

**BỘ GIAO THÔNG VẬN TẢI
TRƯỜNG ĐẠI HỌC HÀNG HẢI
BỘ MÔN: KHOA HỌC MÁY TÍNH
KHOA: CÔNG NGHỆ THÔNG TIN**

BÀI GIẢNG
TRÍ TUỆ NHÂN TẠO

TÊN HỌC PHẦN : TRÍ TUỆ NHÂN TẠO
MÃ HỌC PHẦN : 17210
TRÌNH ĐỘ ĐÀO TẠO : ĐẠI HỌC CHÍNH QUY
DÙNG CHO SV NGÀNH : CÔNG NGHỆ THÔNG TIN

HẢI PHÒNG - 2009

MỤC LỤC

STT	NỘI DUNG	TRANG
Phần I	Giải quyết vấn đề bằng tìm kiếm	1
Chương 1	Các chiến lược tìm kiếm mù	2
1.1	Biểu diễn vấn đề trong không gian trạng thái	2
1.2	Các chiến lược tìm kiếm	4
1.3	Các chiến lược tìm kiếm mù	6
1.4	Quy vấn đề về các vấn đề con. Tìm kiếm trên đồ thị và/hoặc	9
	Bài tập chương I	14
Chương 2	Các chiến lược tìm kiếm kinh nghiệm	15
2.1	Hàm đánh giá và tìm kiếm kinh nghiệm	15
2.2	Tìm kiếm tốt nhất - đầu tiên	16
2.3	Tìm kiếm leo đồi	17
2.4	Tìm kiếm beam	18
	Bài tập chương II	19
Chương 3	Các chiến lược tìm kiếm tối ưu	19
3.1	Tìm đường đi ngắn nhất	19
3.2	Tìm đối tượng tốt nhất	23
3.3	Tìm kiếm mô phỏng sự tiến hóa. Thuật toán di truyền	25
	Bài tập chương III	29
Chương 4	Tìm kiếm có đối thủ	30
4.1	Cây trò chơi và tìm kiếm trên cây trò chơi	30
4.2	Chiến lược Minimax	31
4.3	Phương pháp cắt cụt Alpha-Beta	34
	Bài tập chương 4	36
Phần II	Tri thức và lập luận	37
Chương 5	Logic mệnh đề	37
5.1	Biểu diễn tri thức	37
5.2	Cú pháp và ngữ nghĩa của logic mệnh đề	38
5.3	Dạng chuẩn tắc	41
5.4	Luật suy diễn	42
5.5	Luật phân giải, chứng minh bác bỏ bằng luật phân giải	43
	Bài tập chương 5	47
Chương 6	Logic vị từ cấp I	48
6.1	Cú pháp và ngữ nghĩa của logic vị từ cấp I	49
6.2	Chuẩn hóa công thức	52
6.3	Các luật suy diễn	53
6.4	Thuật toán hợp nhất	54
6.5	Luật phân giải, chứng minh bác bỏ bằng luật phân giải	54
	Bài tập chương 6	54
Chương 7	Biểu diễn tri thức và lập luận	55

7.1	Biểu diễn tri thức bởi các luật nếu – thì	55
7.2	Lập luận tiến và lập luận lùi trong các hệ dựa trên luật	56
7.3	Thủ tục lập luận tiến	56
7.4	Thủ tục lập luận lùi	61
	Bài tập chương 7	65

Tên học phần: TRÍ TUỆ NHÂN TẠO.**1. Số tín chỉ: 3 TC****2. Loại học phần: 2****3. Điều kiện tiên quyết:**

Sinh viên phải nắm được một ngôn ngữ lập trình và học môn CTDL và Thuật toán.

4. Mục đích của học phần:

- Cung cấp các kiến thức cơ bản về phương pháp biểu diễn và giải quyết vấn đề, các kỹ thuật biểu diễn và xử lý tri thức.

- Rèn luyện tư duy khoa học.

5. Nội dung chủ yếu:

Gồm 2 phần:

- Phương pháp biểu diễn và giải quyết vấn đề

- Kỹ thuật biểu diễn, xử lý tri thức và suy diễn

6. Người biên soạn:

KS Nguyễn Duy Trường Giang. Bộ môn Khoa học máy tính.

7. Nội dung chi tiết của học phần:

Tên chương mục	Phân phối số tiết				
	TS	LT	BT	TH	KT
MỞ ĐẦU					
PHẦN THỨ NHẤT: PHƯƠNG PHÁP BIỂU DIỄN VÀ GIẢI QUYẾT VẤN ĐỀ					
Chương 1. Các chiến lược tìm kiếm mù.	09	06		03	
1.1. Biểu diễn vấn đề trong không gian trạng thái		01			
1.2. Các chiến lược tìm kiếm		01			
1.3. Các chiến lược tìm kiếm mù		02			
1.3.1. Tìm kiếm theo bề rộng					
1.3.2. Tìm kiếm theo độ sâu					
1.3.3. Các trạng thái lặp					
1.3.4. Tìm kiếm sâu lặp					
1.4. Quy vấn đề về vấn đề con. Tìm kiếm trên đồ thị và/hoặc		02			
1.4.1. Quy vấn đề về các vấn đề con					
1.4.2. Đồ thị và/hoặc					
1.4.3. Tìm kiếm trên đồ thị và/hoặc					
Chương 2. Các chiến lược tìm kiếm kinh nghiệm.	06	04		02	
2.1. Hàm đánh giá và tìm kiếm kinh nghiệm		01			
2.2. Tìm kiếm tốt nhất đầu tiên		01			
2.3. Tìm kiếm leo đồi		01			
2.4. Tìm kiếm beam		01			
Chương 3. Các chiến lược tìm kiếm tối ưu.	08	05		02	01
3.1. Tìm đường đi ngắn nhất		02			
3.1.1. Thuật toán A*					
3.1.2. Thuật toán tìm kiếm nhánh – và – cận					
3.2. Tìm đối tượng tốt nhất		02			

3.2.1. Tìm kiếm leo đồi					
3.2.2. Tìm kiếm gradient					
3.3. Tìm kiếm mô phỏng sự tiến hóa. Thuật toán di truyền		01			01
Chương 4. Tìm kiếm có đối thủ.	08	06		02	
4.1. Cây trò chơi và tìm kiếm trên cây trò chơi		02			
4.2. Chiến lược Minimax		02			
4.3. Phương pháp cắt cụt alpha – beta		02			
PHẦN THỨ HAI: TRI THỨC VÀ LẬP LUẬN					
Chương 5. Logic mệnh đề.	06	06			
5.1. Biểu diễn tri thức		01			
5.2. Cú pháp và ngữ nghĩa của logic mệnh đề		02			
5.2.1. Cú pháp					
5.2.2. Ngữ nghĩa					
5.3. Dạng chuẩn tắc		1.5			
5.3.1. Sự tương đương của các công thức					
5.3.2. Dạng chuẩn tắc					
5.3.3. Các câu Horn					
5.4. Luật suy diễn		01			
5.5. Luật phân giải, chứng minh bác bỏ bằng luật phân giải		0.5			
Chương 6. Logic vị từ cấp I.	13	09		03	01
6.1. Cú pháp và ngữ nghĩa của logic vị từ cấp I		01			
6.1.1. Cú pháp					
6.1.2. Ngữ nghĩa					
6.2. Chuẩn hóa các công thức		01			
6.3. Các luật suy diễn		01			
6.4. Thuật toán hợp nhất		0.5			
6.5. Chứng minh bằng luật phân giải		0.5			
6.6. Các chiến lược phân giải		02			
6.6.1. Chiến lược phân giải theo bề rộng					
6.6.2. Chiến lược phân giải sử dụng tập hỗ trợ					
6.6.3. Chiến lược tuyến tính					
6.7. Sử dụng logic vị từ cấp I để biểu diễn tri thức		01			
6.7.1. Vị từ hằng					
6.7.2. Danh sách và các phép toán trên danh sách					
6.8. Xây dựng cơ sở tri thức		01			
6.9. Cài đặt cơ sở tri thức		01			
6.9.1. Cài đặt các hạng thức và các câu phân tử					
6.9.2. Cài đặt cơ sở tri thức					01
Chương 7. Biểu diễn tri thức và lập luận.	10	06		03	01
7.1. Biểu diễn tri thức bởi các luật nếu – thì		01			
7.2. Lập luận tiến và lập luận lùi trong các hệ dựa trên luật		02			
7.2.1. Lập luận tiến					
7.2.2. Lập luận lùi					
7.3. Thủ tục lập luận tiến		02			
7.3.1. Thủ tục For_chain					
7.3.2. Thủ tục rete					
7.3.3. Hệ hành động dựa trên luật					
7.4. Thủ tục lập luận lùi		01			01

