Chương 2

CÁC PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI TRONG KHÔNG GIAN TRẠNG THÁI

Quá trình tìm kiếm lời giải của bài toán được biểu diễn trong không gian trạng thái được xem như quá trình dò tìm trên đồ thị, xuất phát từ trạng thái ban đầu, thông qua các toán tử chuyển trạng thái, lần lượt đến các trạng thái tiếp theo cho đến khi gặp được trạng thái đích hoặc không còn trạng thái nào có thể tiếp tục được nữa. Khi áp dụng các phương pháp tìm kiếm trong không gian trạng thái , người ta thường quan tâm đến các vấn đề sau:

- Kỹ thuật tìm kiếm lời giải
- Phương pháp luận của việc tìm kiếm
- Cách thể hiên các nút trong quá trình tìm kiếm (mô tả trạng thái bài toán)
- Việc chọn các toán tử chuyển trạng thái nào để áp dung và có khả năng sử dụng .phương pháp may rủi trong quá trình tìm kiếm.

Tuy nhiên, không phải các phương pháp này có thể áp dụng cho tất cả các bài toán phức tạp mà cho từng lớp bài toán. Việc chọn chiến lược tìm kiếm cho bài toán cụ thể phụ thuộc nhiều vào các đặc trưng của bài toán.

Các thủ tục tìm kiếm điển hình bao gồm:

- Tìm kiếm theo chiều rộng (Breadth – First Search)
- Tìm kiếm theo chiều sâu (Depth – First Search)
- Tìm kiếm sâu dần (Depthwise Search)
- Tìm kiếm cực tiểu hoá giá thành (Cost minimization Search).
- Tìm kiếm với tri thức bổ sung (Heuristic Search).

1. Phương pháp tìm kiếm theo chiều rộng.

1.1. Kỹ thuật tìm kiếm rộng.

Kỹ thuật tìm kiếm rông là tìm kiếm trên tất cả các nút của một mức trong không gian bài toán trước khi chuyển sang các nút của mức tiếp theo.

Kỹ thuật tìm kiếm rộng bắt đầu từ mức thứ nhất của không gian bài toán, theo hướng dẫn của luật trọng tài, chẳng hạn "đi từ trái sang phải". Nếu không thấy lời giải tại mức này, nó chuyển xuống mức sau để tiếp tục … đến khi định vị được lời giải nếu có.

1.2. Giải thuật.

Input:

Cây/Đồ thị G = (V,E) với đỉnh gốc là n_0 (trạng thái đầu)

Tập đích Goals

Output:

Một đường đi p từ n_0 đến một đỉnh $n_* \in$ Goals

Method:

Sử dụng hai danh sách hoạt động theo nguyên tắc FIFO (queue) MO và DONG

Procedure BrFS; (Breadth First Search)

```
Begin
    Append(MO,no)
    DONG=null;
    While MO <> null do
            begin
                n:= Take(MO);
                if n∈ DICH then exit;
                Append(DONG, n);
                For m∈ T(n) and m∉ DONG+MO do
                    Append(MO, m);
            end;
```

Write (‘Không có lời giải’);

End;

Chú ý: Thủ tục Append(MO,n_0) bổ sung một phần tử vào queue MO.

Hàm Take(MO) lấy một phần tử trong queue MO.

- Nếu G là cây thì không cần dùng danh sách DONG.

1.3. Đánh giá độ phức tạp của giải thuật tìm kiếm rộng.

Giả sử rằng, mỗi trạng thái khi được xét sẽ sinh ra k trạng thái kế tiếp. Khi đó ta gọi k là nhân tố nhánh. Nếu bài toán tìm được nghiệm theo phương pháp tìm kiếm rộng có độ dài d. Như vậy, đỉnh đích sẽ nằm ở mức d+1, do đó số đỉnh cần xét lớn nhất là:

$1 + k + k^2 + . . . + k^d$.

Như vậy độ phức tạp thời gian của giải thuật là $O(k^d)$. Độ phức tạp không gian cũng là $O(k^d)$, vì tất cả các đỉnh của cây tìm kiếm ở mức d+1 đều phải lưu vào danh sách.

