kĩ thuật "tham ăn" giúp chúng ta nhanh chóng xây dựng được một phương án, dẫu rằng chưa hẳn tối ưu nhưng chấp nhận được. Kĩ thuật nhánh cận cho phép chúng ta tìm được phương án tối ưu. Trong kĩ thuật nhánh cận, việc phân nhánh không khó nhưng việc xác định giá trị cận là điều quan trọng. Cần phải xác định giá trị cận sao cho càng sát với giá của phương án càng tốt vì như thế thì có thể cắt tỉa được nhiều nút trên cây và đo đó sẽ giảm được thời gian thực hiện chương trình.

Vận dụng phương pháp quy hoạch động có thể giải được rất nhiều bài toán. Điều quan trọng nhất để áp dụng phương pháp quy hoạch động là phải xây dựng được công thức đệ quy để xác định kết quả bài toán thông qua kết quả các bài toán con.

BÀI TẬP CHƯƠNG 3

Bài 1: Giả sử có hai đội A và B tham gia một trận thi đấu thể thao, đội nào thắng trước n hiệp thì sẽ thắng cuộc. Chẳng hạn một trận thi đấu bóng chuyền 5 hiệp, đội nào thắng trước 3 hiệp thì sẽ tháng cuộc. Giả sử hai đội ngang tài ngang sức. Đội A cần thắng thêm i hiệp để thắng cuộc còn đội B thì cần thắng thêm j hiệp nữa. Gọi P(i,j) là xác suất để đội A cần i hiệp nữa để chiến thắng, B cần j hiệp. Dĩ nhiên i,j đều là các số nguyên không âm.

Để tính P(i,j) ta thấy rằng nếu i=0, tức là đội A đã thắng nên P(0,j) = 1. Tương tự nếu j=0, tức là đội B đã thắng nên P(i,0) = 0. Nếu i và j đều lớn hơn không thì ít nhất còn một hiệp nữa phải đấu và hai đội có khả năng 5 ăn, 5 thua trong hiệp này. Như vậy P(i,j) là trung bình cộng của P(i-1,j) và P(i,j-1). Trong đó P(i-1,j) là xác suất để đội A thắng cuộc nếu nó thắng hiệp đó và P(i,j-1) là xác suất để A thắng cuộc nếu nó thua hiệp đó. Tóm lại ta có công thức tính P(i,j) như sau:

P(i,j) = 1 Nếu i = 0
P(i,j) = 0 Nếu j = 0
P(i,j) = (P(i-1,j) + P(i,j-1))/2 Nếu i > 0 và j > 0

1. Viết một hàm đệ quy để tính P(i,j). Tính độ phức tạp của hàm đó.

2. Dùng kĩ thuật quy hoạch động để viết hàm tính P(i,j). Tính độ phức tạp của hàm đó.

3. Viết hàm P(i,j) bằng kĩ thuật quy hoach động nhưng chỉ dùng mảng một chiều (để tiết kiệm bộ nhớ).

Bài 2: Bài toán phân công lao động: Có n công nhân có thể làm n công việc. Công nhân i làm công việc j trong một khoảng thời gian tij. Phải tìm một phương án phân công như thế nào để các công việc đều được hoàn thành, các công nhân đều có việc làm, mỗi công nhân chỉ làm một công việc và mỗi công việc chỉ do một công nhân thực hiện đồng thời tổng thời gian là nhỏ nhất.

1. Mô tả kĩ thuật "tham ăn" (greedy) cho bài toán phân công lao động.
2. Tìm phương án theo giải thuật "háu ăn" cho bài toán phân công lao động được cho trong bảng sau. Trong đó mỗi dòng là một công nhân, mỗi cột là một công

việc, ô (i,j) ghi thời gian tij mà công nhân i cần để hoàn thành công việc j. (Cần chỉ rõ công nhân nào làm công việc gì và tổng thời gian là bao nhiêu)

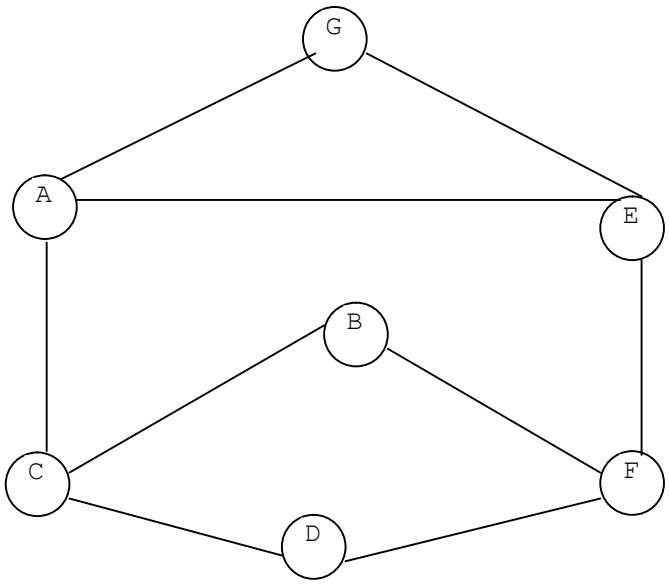
Công việc / Công nhân	1	2	3	4	5
1	5	6	4	7	2
2	5	2	4	5	1
3	4	5	4	6	3
4	5	5	3	4	2
5	3	3	5	2	5