8. Giáo trình và tài liệu tham khảo.

1. Đinh Mạnh Tường - *Trí tuệ nhân tạo* - Nhà xuất bản khoa học kỹ thuật - 2002
2. Nguyễn Thanh Thủy - *Trí tuệ nhân tạo* - Nhà xuất bản giáo dục - 1997
3. Ngô Trung Việt - *Trí tuệ nhân tạo* - Nhà xuất bản giáo dục - 1995

TaiLieu.vn

Phần I

Giải quyết vấn đề bằng tìm kiếm

Vấn đề tìm kiếm, một cách tổng quát, có thể hiểu là tìm một đối tượng thỏa mãn một số điều kiện nào đó, trong một tập hợp rộng lớn các đối tượng. Chúng ta có thể kể ra rất nhiều vấn đề mà việc giải quyết nó được quy về vấn đề tìm kiếm.

Các trò chơi, chẳng hạn cờ vua, cờ carô có thể xem như vấn đề tìm kiếm. Trong số rất nhiều nước đi được phép thực hiện, ta phải tìm ra các nước đi dẫn tới tình thế kết cuộc mà ta là người thắng.

Chứng minh định lý cũng có thể xem như vấn đề tìm kiếm. Cho một tập các tiên đề và các luật suy diễn, trong trường hợp này mục tiêu của ta là tìm ra một chứng minh (một dãy các luật suy diễn được áp dụng) để được đưa đến công thức mà ta cần chứng minh.

Trong các lĩnh vực nghiên cứu của **Trí Tuệ Nhân Tạo**, chúng ta thường xuyên phải đối đầu với vấn đề tìm kiếm. Đặc biệt trong lập kế hoạch và học máy, tìm kiếm đóng vai trò quan trọng.

Trong phần này chúng ta sẽ nghiên cứu các kỹ thuật tìm kiếm cơ bản được áp dụng để giải quyết các vấn đề và được áp dụng rộng rãi trong các lĩnh vực nghiên cứu khác của **Trí Tuệ Nhân Tạo**. Chúng ta lần lượt nghiên cứu các kỹ thuật sau:

- ☐ Các kỹ thuật tìm kiếm mù, trong đó chúng ta không có hiểu biết gì về các đối tượng để hướng dẫn tìm kiếm mà chỉ đơn thuần là xem xét theo một hệ thống nào đó tất cả các đối tượng để phát hiện ra đối tượng cần tìm.
- ☐ Các kỹ thuật tìm kiếm kinh nghiệm (tìm kiếm heuristic) trong đó chúng ta dựa vào kinh nghiệm và sự hiểu biết của chúng ta về vấn đề cần giải quyết để xây dựng nên hàm đánh giá hướng dẫn sự tìm kiếm.
- ☐ Các kỹ thuật tìm kiếm tối ưu.
- ☐ Các phương pháp tìm kiếm có đối thủ, tức là các chiến lược tìm kiếm nước đi trong các trò chơi hai người, chẳng hạn cờ vua, cờ tướng, cờ carô.

Chương 1

Các chiến lược tìm kiếm mù

Trong chương này, chúng tôi sẽ nghiên cứu các chiến lược tìm kiếm mù (blind search): tìm kiếm theo bề rộng (breadth-first search) và tìm kiếm theo độ sâu (depth-first search). Hiệu quả của các phương pháp tìm kiếm này cũng sẽ được đánh giá.

1.1 Biểu diễn vấn đề trong không gian trạng thái

Một khi chúng ta muốn giải quyết một vấn đề nào đó bằng tìm kiếm, đầu tiên ta phải xác định không gian tìm kiếm. Không gian tìm kiếm bao gồm tất cả các đối tượng mà ta cần quan tâm tìm kiếm. Nó có thể là không gian liên tục, chẳng hạn không gian các vectơ thực n chiều; nó cũng có thể là không gian các đối tượng rời rạc.

Trong mục này ta sẽ xét việc biểu diễn một vấn đề trong không gian trạng thái sao cho việc giải quyết vấn đề được quy về việc tìm kiếm trong không gian trạng thái.

Một phạm vi rộng lớn các vấn đề, đặc biệt các câu đố, các trò chơi, có thể mô tả bằng cách sử dụng khái niệm trạng thái và toán tử (phép biến đổi trạng thái). Chẳng hạn, một khách du lịch có trong tay bản đồ mạng lưới giao thông nối các thành phố trong một vùng lãnh thổ (hình 1.1), du khách đang ở thành phố A và anh ta muốn tìm đường đi tới thăm thành phố B. Trong bài toán này, các thành phố có trong các bản đồ là các trạng thái, thành phố A là trạng thái ban đầu, B là trạng thái kết thúc. Khi đang ở một thành phố, chẳng hạn ở thành phố D anh ta có thể đi theo các con đường để nối tới các thành phố C, F và G. Các con đường nối các thành phố sẽ được biểu diễn bởi các toán tử. Một toán tử biến đổi một trạng thái thành một trạng thái khác. Chẳng hạn, ở trạng thái D sẽ có ba toán tử dẫn trạng thái D tới các trạng thái C, F và G. Vấn đề của du khách bây giờ sẽ là tìm một dãy toán tử để đưa trạng thái ban đầu A tới trạng thái kết thúc B.

Một ví dụ khác, trong trò chơi cờ vua, mỗi cách bố trí các quân trên bàn cờ là một trạng thái. Trạng thái ban đầu là sự sắp xếp các quân lúc bắt đầu cuộc chơi. Mỗi nước đi hợp lệ là một toán tử, nó biến đổi một cảnh huống trên bàn cờ thành một cảnh huống khác.

Như vậy muốn biểu diễn một vấn đề trong không gian trạng thái, ta cần xác định các yếu tố sau:

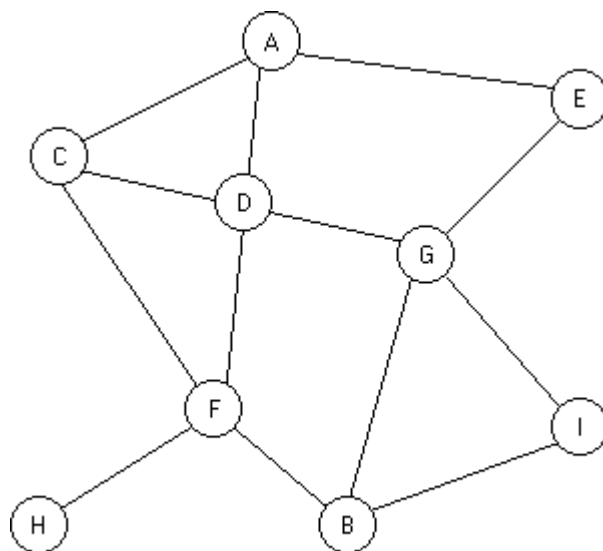
- ☐ Trạng thái ban đầu.
- ☐ Một tập hợp các toán tử. Trong đó mỗi toán tử mô tả một hành động hoặc một phép biến đổi có thể đưa một trạng thái tới một trạng thái khác.

