Từ các cá thể được chọn để lai ghép, người ta cặp đôi chúng một cách ngẫu nhiên. Trong trường hợp các nhiễm sắc thể là các chuỗi nhị phân có độ dài cố định m, ta có thể thực hiện lai ghép như sau: Với mỗi cặp, sinh ra một số nguyên ngẫu nhiên p trên đoạn $[0, m-1]$, p là vị trí điểm ghép. Cặp gồm hai nhiễm sắc thể

$$a = (a_1, ..., a_p, a_{p+1}, ..., a_m)$$
$$a = (b_1, ..., b_p, b_{p+1}, ..., b_m)$$

được thay bởi hai con là:

$$a' = (a_1, ..., a_p, b_{p+1}, ..., b_m)$$
$$b' = (b_1, ..., b_p, a_{p+1}, ..., a_m)$$

3. Đột biến: Ta thực hiện toán tử đột biến trên các cá thể có được sau quá trình lai ghép. Đột biến là thay đổi trạng thái một số gien nào đó trong nhiễm sắc thể. Mỗi gien chịu đột biến với xác suất p_m. Xác suất đột biến p_m do ta xác định và là xác suất thấp. Sau đây là toán tử đột biến trên các nhiễm sắc thể chuỗi nhị phân.

Với mỗi vị trí i trong nhiễm sắc thể:

$$a = (a_1, ..., a_i, ..., a_m)$$

ta sinh ra một số thực ngẫu nhiên p_i trong $[0,1]$. Qua đột biến a được biến thành a' như sau:

$$a' = (a'_1, ..., a'_i, ..., a'_m)$$

trong đó:

$$a'_i = \begin{cases} a_i & \text{nếu } p_i \geq p_m \\ 1 - a_i & \text{nếu } p_i < p_m \end{cases}$$

Sau quá trình chọn lọc, lai ghép, đột biến, một thế hệ mới được sinh ra. Công việc còn lại của thuật toán di truyền bây giờ chỉ là lặp lại các bước trên.

Ví dụ. Xét bài toán tìm max của hàm $f(x) = x^2$ với x là số nguyên trên đoạn $[0,31]$. Để sử dụng *TTDT*, ta mã hoá mỗi số nguyên x trong đoạn $[0,31]$ bởi một số nhị phân độ dài 5, chẳng hạn, chuỗi 11000 là mã của số nguyên 24. Hàm thích nghi được xác định là chính hàm $f(x) = x^2$. Quần thể ban đầu gồm 4 cá thể (cỡ của quần thể là $n = 4$). Thực hiện quá trình chọn lọc, ta nhận được kết quả trong bảng sau. Trong bảng

này, ta thấy cá thể 2 có độ thích nghi cao nhất (576) nên nó được chọn 2 lần, cá thể 3 có độ thích nghi thấp nhất (64) không được chọn lần nào. Mỗi cá thể 1 và 4 được chọn 1 lần.

Bảng kết quả chọn lọc

Số hiệu cá thể	*Quần thể ban đầu*	*x*	*Độ thích nghi* $f(x) = x^2$	*Số lần được chọn*
1	01101	13	169	1
2	11000	24	576	2
3	01000	8	64	0
4	10011	19	361	1

Thực hiện qúa trình lai ghép với xác suất lai ghép $p_c = 1$, cả 4 cá thể sau chọn lọc đều được lai ghép. Kết quả lai ghép được cho trong bảng sau. Trong bảng này, chuỗi thứ nhất được lai ghép với chuỗi thứ hai với điểm ghép là 4, hai chuỗi còn lại được lai ghép với nhau với điểm ghép là 2.

Bảng kết quả lai ghép

Quần thể sau chọn lọc	*Điểm ghép*	*Quần thể sau lai ghép*	*x*	*Độ thích nghi* $f(x) = x^2$
0110|1	4	01100	2	144
1100|0	4	11001	5	625
11|000	2	11011	7	729
10|011	2	10000	6	256

Để thực hiện quá trình đột biến, ta chọn xác suất đột biến p_m= 0.001, tức là ta hy vọng có 5.4.0,001 = 0,02 bit được đột biến. Thực tế sẽ không có bit nào được đột biến. Như vậy thế hệ mới là quần thể sau lai ghép. Trong thế hệ ban đầu, độ thích nghi cao nhất là 576, độ thích nghi trung bình 292. Trong thế hệ sau, độ thích nghi cao nhất là 729, trung bình là 438. Chỉ qua một thế hệ, các cá thể đã "tốt lên" rất nhiều.