1.4. Ưu và nhược điểm của phương pháp tìm kiếm rộng.

1.4.1. Ưu điểm.

- Kỹ thuật tìm kiếm rộng là kỹ thuật vét cạn không gian trạng thái bài toán vì vậy sẽ tìm được lời giải nếu có.
- Đường đi tìm được đi qua ít đỉnh nhất.

1.4.2. Nhược điểm.

- Tìm kiếm lời giải theo thuật toán đã định trước, do vậy tìm kiếm một cách máy móc; khi không có thông tin hổ trợ cho quá trình tìm kiếm, không nhận ra ngay lời giải.
- Không phù hợp với không gian bài oán kích thước lớn. Đối với loại bài toán này, phương pháp tìm rộng đối mặt với các nhu cầu:
  - Cần nhiều bộ nhớ theo số nút cần lưu trữ.

- Cần nhiều công sức xử lý các nút, nhất là khi các nhánh cây dài, số nút tăng.
- Dễ thực hiện các thao tác không thích hợp, thừa, đưa đến việc tăng đáng kể số nút phải xử lý.

- Không hiệu qủa nếu lời giải ở sâu. Phương pháp này không phù hợp cho trường hợp có nhiều đường dẫn đến kết quả nhưng đều sâu.
- Giao tiếp với người dùng không thân thiện. Do duyệt qua tất cả các nút, việc tìm kiếm không tập trung vào một chủ đề.

1.5. Các ví dụ.

Ví dụ 1. Bài toán đong nước với m = 5, n= 4, k =3

Mức 1: Trạng thái đầu (0;0)

Mức 2: Các trạng thái (5;0), (0;4), Mức 3: (5;4), (1;4), (4,0)

Mức 4: (1;0), (4;4) Mức 5: (0;1), (5;3)

Ở mức 5 ta gặp trạng thái đích là (5;3) vì vậy có được lời giải như sau:

(0;0)→(0;4) →(4;0) →(4;4) →(5;3)

Để có được lời giải này ta phải lưu lại vết của đường đi, có thể trình bày quá trình tìm kiếm dưới dạng bảng sau:

i	T(i)	⇑MO ↓	DONG
		(0;0)	
(0;0)	(5;0) **(0;4)**	(5;0) (0;4)	(0;0)
(5;0)	(5;4) (0;0) (1;4)	(0;4) (5;4) (1;4)	(0;0) (5;0)
(0;4)	(5;4) (0;0) **(4;0)**	(5;4) (1;4) (4;0)	(0;0) (5;0) (0;4)
(5;4)	(0;4) (5;0)	(1;4) (4;0)	(0;0) (5;0) (0;4) (5;4)
(1;4)	(5;4) (0;4) (1;0) (5;0)	(4;0) (1;0)	(0;0) (5;0) (0;4) (5;4) (1;4)
(4;0)	(5;0) (4;4) (0;0) **(0;4)**	(1;0) (4;4)	(0;0) (5;0) (0;4) (5;4) (1;4) (4;0)
(1;0)	(5;0) (1;4) (0;1)	(4;4) (0;1)	(0;0) (5;0) (0;4) (5;4) (1;4) (4;0) (1;0)
(4;4)	(5;4) (0;4) (4;0)	(0;1) (5;3)	(0;0) (5;0) (0;4) (5;4) (1;4) (4;0)

	(5;3)		(1;0) (4;4)
(0;1)	(5;1) (0;4) (0;0) (1;0)	(5;3) (5;1)	(0;0) (5;0) (0;4) (5;4) (1;4) (4;0) (1;0) (0;1)
(5;3)			