Tập hợp tất cả các trạng thái có thể đạt tới từ trạng thái ban đầu bằng cách áp dụng một dãy toán tử, lập thành không gian trạng thái của vấn đề.

Ta sẽ ký hiệu không gian trạng thái là U , trạng thái ban đầu là u_0 ($u_0 \in U$). Mỗi toán tử R có thể xem như một ánh xạ $R: U \rightarrow U$. Nói chung R là một ánh xạ không xác định khắp nơi trên U .

- ☐ Một tập hợp T các trạng thái kết thúc (trạng thái đích). T là tập con của không gian U . Trong vấn đề của du khách trên, chỉ có một trạng thái đích, đó là thành phố B. Nhưng trong nhiều vấn đề (chẳng hạn các loại cờ) có thể có nhiều trạng thái đích và ta không thể xác định trước được các trạng thái đích. Nói chung trong phần lớn các vấn đề hay, ta chỉ có thể mô tả các trạng thái đích là các trạng thái thỏa mãn một số điều kiện nào đó.

Khi chúng ta biểu diễn một vấn đề thông qua các trạng thái và các toán tử, thì việc tìm nghiệm của bài toán được quy về việc tìm đường đi từ trạng thái ban đầu tới trạng thái đích. (Một đường đi trong không gian trạng thái là một dãy toán tử dẫn một trạng thái tới một trạng thái khác).



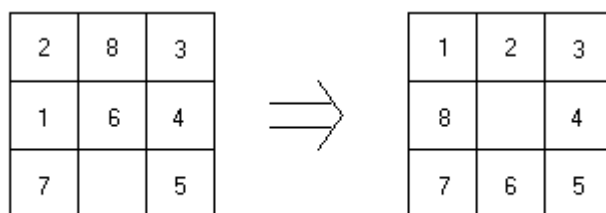
Hình 1.1 Tìm đường đi từ A đến B

Chúng ta có thể biểu diễn không gian trạng thái bằng đồ thị định hướng, trong đó mỗi đỉnh của đồ thị tương ứng với một trạng thái. Nếu có toán tử R biến đổi trạng thái u thành trạng thái v , thì có cung gắn nhãn R đi từ đỉnh u tới đỉnh v . Khi đó một đường đi trong không gian trạng thái sẽ là một đường đi trong đồ thị này.

Sau đây chúng ta sẽ xét một số ví dụ về các không gian trạng thái được xây dựng cho một số vấn đề.

Ví dụ 1: Bài toán 8 số. Chúng ta có bảng 3×3 ô và tám quân mang số hiệu từ 1 đến 8 được xếp vào tám ô, còn lại một ô trống, chẳng hạn như trong hình 2 bên trái. Trong trò chơi này, bạn có thể chuyển dịch các quân ở cạnh ô trống tới ô trống đó. Vấn đề của bạn là tìm ra một dãy các chuyển dịch để biến đổi cảnh huống ban đầu (hình 1.2 bên trái) thành một cảnh huống xác định nào đó, chẳng hạn cảnh huống trong hình 1.2 bên phải.

Trong bài toán này, trạng thái ban đầu là cảnh huống ở bên trái hình 1.2, còn trạng thái kết thúc ở bên phải hình 1.2. Tương ứng với các quy tắc chuyển dịch các quân, ta có bốn toán tử: **up**



Hình 1.2 Trạng thái ban đầu và trạng thái kết thúc của bài toán số.

(đẩy quân lên trên), **down** (đẩy quân xuống dưới), **left** (đẩy quân sang trái), **right** (đẩy quân sang phải). Rõ ràng là, các toán tử này chỉ là các toán tử bộ phận; chẳng hạn, từ trạng thái ban đầu (hình 1.2 bên trái), ta chỉ có thể áp dụng các toán tử **down**, **left**, **right**.

Trong các ví dụ trên việc tìm ra một biểu diễn thích hợp để mô tả các trạng thái của vấn đề là khá dễ dàng và tự nhiên. Song trong nhiều vấn đề việc tìm hiểu được biểu diễn thích hợp cho các trạng thái của vấn đề là hoàn toàn không đơn giản. Việc tìm ra dạng biểu diễn tốt cho các trạng thái đóng vai trò hết sức quan trọng trong quá trình giải quyết một vấn đề. Có thể nói rằng, nếu ta tìm được dạng biểu diễn tốt cho các trạng thái của vấn đề, thì vấn đề hầu như đã được giải quyết.

Ví dụ 2: Vấn đề triệu phú và kẻ cướp. Có ba nhà triệu phú và ba tên cướp ở bên bờ tả ngạn một con sông, cùng một chiếc thuyền chở được một hoặc hai người. Hãy tìm cách đưa mọi người qua sông sao cho không để lại ở bên bờ sông kẻ cướp nhiều hơn triệu phú. Đương nhiên trong bài toán này, các toán tử tương ứng với các hành động chở 1 hoặc 2 người qua sông. Nhưng ở đây ta cần lưu ý rằng, khi hành động xảy ra (lúc thuyền đang bơi qua sông) thì ở bên bờ sông thuyền vừa rời chỗ, số kẻ cướp không được nhiều hơn số triệu phú. Tiếp theo ta cần quyết định cái gì là trạng thái của vấn đề. ở đây ta không cần phân biệt các nhà triệu phú và các tên cướp, mà chỉ số lượng của họ ở bên bờ sông là quan trọng. Để biểu diễn các trạng thái, ta sử dụng bộ ba (a, b, k) , trong đó a là số triệu phú, b là số kẻ cướp ở bên bờ tả ngạn vào các thời điểm mà thuyền ở bờ này hoặc bờ kia, $k = 1$ nếu thuyền ở bờ tả ngạn và $k = 0$ nếu thuyền ở bờ hữu ngạn. Như vậy, không gian trạng thái cho bài toán triệu phú và kẻ cướp được xác định như sau:

- ☐ Trạng thái ban đầu là $(3, 3, 1)$.
- ☐ Các toán tử. Có năm toán tử tương ứng với hành động thuyền chở qua sông 1 triệu phú, hoặc 1 kẻ cướp, hoặc 2 triệu phú, hoặc 2 kẻ cướp, hoặc 1 triệu phú và 1 kẻ cướp.
- ☐ Trạng thái kết thúc là $(0, 0, 0)$.

1.2 Các chiến lược tìm kiếm

Như ta đã thấy trong mục 1.1, để giải quyết một vấn đề bằng tìm kiếm trong không gian trạng thái, đầu tiên ta cần tìm dạng thích hợp mô tả các trạng thái câu vấn đề. Sau đó cần xác định:

- ☐ Trạng thái ban đầu.
- ☐ Tập các toán tử.
- ☐ Tập T các trạng thái kết thúc. (T có thể không được xác định cụ thể gồm các trạng thái nào mà chỉ được chỉ định bởi một số điều kiện nào đó).

Giả sử u là một trạng thái nào đó và R là một toán tử biến đổi u thành v . Ta sẽ gọi v là trạng thái kế u , hoặc v được sinh ra từ trạng thái u bởi toán tử R . Quá trình áp dụng các toán tử để sinh ra các trạng thái kế u được gọi là phát triển trạng thái u . Chẳng hạn, trong bài toán toán số, phát triển trạng thái ban đầu (hình 2 bên trái), ta nhận được ba trạng thái kế (hình 1.3).