Thuật toán di truyền khác với các thuật toán tối ưu khác ở các điểm sau:

- *TTDT* chỉ sử dụng hàm thích nghi để hướng dẫn sự tìm kiếm, hàm thích nghi chỉ cần là hàm thực dương. Ngoài ra, nó không đòi hỏi không gian tìm kiếm phải có cấu trúc nào cả.
- *TTDT* làm việc trên các nhiễm sắc thể là mã của các cá thể cần tìm.
- *TTDT* tìm kiếm từ một quần thể gồm nhiều cá thể.
- Các toán tử trong *TTDT* đều mang tính ngẫu nhiên.

Để giải quyết một vấn đề bằng *TTDT*, chúng ta cần thực hiện các bước sau đây:

- Trước hết ta cần mã hóa các đối tượng cần tìm bởi một cấu trúc dữ liệu nào đó. Chẳng hạn, trong các *TTDT* cổ điển, như trong ví dụ trên, ta sử dụng mã nhị phân.
- Thiết kế hàm thích nghi. Trong các bài toán tối ưu, hàm thích nghi được xác định dựa vào hàm mục tiêu.
- Trên cơ sở cấu trúc của nhiễm sắc thể, thiết kế các toán tử di truyền (lai ghép, đột biến) cho phù hợp với các vấn đề cần giải quyết.
- Xác định cỡ của quần thể và khởi tạo quần thể ban đầu.
- Xác định xác suất lai ghép p_c và xác suất đột biến p_m. Xác suất đột biến cần là xác suất thấp. Người ta (Goldberg, 1989) khuyên rằng nên chọn xác suất lai ghép là 0,6 và xác suất đột biến là 0,03. Tuy nhiên cần qua thử nghiệm để tìm ra các xác suất thích hợp cho vấn đề cần giải quyết.

Nói chung thuật ngữ *TTDT* là để chỉ *TTDT* cổ điển, khi mà cấu trúc của các nhiễm sắc thể là các chuỗi nhị phân với các toán tử di truyền đã được mô tả ở trên. Song trong nhiều vấn đề thực tế, thuận tiện hơn, ta có thể biểu diễn nhiễm sắc thể bởi các cấu trúc khác, chẳng hạn vectơ thực, mảng hai chiều, cây,... Tương ứng với cấu trúc của nhiễm sắc thể, có thể có nhiều cách xác định các toán tử di truyền. Quá trình sinh ra thế hệ mới $P(t)$ từ thế hệ cũ $P(t - 1)$ cũng có nhiều cách chọn lựa. Người ta gọi chung các thuật toán này là thuật toán tiến hóa (evolutionary algorithms) hoặc chương trình tiến hóa (evolution program).

Ví dụ. Bài toán người bán hàng (salesperson problem) là một bài toán kinh điển. Một người bán hàng cần thăm tất cả các thành phố trong một vùng lãnh thổ, mỗi thành phố đúng một lần rồi lại trở về điểm xuất phát. Được cho trước độ dài đường đi nối tất cả các thành phố. Người bán hàng cần tìm một lộ trình sao cho độ dài của lộ trình này là ngắn nhất. Không gian tìm kiếm cho bài toán người bán hàng là tập hợp tất cả các lộ trình có thể có. Nếu có n thành phố cần đi qua (trừ thành phố xuất phát), thì số lộ trình là $n!$. Người ta đã chứng minh được rằng, bài toán người bán hàng là bài toán NP – khó. Vì vậy, nếu số thành phố n lớn thì các thuật toán cho lời giải tối ưu đòi hỏi thời gian cực kỳ lớn. Trong các năm gần đây, nhiều tác giả đã đề xuất các thuật toán tiến hoá cho bài toán người bán hàng. Sau đây chúng ta trình bày một giải pháp. Trước hết, chúng ta giả thiết rằng, n thành phố mà người bán hàng cần đi qua được đánh số từ 1, 2, ..., n (thành phố xuất phát được đánh số là 0). Độ dài đường đi từ thành phố i tới thành phố j là $cost(i,j)$.