Ví dụ 2. Bài toán trò chơi 8 số

Bảng xuất phát

2	8	3
1	6	4
7		5

Bảng kết thúc

1	2	3
8		4
7	6	5

Mức 1: Có một trạng thái

2	8	3
1	6	4
7		5

Mức 2: Có ba trạng thái

2	8	3
1		4
7	6	5

2	8	3
1	6	4
	7	5

2	8	3
1	6	4
7	5	

Mức 3: Có năm trạng thái

2	8	3
	1	4
7	6	5

2	8	3
1	4	
7	6	5

2		3
1	8	4
7	6	5

2	8	3
	6	4
1	7	5

2	8	3
1	6	
7	5	4

Mức 4: Có mười trạng thái

	8	3
2	1	4
7	6	5

2	8	3
7	1	4
	6	5

2	8	
1	4	3

2	8	3
1	4	5

1	7	5

7	6	

	2	3
1	8	4
7	6	5

2	3	
1	8	4
7	6	5

	8	3
2	6	4
1	7	5

2	8	3
6		4
1	7	5

2	8	
1	6	3
7	5	4

2	8	3
1	6	4
7	5	

Mức 6: Có 12 trạng thái

1	2	3
	8	4
7	6	5

2	3	4
1	8	
7	6	5

8		3
2	1	4
7	6	5

2	8	3
7	1	4
6		5

2		8
1	4	3
7	6	5

2	8	3
1	4	5
7		6

8		3
2	6	4
1	7	5

2		3
6	8	4
1	7	5

2	8	3
6	7	4
1		5

2		8
1	6	3
7	5	4

2		3
1	8	3
7	5	4

2	8	3
1	5	6
7		4

Mức 6: Có 24 trạng thái

1	2	3
8		4
7	6	5

1	2	3
7	8	4
	6	5

. . .

Ở mức này ta gặp được trạng thái đích.

1	2	3
8		4
7	6	5

2. Phương pháp tìm kiếm theo chiều sâu.

2.1. Kỹ thuật tìm kiếm sâu.

Tìm kiếm sâu trong không gian bài toán được bắt đầu từ một nút rồi tiếp tục cho đến khi hoặc đến ngõ cụt hoặc đến đích. Tại mỗi nút có luật trong tài, chẳng hạn, "đi theo nút cực trái", hướng dẫn việc tìm. Nếu không đi tiếp đượoc, gọi là đến ngõ cụt, hệ thống quay lại một mức trên đồ thị và tìm theo hướng khác, chẳng hạn, đến nút "sát nút cực trái". Hành động này gọi là quay lui.

Thuật toán tìm kiếm theo chiều sâu được hình dung như việc khảo sát một cây bắt đầu từ gốc đi theo mọi cành có thể được, khi gặp cành cụt thì quay lại xét cành chưa đi qua.

- Ở bước tổng quát, giả sử đang xét đỉnh i, khi đó các đỉnh kề với i có các trường hợp:

+ Nếu tồn tại đỉnh j kề i chưa được xét thì xét đỉnh này (nó trở thành đỉnh đã xét) và bắt đầu từ đó tiếp tục quá trình tìm kiếm với đỉnh này..

+ Nếu với mọi đỉnh kề với i đều đã được xét thì i coi như duyệt xong và quay trở lại tìm kiếm từ đỉnh mà từ đó ta đi đến được i.

2.2. Giải thuật.

Input:

Cây/Đồ thị G = (V,E) với đỉnh gốc là n_0 (trạng thái đầu)

Tập đích Goals

Output:

Một đường đi p từ n_0 đến một đỉnh $n_* \in$ Goals

Method:

Sử dụng hai danh sách hoạt động theo nguyên tắc LIFO (Stack) MO và DONG

Procedure DFS; (Depth First Search)

Begin

Push (MO,n_o)

```
DONG=null;
While  MO <> null  do
        begin
            n:=pop (MO);
            if n∈ DICH then exit;
            push (DONG, n);
            For  m∈ T(n) and m∉ DONG+MO do
                Push (MO, m);
        end;
Write ('Không có lời giải');
End;
```

Chú ý: Thủ tục Push(MO,n_0) thực hiện việc bổ sung n_0 vào stack MO

Hàm Pop(MO) lấy phần tử đầu tiên trong Stack MO.

2.3. Đánh giá độ phức tạp của thuật toán tìm kiếm sâu.