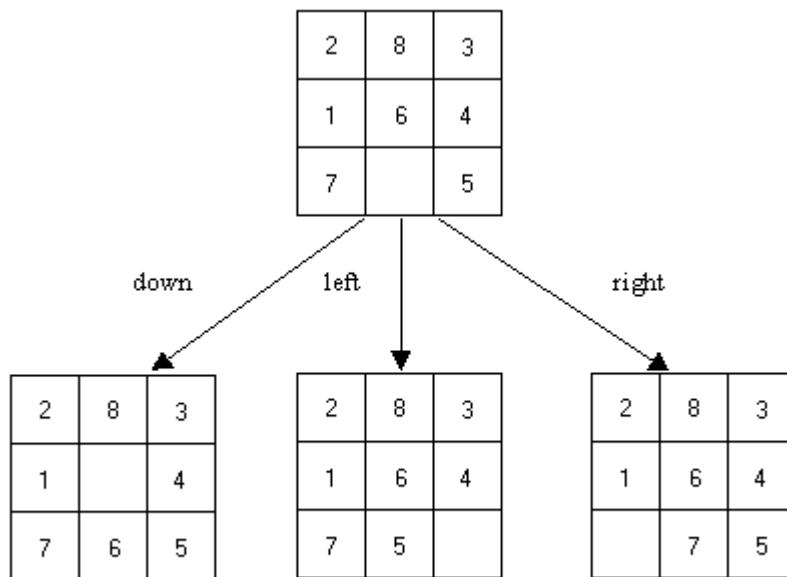
Khi chúng ta biểu diễn một vấn đề cần giải quyết thông qua các trạng thái và các toán tử thì việc tìm lời giải của vấn đề được quy về việc tìm đường đi từ trạng thái ban đầu tới một trạng thái kết thúc nào đó.

Có thể phân các chiến lược tìm kiếm thành hai loại:

- ☐ Các chiến lược tìm kiếm mù. Trong các chiến lược tìm kiếm này, không có một sự hướng dẫn nào cho sự tìm kiếm, mà ta chỉ phát triển các trạng thái ban đầu cho tới khi gặp một trạng thái đích nào đó. Có hai kỹ thuật tìm kiếm mù, đó là tìm kiếm theo bề rộng và tìm kiếm theo độ sâu.

Tư tưởng của tìm kiếm theo bề rộng là các trạng thái được phát triển theo thứ tự mà chúng được sinh ra, tức là trạng thái nào được sinh ra trước sẽ được phát triển trước.

Trong nhiều vấn đề, dù chúng ta phát triển các trạng thái theo hệ thống nào (theo bề rộng



Hình 1.3 Phát triển một trạng thái

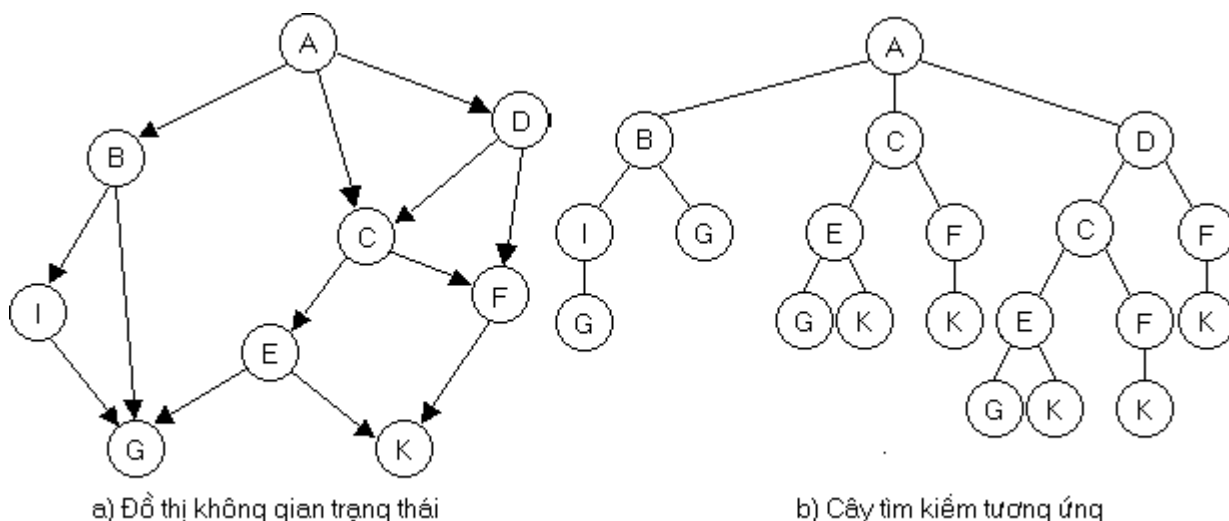
hoặc theo độ sâu) thì số lượng các trạng thái được sinh ra trước khi ta gặp trạng thái đích thường là cực kỳ lớn. Do đó các thuật toán tìm kiếm mù kém hiệu quả, đòi hỏi rất nhiều không gian và thời gian. Trong thực tế, nhiều vấn đề không thể giải quyết được bằng tìm kiếm mù.

□ Tìm kiếm kinh nghiệm (tìm kiếm heuristic). Trong rất nhiều vấn đề, chúng ta có thể dựa vào sự hiểu biết của chúng ta về vấn đề, dựa vào kinh nghiệm, trực giác, để đánh giá các trạng thái. Sử dụng sự đánh giá các trạng thái để hướng dẫn sự tìm kiếm: trong quá trình phát triển các trạng thái, ta sẽ chọn trong số các trạng thái chờ phát triển, trạng thái được đánh giá là tốt nhất để phát triển. Do đó tốc độ tìm kiếm sẽ nhanh hơn. Các phương pháp tìm kiếm dựa vào sự đánh giá các trạng thái để hướng dẫn sự tìm kiếm gọi chung là các phương pháp tìm kiếm kinh nghiệm.

Như vậy chiến lược tìm kiếm được xác định bởi chiến lược chọn trạng thái để phát triển ở mỗi bước. Trong tìm kiếm mù, ta chọn trạng thái để phát triển theo thứ tự mà chúng được sinh ra; còn trong tìm kiếm kinh nghiệm ta chọn trạng thái dựa vào sự đánh giá các trạng thái.

Cây tìm kiếm

Chúng ta có thể nghĩ đến quá trình tìm kiếm như quá trình xây dựng *cây tìm kiếm*. Cây tìm kiếm là cây mà các đỉnh được gắn bởi các trạng thái của không gian trạng thái. Gốc của cây tìm kiếm tương ứng với trạng thái ban đầu. Nếu một đỉnh ứng với trạng thái u , thì các đỉnh con của nó ứng với các trạng thái v kề u . Hình 1.4a là đồ thị biểu diễn một không gian trạng thái với



Hình 1.4

trạng thái ban đầu là A, hình 1.4b là cây tìm kiếm tương ứng với không gian trạng thái đó.

Mỗi chiến lược tìm kiếm trong không gian trạng thái tương ứng với một phương pháp xây dựng cây tìm kiếm. Quá trình xây dựng cây bắt đầu từ cây chỉ có một đỉnh là trạng thái ban đầu. Giả sử tới một bước nào đó trong chiến lược tìm kiếm, ta đã xây dựng được một cây nào đó, các lá của cây tương ứng với các trạng thái chưa được phát triển. Bước tiếp theo phụ thuộc vào chiến lược tìm kiếm mà một đỉnh nào đó trong các lá được chọn để phát triển. Khi phát triển đỉnh đó, cây tìm kiếm được mở rộng bằng cách thêm vào các đỉnh con của đỉnh đó. Kỹ thuật tìm kiếm theo bề rộng (theo độ sâu) tương ứng với phương pháp xây dựng cây tìm kiếm theo bề rộng (theo độ sâu).