Bây giờ chúng ta phải lựa chọn một cách biểu diễn một lộ trình. Có rất nhiều phương pháp (xem [16]), song tự nhiên nhất là, mỗi lộ trình được biểu diễn bởi một hoán vị của n số 1, 2, ..., n. Chẳng hạn, lộ trình

$$4 \rightarrow 1 \rightarrow 9 \rightarrow 5 \rightarrow 8 \rightarrow 6 \rightarrow 2 \rightarrow 3 \rightarrow 7$$

được biểu diễn bởi hoán vị

$$(4, 1, 9, 5, 8, 6, 2, 3, 7)$$

Như vậy, mỗi nhiễm sắc thể là một hoán vị của n số. Giả sử cá thể a được biểu diễn bởi hoán vị

$$a = (i_1, i_2, \ldots, i_n)$$

khi đó độ thích nghi của nó được xác định là

$$f(a) = M - (cost(0, i_1) + \sum_{k=1}^{n-1} cost(i_k, i_{k+1}) + cost(i_n, 0))$$

trong đó, M là số đủ lớn sao cho $f(a)$ là số thực dương.

Sau đây chúng ta xác định toán tử lai ghép và đột biến. Với cách biểu diễn mỗi cá thể bởi một hoán vị của n số, các nhà nghiên cứu đã đưa ra nhiều cách xác định toán tử lai ghép. Sau đây là toán tử lai ghép được đề xuất bởi Davis. Giả sử chúng ta có hai cá thể

p_1 = (5, 6, 2,| 9, 1, 8, 5,| 4, 3) và

p_2 = (1, 8, 4,| 2, 6, 5, 9,| 7, 3)

Hai điểm cắt được chọn ngẫu nhiên và được đánh dấu bởi dấu gạch đứng. Hai cá thể trên sinh ra hai con bằng cách sau đây. Các đoạn nằm giữa hai điểm cắt được sao chép vào hai cá thể con:

o_1 = (*x*, *x*, *x*,| 9, 1, 8, 7,| *x*, *x*) và

o_2 = (*x*, *x*, *x*,| 2, 6, 5, 9,| *x*, *x*)

Từ cá thể cha p_2, chúng ta thành lập dãy các thành phố bắt đầu từ điểm cắt thứ hai:

7, 3, 1, 8, 4, 2, 6, 5, 9

Loại bỏ các thành phố nằm giữa hai điểm cắt trong cá thể cha p_1, chúng ta có dãy:

3, 4, 2, 6, 5

Đặt dãy này vào các vị trí *x* trong cá thể con o_1 bắt đầu từ điểm cắt thứ hai, ta nhận được cá thể con:

o_1 = (2, 6, 5, 9, 1, 8, 7, 3, 4)

Một cách tương tự, ta nhận được cá thể con thứ hai:

o_2 = (1, 8, 7, 2, 6, 5, 9, 4, 3)

Chúng ta cũng có nhiều cách để xác định toán tử đột biến. Một cách đột biến đơn giản là, từ một cá thể chúng ta chọn hai gen một cách ngẫu nhiên và trao đổi vị trí của hai gen đó. Chẳng hạn, từ cá thể

α = (8, 1,| 2, 5, 9,| 6, 4, 3)

bằng đột biến sinh ra cá thể mới

α' = (8, 9,| 2, 5, 1,| 6, 4, 3)

Thuật toán tiến hóa đã được áp dụng thành công trong nhiều vấn đề tối ưu và học máy. Để hiểu biết sâu sắc hơn về thuật toán tiến hoá, bạn đọc có thể tìm đọc [3], [9] và [16], [9] và [16] được xem là các sách hay nhất viết về *TTDT*. [3] cho ta cái nhìn tổng quát về sự phát triển gần đây của *TTDT*.

CHƯƠNG 4
TÌM KIẾM CÓ ĐỐI THỦ

Nghiên cứu máy tính chơi cờ đã xuất hiện rất sớm. Không lâu sau khi máy tính lập trình được ra đời, vào năm 1950, Claude Shannon đã viết chương trình chơi cờ đầu tiên. Các nhà nghiên cứu trí tuệ nhân tạo đã quan tâm nghiên cứu máy tính chơi cờ, vì rằng khi đó người ta còn đang bàn cãi về khả năng làm cho máy tính trở thành thông minh, thì việc máy tính chơi được cờ là một bằng chứng rõ ràng về khả năng máy tính có thể làm được các công việc đòi hỏi trí thông minh của con người. Trong chương này chúng ta sẽ xét các vấn đề sau đây:

- Chơi cờ có thể xem như vấn đề tìm kiếm trong không gian trạng thái.
- Chiến lược tìm kiếm nước đi Minimax.
- Phương pháp cắt cụt α-β, một kỹ thuật để tăng hiệu quả của tìm kiếm Minimax.