Gải sử nghiệm của bài toán là đường đi có độ dài d, cây tìm kiếm có nhân tố nhánh là k. Có thể xãy ra nghiệm là đỉnh cuối cùng được xét ở mức d+1 theo luật trọng tài. Khi đó độ phức tạp thời gian của thuật toán tìm kiếm theo chiều sâu trong trường hợp xấu nhất là $O(k^d)$.

Để đánh giá độ phức tạp không gian của thuật toán tìm kiếm sâu ta có nhận xét ràng: Khi xét đỉnh j, ta chỉ cần lưu các đỉnh chưa được xét mà chúng là những đỉnh con của những đỉnh nằm trên đường đi từ đỉnh gốc đến j. Vì vậy chỉ cần lưu tối đa la k*d. Do đó độ phức tạp không gian của thuật toán là O(k*d).

2.4. Ưu và nhược điểm của phương pháp tìm kiếm sâu.

2.4.1. Ưu điểm.

- Nếu bài toán có lời giải, phương pháp tìm kiếm sâu bảo đảm tìm ra lời giải.

- Kỹ thuật tìm kiếm sâu tập trung vào đích, con người cảm thấy hài lòng khi các câu hỏi tập trung vào vấn đề chính.
- Do cách tìm của kỹ thuật này, nếu lời giải ở rất sâu, kỹ thuật tìm sâu sẽ tiết kiệm thời gian.

2.4.2. Nhược điểm.

- Tìm sâu khai thác không gian bài toán để tìm lời giải theo thuật toán đơn giản một cách cứng nhắc. Trong quá trình tìm nó không có thông tin nào hổ trợ để phát hiện lời giải. Nếu chọn nút ban đầu không thích hợp có thể không dẫn đến đích của bài toán.
- Không phù hợp với không gian bài toán lớn, kỹ thuật tìm kiếm sâu có thể không đến lời giải trong khoảng thời gian vừa phải.

2.5. Các ví dụ.

Ví dụ 1. Bài toán đong nước với m = 5, n = 4, k = 3

Nếu ta chọn nhánh ưu tiên đổ đầy bình thứ hai thì sẽ tìm thấy lời giải rất nhanh. Quá trình tìm kiếm có thể trình bày bằng bảng dưới đây

i	T(i)	MO ↓↑	DONG
		(0;0)	
(0;0)	(5;0) **(0;4)**	(5;0) (0;4)	(0;0)
(0;4)	(5;4) (0;0) **(4;0)**	(5;0) (5;4) (4;0)	(0;0) (0;4)
(4;0)	(5;0) (4;4) (0;0) **(0;4)**	(5;0) (5;4) (4;4)	(0;0) (0;4) (4;0)
(4;4)	(5;4) (0;4) (4;0) **(5;3)**	(5;0) (5;4) (5;3)	(0;0) (0;4) (4;0) (4;4)
(5;3)			

Lời giải tìm được: (0;0) →(0;4) →(4;0) →(4;4) →(5;3)

Ví dụ 2. Bài toán Tháp Hà nội với n = 3.

Nhắc lại, dùng bộ ba (x_1; x_2; x_3) biểu diễn trạng thái bài toán, với x_i là cọc chứa đĩa lớn thứ i.

i	T(i)	MO ↓↑	DONG

		(1;1;1)	
(1;1;1)	(1;1;2) (1;1;3)	(1;1;2) (1;1;3)	(1;1;1)
(1;1;3)	(1;1;1)(1;1;2) (1;2;3)	(1;1;2)(1;2;3)	(1;1;1)(1;1;3)
(1;2;3)	(1;1;3) (1;2;1) (1;2;2)	(1;1;2)(1;2;1)(1;2;2)	(1;1;1)(1;1;3)(1;2;3)
(1;2;2)	(1;2;3) (1;2;1) (3;2;2)	(1;1;2)(1;2;1)(3;2;2)	(1;1;1)(1;1;3)(1;2;3)(1;2;2)
(3;2;2)	(1;2;2) (3;2;3) (3;2;1)	(1;1;2)(1;2;1)(3;2;1)	(1;1;1)(1;1;3)(1;2;3)(1;2;2) (3;2;2)
(3;2;1)	(3;2;2) (3;2;3) (3;3;1)	(1;1;2)(1;2;1)(3;3;1)	(1;1;1)(1;1;3)(1;2;3)(1;2;2) (3;2;2) (3;2;1)
(3;3;1)	(3;2;1) (3;3;2) (3;3;3)	(1;1;2)(1;2;1)(3;3;3)	(1;1;1)(1;1;3)(1;2;3)(1;2;2) (3;2;2) (3;2;1) (3;3;3)
(3;3;3)			