1.3 Các chiến lược tìm kiếm mù

Trong mục này chúng ta sẽ trình bày hai chiến lược tìm kiếm mù: tìm kiếm theo bề rộng và tìm kiếm theo độ sâu. Trong tìm kiếm theo bề rộng, tại mỗi bước ta sẽ chọn trạng thái để phát triển là trạng thái được sinh ra trước các trạng thái chờ phát triển khác. Còn trong tìm kiếm theo độ sâu, trạng thái được chọn để phát triển là trạng thái được sinh ra sau cùng trong số các trạng thái chờ phát triển.

Chúng ta sử dụng danh sách L để lưu các trạng thái đã được sinh ra và chờ được phát triển. Mục tiêu của tìm kiếm trong không gian trạng thái là tìm đường đi từ trạng thái ban đầu tới trạng thái đích, do đó ta cần lưu lại vết của đường đi. Ta có thể sử dụng hàm *father* để lưu lại cha của mỗi đỉnh trên đường đi, ***father***(v) = u nếu cha của đỉnh v là u.

1.3.1 Tìm kiếm theo bề rộng

Thuật toán tìm kiếm theo bề rộng được mô tả bởi thủ tục sau:

procedure *Breadth_First_Search*;

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu;

2. **loop do**

 2.1 **if** L rỗng **then**

 {thông báo tìm kiếm thất bại; **stop**};

 2.2 Loại trạng thái u ở đầu danh sách L;

 2.3 **if** u là trạng thái kết thúc **then**

 {thông báo tìm kiếm thành công; **stop**};

 2.4 **for** mỗi trạng thái v kề u **do** {

 Đặt v vào cuối danh sách L;

father(v) <- u}

end;

Chúng ta có một số nhận xét sau đây về thuật toán tìm kiếm theo bề rộng:

□ Trong tìm kiếm theo bề rộng, trạng thái nào được sinh ra trước sẽ được phát triển trước, do đó danh sách L được xử lý như hàng đợi. Trong bước 2.3, ta cần kiểm tra xem u có là trạng thái kết thúc hay không. Nói chung các trạng thái kết thúc được xác định bởi một số điều kiện nào đó, khi đó ta cần kiểm tra xem u có thỏa mãn các điều kiện đó hay không.

□ Nếu bài toán có nghiệm (tồn tại đường đi từ trạng thái ban đầu tới trạng thái đích), thì thuật toán tìm kiếm theo bề rộng sẽ tìm ra nghiệm, đồng thời đường đi tìm được sẽ là ngắn nhất. Trong trường hợp bài toán vô nghiệm và không gian trạng thái hữu hạn, thuật toán sẽ dừng và cho thông báo vô nghiệm.

Đánh giá tìm kiếm theo bề rộng

Bây giờ ta đánh giá thời gian và bộ nhớ mà tìm kiếm theo bề rộng đòi hỏi. Giả sử rằng, mỗi trạng thái khi được phát triển sẽ sinh ra b trạng thái kế. Ta sẽ gọi b là **nhân tố nhánh**. Giả sử rằng, nghiệm của bài toán là đường đi có độ dài d . Bởi nhiều nghiệm có thể được tìm ra tại một đỉnh bất kỳ ở mức d của cây tìm kiếm, do đó số đỉnh cần xem xét để tìm ra nghiệm là:

$$1 + b + b^2 + \dots + b^{d-1} + k$$

Trong đó k có thể là $1, 2, \dots, b^d$. Do đó số lớn nhất các đỉnh cần xem xét là:

$$1 + b + b^2 + \dots + b^d$$

Như vậy, độ phức tạp thời gian của thuật toán tìm kiếm theo bề rộng là $O(b^d)$. Độ phức tạp không gian cũng là $O(b^d)$, bởi vì ta cần lưu vào danh sách L tất cả các đỉnh của cây tìm kiếm ở mức d , số các đỉnh này là b^d .

Để thấy rõ tìm kiếm theo bề rộng đòi hỏi thời gian và không gian lớn tới mức nào, ta xét trường hợp nhân tố nhánh $b = 10$ và độ sâu d thay đổi. Giả sử để phát hiện và kiểm tra 1000 trạng thái cần 1 giây, và lưu giữ 1 trạng thái cần 100 bytes. Khi đó thời gian và không gian mà thuật toán đòi hỏi được cho trong bảng sau:

Độ sâu d	Thời gian	Không gian
4	11 giây	1 megabyte
6	18 giây	111 megabytes
8	31 giờ	11 gigabytes
10	128 ngày	1 terabyte
12	35 năm	111 terabytes
14	3500 năm	11.111 terabytes

1.3.2 Tìm kiếm theo độ sâu

Như ta đã biết, tư tưởng của chiến lược tìm kiếm theo độ sâu là, tại mỗi bước trạng thái được chọn để phát triển là trạng thái được sinh ra sau cùng trong số các trạng thái chờ phát triển. Do đó thuật toán tìm kiếm theo độ sâu là hoàn toàn tương tự như thuật toán tìm kiếm theo bề rộng, chỉ có một điều khác là, ta xử lý danh sách L các trạng thái chờ phát triển không phải như hàng đợi mà như ngăn xếp. Cụ thể là trong bước 2.4 của thuật toán tìm kiếm theo bề rộng, ta cần sửa lại là “Đặt v vào **đầu** danh sách L ”.

Sau đây chúng ta sẽ đưa ra các nhận xét so sánh hai chiến lược tìm kiếm mù:

□ Thuật toán tìm kiếm theo bề rộng luôn luôn tìm ra nghiệm nếu bài toán có nghiệm. Song không phải với bất kỳ bài toán có nghiệm nào thuật toán tìm kiếm theo độ sâu cũng tìm ra nghiệm! Nếu bài toán có nghiệm và không gian trạng thái hữu hạn, thì thuật toán tìm kiếm theo độ sâu sẽ tìm ra nghiệm. Tuy nhiên, trong trường hợp không gian trạng thái vô hạn, thì có thể nó không tìm ra nghiệm, lý do là ta luôn luôn đi xuống theo độ sâu, nếu ta đi theo một nhánh vô hạn mà nghiệm không nằm trên nhánh đó thì thuật toán sẽ không dừng. Do đó người ta khuyên rằng, không nên áp dụng tìm kiếm theo độ sâu cho các bài toán có cây tìm kiếm chứa các nhánh vô hạn.

□ Độ phức tạp của thuật toán tìm kiếm theo độ sâu.

Giả sử rằng, nghiệm của bài toán là đường đi có độ dài d , cây tìm kiếm có nhân tố nhánh là b và có chiều cao là d . Có thể xảy ra, nghiệm là đỉnh ngoài cùng bên phải trên mức d của cây tìm kiếm, do đó độ phức tạp thời gian của tìm kiếm theo độ sâu trong trường hợp xấu nhất là $O(b^d)$, tức là cũng như tìm kiếm theo bề rộng. Tuy nhiên, trên thực tế đối với nhiều bài toán, tìm kiếm theo độ sâu thực sự nhanh hơn tìm kiếm theo bề rộng. Lý do là tìm kiếm theo bề rộng phải xem

xét toàn bộ cây tìm kiếm tới mức $d-1$, rồi mới xem xét các đỉnh ở mức d . Còn trong tìm kiếm theo độ sâu, có thể ta chỉ cần xem xét một bộ phận nhỏ của cây tìm kiếm thì đã tìm ra nghiệm.