4.1. CÂY TRÒ CHƠI VÀ TÌM KIẾM TRÊN CÂY TRÒ CHƠI

Trong chương này chúng ta chỉ quan tâm nghiên cứu các trò chơi có hai người tham gia, chẳng hạn các loại cờ (cờ vua, cờ tướng, cờ ca rô...). Một người chơi được gọi là Trắng, đối thủ của anh ta được gọi là Đen. Mục tiêu của chúng ta là nghiên cứu chiến lược chọn nước đi cho Trắng (Máy tính cầm quân Trắng).

Chúng ta sẽ xét các trò chơi hai người với các đặc điểm sau. Hai người chơi thay phiên nhau đưa ra các nước đi tuân theo các luật đi nào đó, các luật này là như nhau cho cả hai người. Điển hình là cờ vua, trong cờ vua hai người chơi có thể áp dụng các luật đi con tốt, con xe,... để đưa

ra nước đi. Luật đi con tốt Trắng xe Trắng,... cũng giống như luật đi con tốt Đen, xe Đen,... Một đặc điểm nữa là hai người chơi đều được biết thông tin đầy đủ về các tình thế trong trò chơi (không như trong chơi bài, người chơi không thể biết các người chơi khác còn những con bài gì). Vấn đề chơi cờ có thể xem như vấn đề tìm kiếm nước đi, tại mỗi lần đến lượt mình, người chơi phải tìm trong số rất nhiều nước đi hợp lệ (tuân theo đúng luật đi), một nước đi tốt nhất sao cho qua một dãy nước đi đã thực hiện, anh ta giành phần thắng. Tuy nhiên vấn đề tìm kiếm ở đây sẽ phức tạp hơn vấn đề tìm kiếm mà chúng ta đã xét trong các chương trước, bởi vì ở đây có đối thủ, người chơi không biết được đối thủ của mình sẽ đi nước nào trong tương lai. Sau đây chúng ta sẽ phát biểu chính xác hơn vấn đề tìm kiếm này.

Vấn đề chơi cờ có thể xem như vấn đề tìm kiếm trong không gian trạng thái. Mỗi trạng thái là một tình thế (sự bố trí các quân của hai bên trên bàn cờ).

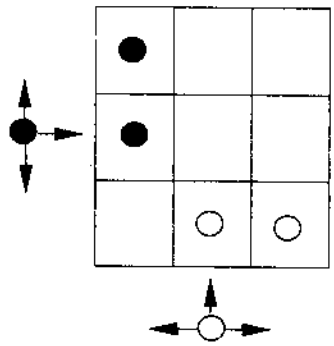
- Trạng thái ban đầu là sự sắp xếp các quân cờ của hai bên lúc bắt đầu cuộc chơi.
- Các toán tử là các nước đi hợp lệ.
- Các trạng thái kết thúc là các tình thế mà cuộc chơi dừng, thường được xác định bởi một số điều kiện dừng nào đó.
- Một hàm kết cuộc (payoff function) ứng mỗi trạng thái kết thúc với một giá trị nào đó. Chẳng hạn như cờ vua, mỗi trạng thái kết thúc chỉ có thể là thắng, hoặc thua (đối với Trắng) hoặc hòa. Do đó, ta có thể xác định hàm kết cuộc là hàm nhận giá trị 1 tại các trạng thái kết thúc là thắng (đối với Trắng), -1 tại các trạng thái kết thúc là thua (đối với Trắng) và 0 tại các trạng thái kết thúc hòa. Trong một số trò chơi khác, chẳng hạn trò chơi tính điểm, hàm kết cuộc có thể nhận giá trị nguyên trong khoảng $[-k, k]$ với k là một số nguyên dương nào đó.

Như vậy vấn đề của Trắng là, tìm một dãy nước đi sao cho xen kẽ với các nước đi của Đen tạo thành một đường đi từ trạng thái ban đầu tới trạng thái kết thúc là thắng cho Trắng.