Lời giải của bài toán:

(1;1;1) →(1;1;3) →(1;2;3) →(1;2;2) →(3;2;2) →(3;2;1) →(3;3;1) →(3;3;3)

- Cả hai ví dụ trên, chúng ta đều thấy, tìm kiếm theo chiều sâu đều cho lời giải tốt và nhanh.

Ví dụ 3. Bài toán tìm dãy hợp lý với số hạng đầu $a_1 = 26$

Nhắc lại: Dãy $a_1, a_2, \ldots, a_n$ được gọi là hợp lý nếu thoả hai điều kiện:

- a_n là số nguyên tố
- $a_{k+1} = a_k+1$ hoặc $2*a_k$

Như vậy, khi biết a_k thì ta xác định được a_{k+1}. Vì vậy có thể mô tả trạng thái bài toán tương ứng với giá trj a_k tại thòi điểm đang xét. Ta có thể chỉ ra một cách tìm kiếm theo chiều sâu như sau

I	T(i)	MO ↓↑	DONG
		26	
26	27 52	27 52	26
52	53 104	27 53 104	26 52
104	105 208	27 53 105 208	26 52 104
208	209 416	27 53 105 209 416	26 52 104 208
. . .			

Với cách tìm kiếm theo theo thuật toán một cách máy móc như vậy thì rõ ràng không bao giờ đạt được đích. Trong khi chúng ta dễ dàng nhận được lời giải, chăng hạn:

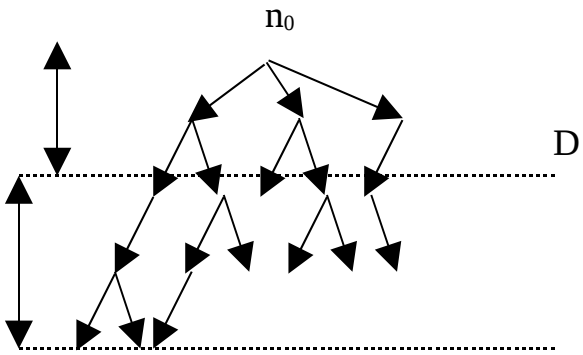
$a_1 = 26$; $a_2 = 52$; $a_3 = 53$. Như vậy n =3

3. Tìm kiếm sâu dần

3.1. Kỹ thuật tìm kiếm sâu dần.

Kỹ thuật tìm kiếm sâu dần là thực hiện việc tìm kiếm với độ sâu ở mức giưới hạn d nào đó. Nếu không tìm ra nghiệm ta tăng độ sâu lên d+1 và lại tìm kiếm theo độ sâu tới mức d+1. Quá trình trên được lặp lại với d lần lượt là 1, 2,...đến độ sâu max nào đó.

Kỹ thuật tìm kiếm sâu dần thường được thực hiện khi cây tìm kiếm chứa nhánh vô hạn, và nếu sử dụng tìm kiếm theo độ sâu ta có thể mắc kẹt ở một nhánh nào đó (thuật toán không dừng) và không tìm ra nghiệm.



3.2. Giải thuật.

Thuật toán tìm kiếm sâu dần sử dụng thuật toán tìm kiếm sâu hạn chế như thủ tục con. Đó là thủ tục tìm kiếm theo chiều sâu nhưng chỉ tới độ sâu d nào đó rồi quay lên.