Bài 3: Bài toán tô màu bản đồ thế giới

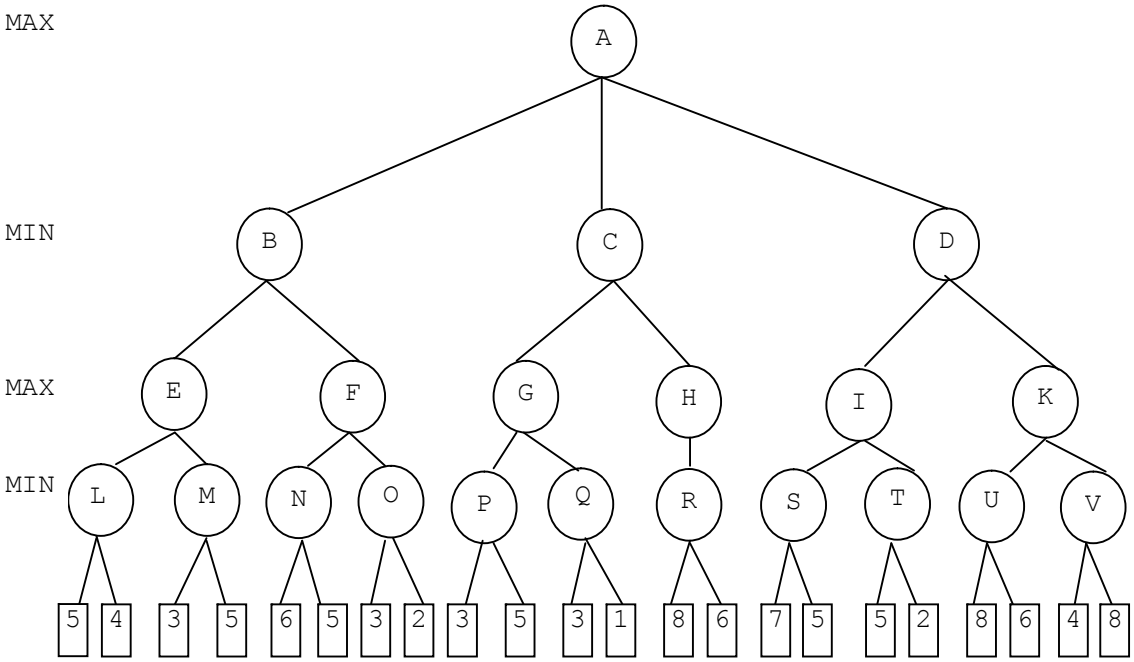
Người ta muốn tô màu bản đồ các nước trên thế giới, mỗi nước đều được tô màu và hai nước có biên giới chung nhau thì không được có màu giống nhau (các nước không chung biên giới có thể được tô màu giông nhau). Tìm một phương án tô màu sao cho số loại màu phải dùng ít nhất.

Người ta có thể mô hình hóa bản đồ thế giới bằng một đồ thị không có hướng, trong đó mỗi đỉnh biểu diễn cho một nước, biên giới của hai nước được biểu diễn bằng cạnh nối hai đỉnh. Bài toán tô màu bản đồ thế giới trở thành bài toán tô màu các đỉnh của đồ thi: Mỗi đỉnh của đồ thị phải được tô màu và hai đỉnh có chung một cạnh thì không được tô cùng một màu (cá đỉnh không chung cạnh có thể được tô cùng một màu). Tìm một phương án tô màu sao cho số loại màu phải dùng là ít nhất.

1. Hãy mô tả kĩ thuật "tham ăn" (Greedy) để giải bài toán tô màu cho đồ thị.
2. Áp dụng kĩ thuật háu ăn để tô màu cho các đỉnh của đồ thị sau (các màu có thể sử dung để tô là: ĐỎ, CAM, VÀNG, XANH, ĐEN, NÂU, TÍM)



Bài 4: 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 5: Xét một trò chơi có 6 viên bi, hai người thay phiên nhau nhặt từ 1 đến 3 viên. Người phải nhặt chỉ một viên bi cuối cùng thì bị thua.

1. Vẽ toán bộ cây trò chơi
2. Sử dụng kĩ thuật cắt tỉa alpha-beta định trị cho nút gốc
3. Ai sẽ thắng trong trò chơi này nếu hai người đều đi những nước tốt nhất. Hãy cho một nhận xét về trường hợp tổng quát khi ban đầu có n viên bi và mỗi lần có thể nhặt từ 1 đến m viên.

Bài 6: Xét một trò chơi có 7 cái đĩa. Người chơi 1 chia thành 2 chồng có số đĩa không bằng nhau. Người chơi 2 chọn một chồng trong số các chồng có thể chia và tiếp tục chia thành hai chồng không bằng nhau. Hai người luân phiên nhau chia đĩa như vậy cho đến khi không thể chia được nữa thì thua.

1. Vẽ toàn bộ cây trò chơi.
2. Sử dụng kĩ thuật cắt tỉa alpha-beta định trị cho nút gốc
3. Ai sẽ thắng trong trò chơi này nếu hai người đều đi những nước tốt nhất.

Bài 7: Cho bài toán cái ba lô với trọng lượng của ba lô W = 30 và 5 loại đồ vật được cho trong bảng bên. Tất cả các loại đồ vật đều **chỉ có một cái**.

1. Giải bài toán bằng kỹ thuật "Tham ăn" (Greedy).
2. Giải bài toán bằng kỹ thuật nhánh cận.

Loại đồ vật	Trọng lượng	Giá trị
A	15	30
B	10	25
C	2	2
D	4	6
E	8	24