Để thuận lợi cho việc nghiên cứu các chiến lược chọn nước đi, ta biểu diễn không gian trạng thái trên dưới dạng cây trò chơi.

CÂY TRÒ CHƠI

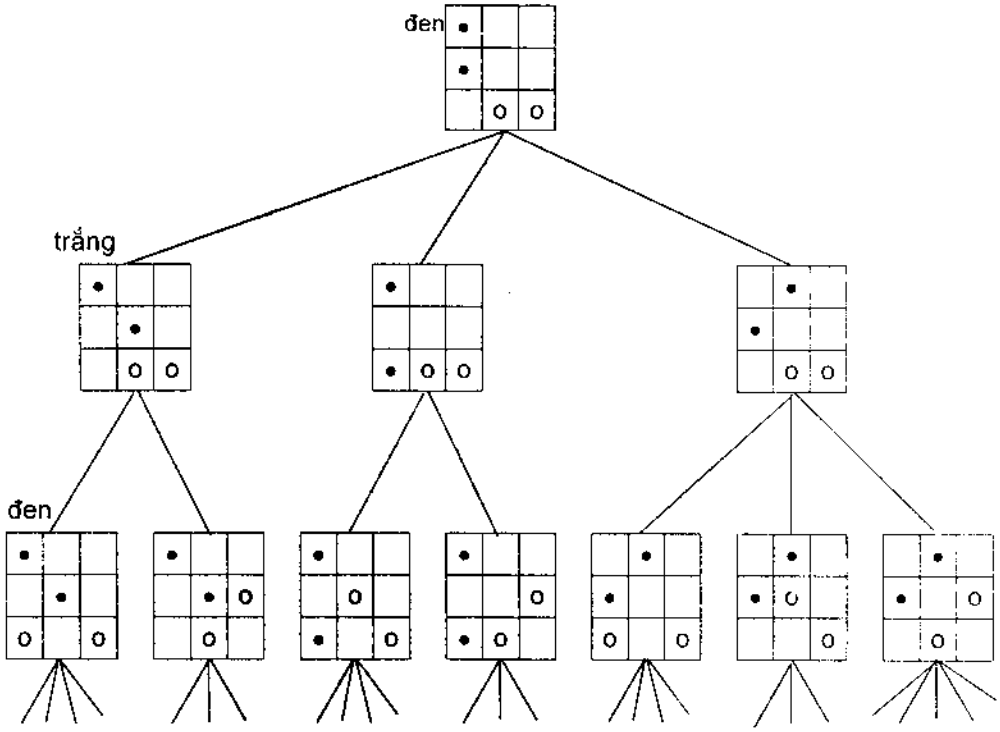
Cây trò chơi được xây dựng như sau. Gốc của cây ứng với trạng thái ban đầu. Ta sẽ gọi đỉnh ứng với trạng thái mà Trắng (Đen) đưa ra nước đi là đỉnh Trắng (Đen). Nếu một đỉnh là Trắng (Đen) ứng với trạng thái u, thì các đỉnh con của nó là tất cả các đỉnh biểu diễn trạng thái v, v nhận được từ u do Trắng (Đen) thực hiện nước đi hợp lệ nào đó. Do đó, trên cùng một mức của cây các đỉnh đều là Trắng hoặc đều là Đen, các lá của cây ứng với các trạng thái kết thúc.



Hình 4.1. **Trò chơi Dodgem.**

Ví dụ. Xét trò chơi Dodgem (được tạo ra bởi Colin Vout). Có hai quân Trắng và hai quân Đen, ban đầu được xếp vào bàn cờ 3*3 (Hình 4.1). Quân Đen có thể đi tới ô trống ở bên phải, ở trên hoặc ở dưới. Quân Trắng có thể đi tới ô trống ở bên trái, bên phải, ở trên. Quân Đen nếu ở cột ngoài cùng bên phải có thể đi ra khỏi bàn cờ, quân Trắng nếu ở hàng trên cùng có thể đi ra khỏi bàn cờ. Ai đưa hai quân của mình ra khỏi bàn cờ trước sẽ thắng, hoặc tạo ra tình thế mà đối phương không đi được cũng sẽ thắng.

Giả sử Đen đi trước, ta có cây trò chơi được biểu diễn như trong hình 4.2.