Thủ tục tìm kiếm sâu hạn chế (depth_limitedsearch)

```
Procedure Depth_limited_search(d);          {d là tham số độ sâu}
    Begin
        Push (MO,n₀);
        Depth(n₀)=0;                        {hàm depth ghi lại độ sâu mỗi đỉnh}
        DONG=null;
        While  MO <> null  do
            begin
                n:=pop (MO);
                if n∈ DICH then exit;
                push (DONG, n);
                if depth(n)<=d then
                    For  m∈ T(n) and m∉ DONG do
                     begin
                        Push (MO, m);
                        depth(m)=depth(n)+1;
                     end;
            end;
        Write ('Không có lời giải');
    End
```

Thuật toán tìm kiếm sâu dần (Depth_deepening_search) sẽ sử dụng thủ tục tìm kiếm sâu hạn chế như thủ tục con:

Procedure Depth_deepening_search;

Begin

For d:=0 to max do

Depth_limited_search(d);

If thành công then exit;

End;

3.3. Nhận xét.

- Luôn tìm ra nghiệm (nếu bài toán có nghiệm), miễn là chọn max đủ lớn (giống như tìm kiếm theo chiều rộng)
- Có độ phức tạp thời gian là $O(k^d)$ (giống tìm kiếm rộng)
- Có độ phức tạp không gian là O(k*d) (giống tìm kiếm sâu)
- Giải thuật tìm kiếm sâu dần thương áp dụng cho các bài toán có không gian trạng thái lớn và độ sâu của nghiệm không biết trước.

4. Phương pháp tìm kiếm tốt nhất đầu tiên (Best First Search).

Cả hai kỹ thuật tìm kiếm rộng và tìm kiếm sâu đều là phương pháp cơ bản để khai thác không gian bài toán. Chúng đều vét cạn không gian để tìm ra lời giải theo thủ tục xác định trước. Mặc dù có sử dụng tri thức về trạng thái của bài toán để hướng dẫn tìm kiếm nhưng không phổ biến. Cho dù có một số ưu điểm, nhưng chúng là những kỹ thuật thực hiện một cách máy móc. Chính vì vậy chúng bị gán tên là *"kỹ thuật tìm kiếm mù"*.

4.1. Kỹ thuật tìm kiếm tốt nhất đầu tiên.

Kỹ thuật tìm kiếm tốt nhất đầu tiên tìm lời giải có dùng tri thức về bài toán để hướng dẫn. Tri thức này hướng việc tìm kiếm về nút lời giải trong không gian bài toán.

Tại mỗi nút được xem xét, người ta sẽ quyết định việc tìm kiếm tiếp tục theo nhánh nào tin tưởng sẽ dẫn đến lời giải.

Trong các chương trình trí tuệ nhân tạo, kỹ thuật tìm kiếm tốt nhất đầu tiên sử dụng hàm đánh giá. Hàm này dùng các thông tin hiện tại về mức độ quan trọng của bài toán tại nút đó để gán giá trị cho nút này, gọi là trọng số

của nút. Giá trị này được xem xét trong lúc tìm kiếm. Thông thường, nút có trọng số nhỏ (lớn) nhất sẽ được chọn trong quá trình tìm kiếm.

4.2. Hàm đánh giá

Trong nhiều vấn đề, ta có thể sử dụng kinh nghiệm, tri thức của chúng ta về vấn đề đó để đánh giá các trạng thái của vấn đề.

Với mỗi trạng thái u, ta sẽ xác dịnh một giá trị số h(u), số này đánh giá "sự gần đích" của trạng thái u. Hàm ***h(u) được gọi là hàm đánh giá***.

Phương pháp tìm kiếm kinh nghiệm là phương pháp tìm kiếm có sử dụng đến hàm đánh giá. Trong quá trình tìm kiếm, tại mỗi bước ta sẽ chọn trạng thái kế tiếp là trạng thái có nhiều hứa hẹn dẫn tới đích nhiều nhất.

Quá trình tìm kiếm trong không gian trạng thái có sử dụng hàm đánh giá bao gồm các bước cơ bản sau:

- Biểu diễn thích hợp các trạng thái và các toán tử chuyển trạng thái
- Xây dựng hàm đánh giá
- Thiết kế chiến lược chọn trạng thái ở mỗi bước

4.3. Ưu và nhược điểm của phương pháp tìm kiếm tốt nhất đầu tiên.