Để đánh giá độ phức tạp không gian của tìm kiếm theo độ sâu ta có nhận xét rằng, khi ta phát triển một đỉnh u trên cây tìm kiếm theo độ sâu, ta chỉ cần lưu các đỉnh chưa được phát triển mà chúng là các đỉnh con của các đỉnh nằm trên đường đi từ gốc tới đỉnh u . Như vậy đối với cây tìm kiếm có nhân tố nhánh b và độ sâu lớn nhất là d , ta chỉ cần lưu ít hơn db đỉnh. Do đó độ phức tạp không gian của tìm kiếm theo độ sâu là $O(db)$, trong khi đó tìm kiếm theo bề rộng đòi hỏi không gian nhớ $O(b^d)$!

1.3.3 Các trạng thái lặp

Như ta thấy trong mục 1.2, cây tìm kiếm có thể chứa nhiều đỉnh ứng với cùng một trạng thái, các trạng thái này được gọi là trạng thái lặp. Chẳng hạn, trong cây tìm kiếm hình 4b, các trạng thái C, E, F là các trạng thái lặp. Trong đồ thị biểu diễn không gian trạng thái, các trạng thái lặp ứng với các đỉnh có nhiều đường đi dẫn tới nó từ trạng thái ban đầu. Nếu đồ thị có chu trình thì cây tìm kiếm sẽ chứa các nhánh với một số đỉnh lặp lại vô hạn lần. Trong các thuật toán tìm kiếm sẽ lãng phí rất nhiều thời gian để phát triển lại các trạng thái mà ta đã gặp và đã phát triển. Vì vậy trong quá trình tìm kiếm ta cần tránh phát sinh ra các trạng thái mà ta đã phát triển. Chúng ta có thể áp dụng một trong các giải pháp sau đây:

1. Khi phát triển đỉnh u , không sinh ra các đỉnh trùng với cha của u .
2. Khi phát triển đỉnh u , không sinh ra các đỉnh trùng với một đỉnh nào đó nằm trên đường đi dẫn tới u .
3. Không sinh ra các đỉnh mà nó đã được sinh ra, tức là chỉ sinh ra các đỉnh mới.

Hai giải pháp đầu dễ cài đặt và không tốn nhiều không gian nhớ, tuy nhiên các giải pháp này không tránh được hết các trạng thái lặp.

Để thực hiện giải pháp thứ 3 ta cần lưu các trạng thái đã phát triển vào tập Q , lưu các trạng thái chờ phát triển vào danh sách L . Đương nhiên, trạng thái v lần đầu được sinh ra nếu nó không có trong Q và L . Việc lưu các trạng thái đã phát triển và kiểm tra xem một trạng thái có phải lần đầu được sinh ra không đòi hỏi rất nhiều không gian và thời gian. Chúng ta có thể cài đặt tập Q bởi bảng băm (xem [1]).

1.3.4 Tìm kiếm sâu lặp

Như chúng ta đã nhận xét, nếu cây tìm kiếm chứa nhánh vô hạn, khi sử dụng tìm kiếm theo độ sâu, ta có thể mắc kẹt ở nhánh đó và không tìm ra nghiệm. Để khắc phục hoàn cảnh đó, ta tìm kiếm theo độ sâu chỉ tới mức 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 một độ sâu max nào đó. Như vậy, thuật toán tìm kiếm sâu lặp (iterative deepening search) sẽ sử dụng thủ tục tìm kiếm sâu hạn chế (depth_limited search) như thủ tục con. Đó là thủ tục tìm kiếm theo độ sâu, nhưng chỉ đi tới độ sâu d nào đó rồi quay lên.

Trong thủ tục tìm kiếm sâu hạn chế, d là tham số độ sâu, hàm depth ghi lại độ sâu của mỗi đỉnh

procedure Depth_Limited_Search(d);

begin

1. Khởi tạo danh sách L chỉ chứa trạng thái ban đầu u_0 ;

depth(u_0) $\leftarrow 0$;

2. **loop do**

2.1 **if** L rỗng **then**

{thông báo thất bại; **stop**};

2.2 Loại trạng thái u ở đầu danh sách L ;


```

2.3 if u là trạng thái kết thúc then
    {thông báo thành công; stop};
2.4 if depth(u) <= d then
    for mỗi trạng thái v kề u do
        {Đặt v vào đầu danh sách L;
         depth(v) ← depth(u) + 1};
end;

procedure Depth_Deepening_Search;
begin
for d ← 0 to max do
    {Depth_Limited_Search(d);
    if thành công then exit}
end;

```

Kỹ thuật tìm kiếm sâu lặp kết hợp được các ưu điểm của tìm kiếm theo bề rộng và tìm kiếm theo độ sâu. Chúng ta có một số nhận xét sau:

- ☐ Cũng như tìm kiếm theo bề rộng, tìm kiếm sâu lặp luôn luôn tìm ra nghiệm (nếu bài toán có nghiệm), miễn là ta chọn độ sâu mã đủ lớn.
- ☐ Tìm kiếm sâu lặp chỉ cần không gian nhớ như tìm kiếm theo độ sâu.
- ☐ Trong tìm kiếm sâu lặp, ta phải phát triển lặp lại nhiều lần cùng một trạng thái. Điều đó làm cho ta có cảm giác rằng, tìm kiếm sâu lặp lãng phí nhiều thời gian. Thực ra thời gian tiêu tốn cho phát triển lặp lại các trạng thái là không đáng kể so với thời gian tìm kiếm theo bề rộng. Thật vậy, mỗi lần gọi thủ tục tìm kiếm sâu hạn chế tới mức d, nếu cây tìm kiếm có nhân tố nhánh là b, thì số đỉnh cần phát triển là:

$$1 + b + b^2 + \dots + b^d$$

Nếu nghiệm ở độ sâu d, thì trong tìm kiếm sâu lặp, ta phải gọi thủ tục tìm kiếm sâu hạn chế với độ sâu lần lượt là 0, 1, 2, ..., d. Do đó các đỉnh ở mức 1 phải phát triển lặp d lần, các đỉnh ở mức 2 lặp d-1 lần, ..., các đỉnh ở mức d lặp 1 lần. Như vậy tổng số đỉnh cần phát triển trong tìm kiếm sâu lặp là:

$$(d+1)1 + db + (d-1)b^2 + \dots + 2b^{d-1} + 1b^d$$

Do đó thời gian tìm kiếm sâu lặp là $O(b^d)$.

Tóm lại, tìm kiếm sâu lặp có độ phức tạp thời gian là $O(b^d)$ (như tìm kiếm theo bề rộng), và có độ phức tạp không gian là $O(b^d)$ (như tìm kiếm theo độ sâu). Nói chung, chúng ta nên áp dụng tìm kiếm sâu lặp cho các vấn đề có không gian trạng thái lớn và độ sâu của nghiệm không biết trước.

1.4 Quy vấn đề về các vấn đề con. Tìm kiếm trên đồ thị và/hoặc.

1.4.1 Quy vấn đề về các vấn đề con:

Trong mục 1.1, chúng ta đã nghiên cứu việc biểu diễn vấn đề thông qua các trạng thái và các toán tử. Khi đó việc tìm nghiệm của vấn đề được quy về việc tìm đường trong không gian trạng thái. Trong mục này chúng ta sẽ nghiên cứu một phương pháp luận khác để giải quyết vấn đề, dựa trên việc quy vấn đề về các vấn đề con. Quy vấn đề về các vấn đề con (còn gọi là rút gọn vấn đề) là một phương pháp được sử dụng rộng rãi nhất để giải quyết các vấn đề. Trong đời sống hàng ngày, cũng như trong khoa học kỹ thuật, mỗi khi gặp một vấn đề cần giải quyết, ta vẫn thường cố gắng tìm cách đưa nó về các vấn đề đơn giản hơn. Quá trình rút gọn vấn đề sẽ được

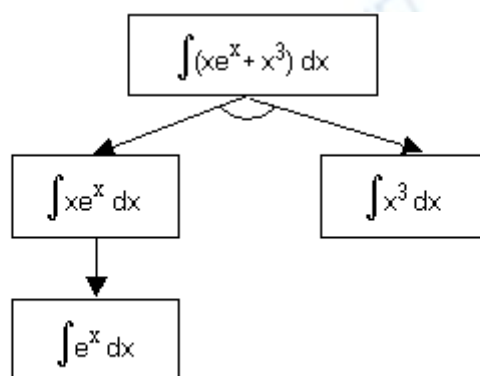
tiếp tục cho tới khi ta dẫn tới các vấn đề con có thể giải quyết được dễ dàng. Sau đây chúng ta xét một số vấn đề.