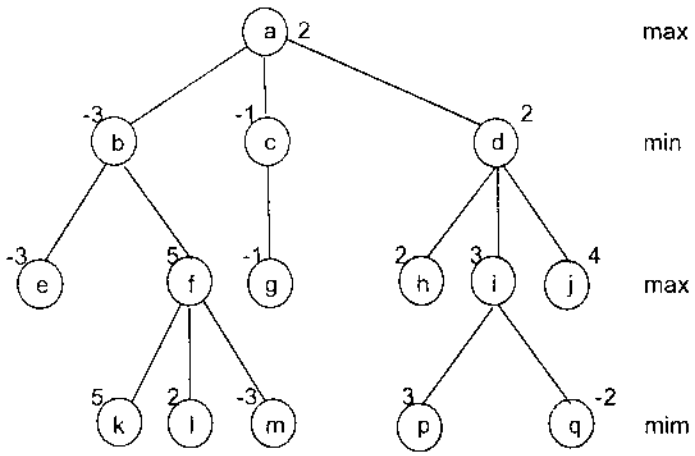
Hình 4.2. **Cây trò chơi Dodgem với Đen đi trước.**

4.2. CHIẾN LƯỢC MINIMAX

Quá trình chơi cờ là quá trình Trắng và Đen thay phiên nhau đưa ra quyết định, thực hiện một trong số các nước đi hợp lệ. Trên cây trò chơi, quá trình đó sẽ tạo ra đường đi từ gốc tới lá. Giả sử tới một thời điểm nào đó, đường đi đã dẫn tới đỉnh u. Nếu u là đỉnh Trắng (Đen) thì Trắng (Đen) cần chọn đi tới một trong các đỉnh Đen (Trắng) v là con của u. Tại đỉnh Đen (Trắng) v mà Trắng (Đen) vừa chọn, Đen (Trắng) sẽ phải chọn đi tới một trong các đỉnh Trắng (Đen) w là con của v. Quá trình trên sẽ dừng lại khi đạt tới một đỉnh là lá của cây.

Giả sử Trắng cần tìm nước đi tại đỉnh u. Nước đi tối ưu cho Trắng là nước đi dẫn tới đỉnh con v, v là đỉnh tốt nhất (cho Trắng) trong số các đỉnh con của u. Ta cần giả thiết rằng, đến lượt đối thủ chọn nước đi từ v, Đen cũng sẽ chọn nước đi tốt nhất cho anh ta. Như vậy, để chọn nước đi tối ưu cho Trắng tại đỉnh u, ta cần phải xác định giá trị các đỉnh của cây

trò chơi gốc u. Giá trị của các đỉnh lá (ứng với các trạng thái kết thúc) là giá trị của hàm kết cuộc. Đỉnh có giá trị càng lớn càng tốt cho Trắng, đỉnh có giá trị càng nhỏ càng tốt cho Đen. Để xác định giá trị các đỉnh của cây trò chơi gốc u, ta đi từ mức thấp nhất lên gốc u. Giả sử v là đỉnh trong của cây và giá trị các đỉnh con của nó đã được xác định. Khi đó nếu v là đỉnh Trắng thì giá trị của nó được xác định là giá trị lớn nhất trong các giá trị của các đỉnh con. Còn nếu v là đỉnh Đen thì giá trị của nó là giá trị nhỏ nhất trong các giá trị của các đỉnh con.



Hình 4.3. **Gán giá trị cho các đỉnh của cây trò chơi.**

Ví dụ. Xét cây trò chơi trong hình 4.3, gốc a là đỉnh Trắng. Giá trị của các đỉnh là số ghi cạnh mỗi đỉnh. Đỉnh i là Trắng, nên giá trị của nó là max(3,-2) = 3, đỉnh d là đỉnh Đen, nên giá trị của nó là min(2, 3, 4) = 2.

Việc gán giá trị cho các đỉnh được thực hiện bởi các hàm đệ qui MaxVal và MinVal. Hàm MaxVal xác định giá trị cho các đỉnh Trắng, hàm MinVal xác định giá trị cho các đỉnh Đen.

```
Function   MaxVal(u);
begin
   if  u là đỉnh kết thúc  then  MaxVal(u) ← f(u)
          else  MaxVal(u) ← max{MinVal(v) | v là đỉnh con của u}
end;
```