4.3.1. Ưu điểm.

- Phương pháp tìm kiếm tốt nhất đầu tiên tổ hợp các ưu điểm của phương pháp tìm kiếm rộng và tìm kiếm sâu.

- Ưu điểm chủ yếu của phương pháp tìm kiếm tốt nhất đầu tiên là dùng tri thức để dẫn dắt việc tìm kiếm. Tri thức này giúp người ta bắt đầu từ đâu là tốt nhất và cách tốt nhất để tiến hành tìm lời giải.

- Tìm kiếm tốt nhất đầu tiên tuân theo cách suy lý của một chuyên gia. Do đó có thể thấy rõ đường đi hơn tìm kiếm rộng và tìm kiếm sâu.

4.3.2. Nhược điểm.

- Quá trình tìm kiếm có thể đi xa khỏi lời giải. Kỹ thuật này chỉ xét một phần của không gian và coi đó là phần hứa hẹn hơn cả.

4.4. Giải thuật.

Dữ liệu tương tự như giải thuật tìm kiếm rông và sâu, sử dụng danh sách MO để lưu các đỉnh sẽ xét.

```
Procedure BFS; {Best First Search}
Begin
    Push(MO,n0);
    while  MO <> null do
    begin
        i := Pop(MO);
        if  i ∈ Goals then
            exit;
        for j ∈ T(i) do
            Push(MO,j);
        Sort(MO); {theo thứ tự của hàm đánh giá}
    end;
write('Khong co loi giai');
end;
```

4.5. Các ví dụ.

Ví dụ 1 Trong bài toán tìm kiếm đường đi trên bản đồ giao thông, ta có thể lấy độ dài của đường chim bay từ một thành phố đang xét tới một thành phố đích làm giá trị của hàm đánh giá của thành phố đang xét.

Ví dụ 2 Bài toán 8 số. Chúng ta có thể đưa ra hai cách đánh giá

u =

3	2	8
	6	4
7	1	5

đích =

1	2	3
8		4
7	6	5

- Hàm h_1: Với mỗi trạng thái u thì $h_1(u)$ là số quân không nằm đúng vị trí của nó trong trạng thái đích.

Với ví dụ trên, ta có **$h_1(u)=4$**

- Hàm h_2: Gọi $h_2(u)$ là là tổng khoảng cách giữa vị trí của các quân trong trạng thái u và vị trí của nó trong trạng thái đích. Ở đây, khoảng cách được hiểu là số lần dịch chuyển ít nhất theo hàng hoặc cột để đưa một quân ở vị trí của hiện tại tới trạng thái đích.

Với ví dụ trên, ta có:**$h_2(u)$=2+3+1+3= 9** (vì quân 3 cần ít nhất 2 dịch chuyển, quân 8 cần ít nhất 3 dịch chuyển, quân 6 cần ít nhất 1 dịch chuyển và quân 1 cần ít nhất 3 dịch chuyển)

5. Tìm kiếm đường đi có giá thành cực tiểu - Thuật toán AT

Cho đồ thị G= (V, E) biểu diễn bài toán với đỉnh xuất phát n_0 và tập đích DICH xác định.

Với mỗi phép chuyển trạng thái $n_i \rightarrow n_{i+1}$ tốn chi phí $c(n_i, n_{i+1})$ ký hiệu c(u) với $u= (n_i, ni_{+1}) \in E$

$$n_i \xrightarrow{c(u)} n_{i+1}$$

Vấn đề:

Tìm đường đi p: $\overrightarrow{n_0}$ $n^* \in$ DICH sao cho $c(p) = \sum_{u \in p} c(u) \rightarrow \min$

Chẳng hạn trong bài toán tìm đường đi trong bản đồ giao thông, giá của cung (i,j) chính là độ dài của đường nối thành phố i với thành phố j. Độ dài đường đi được xác định là tổng độ dài các cung trên đường đi. Vấn đề đặt ra là tìm đường đi ngắn nhất từ trạng thái ban đầu đến trạng thái đích.