Vấn đề tính tích phân bất định

Giả sử ta cần tính một tích phân bất định, chẳng hạn $\int (xe^x + x^3) dx$. Quá trình chúng ta vẫn thường làm để tính tích phân bất định là như sau. Sử dụng các quy tắc tính tích phân (quy tắc tính tích phân của một tổng, quy tắc tính tích phân từng phần...), sử dụng các phép biến đổi biến số, các phép biến đổi các hàm (chẳng hạn, các phép biến đổi lượng giác),... để đưa tích phân cần tính về tích phân của các hàm số sơ cấp mà chúng ta đã biết cách tính. Chẳng hạn, đối với tích phân $\int (xe^x + x^3) dx$, áp dụng quy tắc tính tích phân của tổng ta đưa về hai tích phân $\int xe^x dx$ và $\int x^3 dx$. áp dụng quy tắc tính tích phân từng phần ta đưa tích phân $\int xe^x dx$ về tích phân $\int e^x dx$. Quá trình trên có thể biểu diễn bởi đồ thị trong hình 1.5.

Các tích phân $\int e^x dx$ và $\int x^3 dx$ là các tích phân cơ bản đã có trong bảng tích phân. Kết hợp các kết quả của các tích phân cơ bản, ta nhận được kết quả của tích phân đã cho.

Chúng ta có thể biểu diễn việc quy một vấn đề về các vấn đề con cơ bởi các trạng thái và



Hình 1.5 Quy một số tích phân về các tích phân cơ bản.

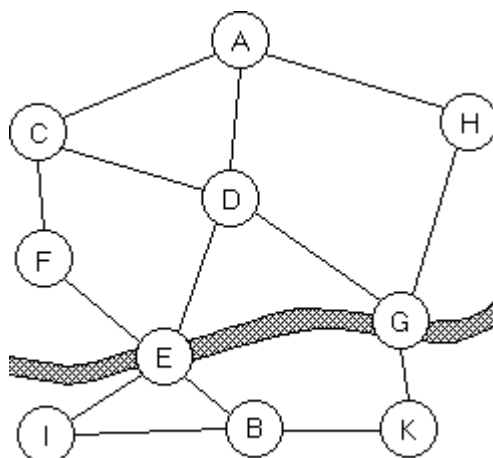
các toán tử. ở đây, bài toán cần giải là trạng thái ban đầu. Mỗi cách quy bài toán về các bài toán con được biểu diễn bởi một toán tử, toán tử $A \rightarrow B, C$ biểu diễn việc quy bài toán A về hai bài toán B và C. Chẳng hạn, đối với bài toán tính tích phân bất định, ta có thể xác định các toán tử dạng:

$$\int (f_1 + f_2) dx \rightarrow \int f_1 dx, \int f_2 dx \quad \text{và} \quad \int u dv \rightarrow \int v du$$

Các trạng thái kết thúc là các bài toán sơ cấp (các bài toán đã biết cách giải). Chẳng hạn, trong bài toán tính tích phân, các tích phân cơ bản là các trạng thái kết thúc. Một điều cần lưu ý là, trong không gian trạng thái biểu diễn việc quy vấn đề về các vấn đề con, các toán tử có thể là đa trị, nó biến đổi một trạng thái thành nhiều trạng thái khác.

Vấn đề tìm đường đi trên bản đồ giao thông

Bài toán này đã được phát triển như bài toán tìm đường đi trong không gian trạng thái (xem 1.1), trong đó mỗi trạng thái ứng với một thành phố, mỗi toán tử ứng với một con đường nối, nối thành phố này với thành phố khác. Bây giờ ta đưa ra một cách biểu diễn khác dựa trên



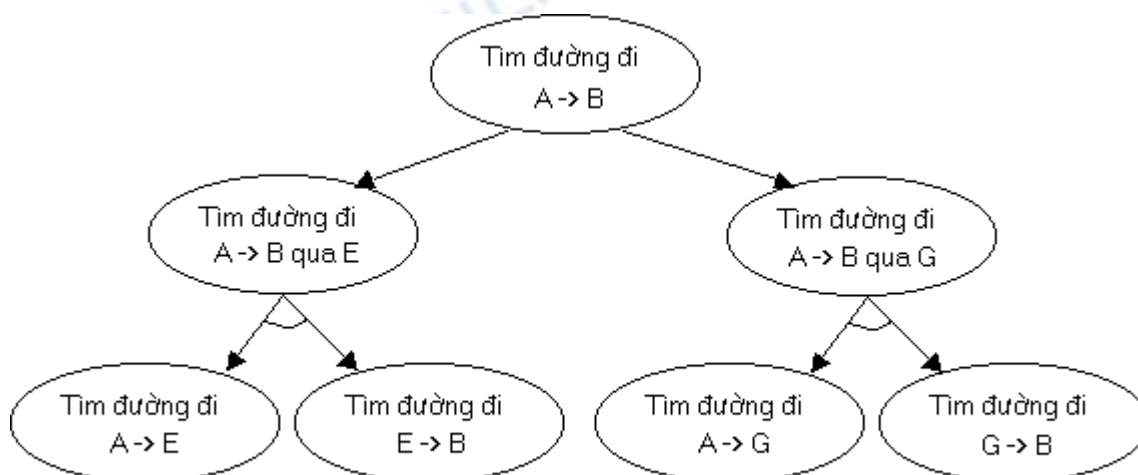
Hình 1.6

việc quy vấn đề về các vấn đề con. Giả sử ta có bản đồ giao thông trong một vùng lãnh thổ (xem hình 1.6). Giả sử ta cần tìm đường đi từ thành phố A tới thành phố B. Có con sông chảy qua hai thành phố E và G và có cầu qua sông ở mỗi thành phố đó. Mọi đường đi từ A đến B chỉ có thể qua E hoặc G. Như vậy bài toán tìm đường đi từ A đến B được quy về:

- 1) Bài toán tìm đường đi từ A đến B qua E (hoặc)
 - 2) Bài toán tìm đường đi từ A đến B qua G.
- Mỗi một trong hai bài toán trên lại có thể phân nhỏ như sau
- 1) Bài toán tìm đường đi từ A đến B qua E được quy về:
 - 1.1 Tìm đường đi từ A đến E (và)
 - 1.2 Tìm đường đi từ E đến B.
 - 2) Bài toán tìm đường đi từ A đến B qua G được quy về:
 - 2.1 Tìm đường đi từ A đến G (và)
 - 2.2 Tìm đường đi từ G đến B.

Quá trình rút gọn vấn đề như trên có thể biểu diễn dưới dạng đồ thị (đồ thị và/hoặc) trong hình 1.7. ở đây mỗi bài toán tìm đường đi từ một thành phố tới một thành phố khác ứng với một trạng thái. Các trạng thái kết thúc là các trạng thái ứng với các bài toán tìm đường đi, chẳng hạn từ A đến C, hoặc từ D đến E, bởi vì đã có đường nối A với C, nối D với E.