- **Phương pháp giải**

1) Nếu $c(u) = k(const) \quad \forall u \in E$ thì $c(p) \rightarrow \min \Leftrightarrow \#p \rightarrow \min \Rightarrow$ Dùng phương pháp tìm kiếm theo chiều rộng.

2) Gọi g(n) là giá của đường đi cực tiểu từ đỉnh n_0 đến n, khi đó bài toán có thể phát biểu như sau:

Tìm đường đi từ đỉnh $n_0 \rightarrow n_k \in DICH$ **sao cho:** $g(n_k) = \min\{g(n) / n \in DICH\}$

Lúc đó, ta có: $g(n_0) = 0$

$$g(m) = \min_{(n,m) \in E} \{g(n) + c(n,m)\}$$

Dùng 2 danh sách MO, DONG như trên. Tại mỗi thời điểm chọn đỉnh n trong MO ra xét là đỉnh thoả.

- **Thuật toán AT**

Input:

Đồ thị G = (V,E), Đỉnh xuất phát n_0

Hàm chi phí c: $E \rightarrow R^+$

c(i,j): xác định chi phí chuyển từ đỉnh i sang đỉnh j với $(i,j) \in E$

Tập các đỉnh đích DICH

Output:

Đường đi từ đỉnh n_0 đến đỉnh $n_* \in$ DICH sao cho $g(n_*) = c(p) = \min\{g(n) | n \in DICH\}$.

Procedure AT;

{ Dùng $g^0(n)$ là chi phí cực tiểu của đường đi từ đỉnh xuất phát đến đỉnh n tại thời điểm đang xét và xem như hàm g}

```
Begin
    g(n0):= 0;
    push(MO, n0);
    While MO<>null do
        begin
            g(n) := min_{m∈MO} g(m)
            if n∈DICH then
                exit {xay dung duong di cuc tieu}
            push(DONG, n);
            if T(n) <>null then
                for  m∈T(n) do
                    if m∉MO+DONG then
                        begin
                            push(MO,m);
```

```
                    g(m):=g(n)+c(n,m);
                    cha(m):=n;
                end
            else
                if g(m) >g(n)+c(n,m) then
                    begin
                        g(m):=g(n)+c(n,m);
                        cha(m):=n;
                    end;
        end;
        writeln('Khong co duong di');
End;
```

Ví dụ 1. Bài toán Tháp Hà Nội -với chi phí chuyển đĩa như sau:

Chi phí chuyển đĩa nhỏ giữa 2 cọc gần	1
Chi phí chuyển đĩa nhỏ giữa 2 cọc xa	3
Chi phí chuyển đĩa vừa giữa 2 cọc gần	2
Chi phí chuyển đĩa vừa giữa 2 cọc xa	5
Chi phí chuyển đĩa lớn giữa 2 cọc gần	4
Chi phí chuyển đĩa lớn giữa 2 cọc xa	8

Xuất phát từ đỉnh (1,1,1), ta có g(1,1,1) = 0.

Khi xét đỉnh **(1,1,1)** ta có các đỉnh kề và chi phí tương ứng :

g(1,1,2) = 1; g(1,1,3) = 3; như vậy đỉnh **(1,1,2)** được chọn

Các đỉnh kề của (1,1,2) có giá trị hàm g:

g(1,1,3) = 2 (ở đây giá của đỉnh (1,1,3) được tính lại); g(1,3,2) = 5; chọn đỉnh **(1,1,3)**, ta lại tính tiếp giá trị hàm g của các đỉnh kề với đỉnh này:

g(1,2,3) = 2; lại chọn đỉnh **(1,2,3)**; chi phí của các đỉnh kề với nó:

g(1,2,1) = 2 + 3 = 5; g(1,2,2) = 2 + 1 = 3; chọn đỉnh **(1,2,2)**

g(1,2,1) = 3 +1 = 4 (được tính lại); g(3,2,2) = 3 + 8 = 11, chọn đỉnh (1,2,1)