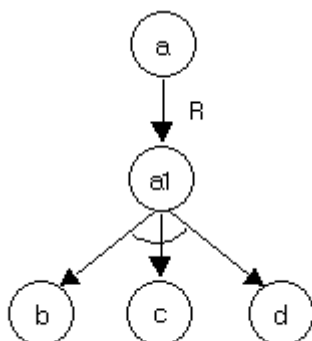
1.4.2 Đồ thị và/hoặc



Hình 1.7 Đồ thị và/hoặc của vấn đề tìm đường đi.

Không gian trạng thái mô tả việc quy vấn đề về các vấn đề con có thể biểu diễn dưới dạng đồ thị định hướng đặc biệt được gọi là đồ thị và/hoặc. Đồ thị này được xây dựng như sau:

Mỗi bài toán ứng với một đỉnh của đồ thị. Nếu có một toán tử quy một bài toán về một bài toán khác, chẳng hạn $R : a \rightarrow b$, thì trong đồ thị sẽ có cung gán nhãn đi từ đỉnh a tới đỉnh b. Đối với mỗi toán tử quy một bài toán về một số bài toán con, chẳng hạn $R : a \rightarrow b, c, d$ ta đưa vào một đỉnh mới a_1 , đỉnh này biểu diễn tập các bài toán con $\{b, c, d\}$ và toán tử $R : a \rightarrow b, c, d$ được biểu diễn bởi đồ thị hình 1.8.



Hình 1.8 Đồ thị biểu diễn toán tử $R : a \rightarrow b, c, d$

Ví dụ: Giả sử chúng ta có không gian trạng thái sau:

□ Trạng thái ban đầu (bài toán cần giải) là a.

□ Tập các toán tử quy gồm:

$R_1 : a \rightarrow d, e, f$

$R_2 : a \rightarrow d, k$

$R_3 : a \rightarrow g, h$

$R_4 : d \rightarrow b, c$

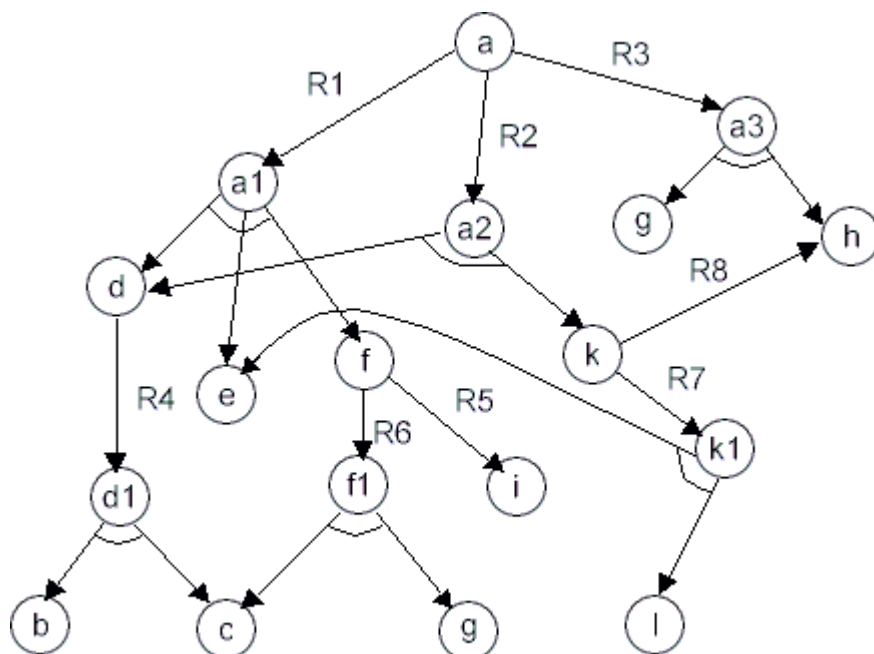
$R_5 : f \rightarrow i$

$R_6 : f \rightarrow c, j$

$R_7 : k \rightarrow e, l$

$R_8 : k \rightarrow h$

□ Tập các trạng thái kết thúc (các bài toán sơ cấp) là $T = \{b, c, e, j, l\}$.

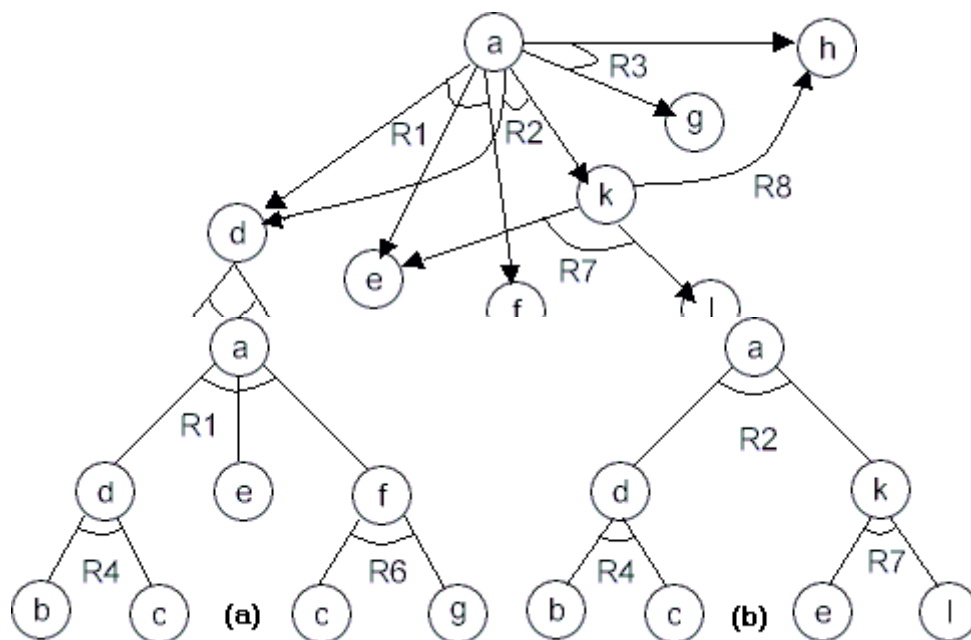


Hình 1.9 Một đồ thị và/hoặc

Không gian trạng thái trên có thể biểu diễn bởi đồ thị và/hoặc trong hình 1.9. Trong đồ thị đó, các đỉnh, chẳng hạn a_1, a_2, a_3 được gọi là đỉnh **và**, các đỉnh chẳng hạn a, f, k được gọi là đỉnh **hoặc**. Lý do là, đỉnh a_1 biểu diễn tập các bài toán $\{d, e, f\}$ và a_1 được giải quyết nếu d và e và f được giải quyết. Còn tại đỉnh a , ta có các toán tử R_1, R_2, R_3 quy bài toán a về các bài toán con khác nhau, do đó a được giải quyết nếu hoặc $a_1 = \{d, e, f\}$, hoặc $a_2 = \{d, k\}$, hoặc $a_3 = \{g, h\}$ được giải quyết.

Người ta thường sử dụng đồ thị và/hoặc ở dạng rút gọn. Chẳng hạn, đồ thị và/hoặc trong hình 1.9 có thể rút gọn thành đồ thị trong hình 1.10. Trong đồ thị rút gọn này, ta sẽ nói chẳng hạn d, e, f là các đỉnh kề đỉnh a theo toán tử R_1 , còn d, k là các đỉnh kề a theo toán tử R_2 .

Khi đã có các toán tử rút gọn vấn đề, thì bằng cách áp dụng liên tiếp các toán tử, ta có thể



Hình 1.11 Các cây nghiệm

đưa bài toán cần giải về một tập các bài toán con. Chẳng hạn, trong ví dụ trên nếu ta áp dụng các toán tử R_1, R_4, R_6 , ta sẽ quy bài toán a về tập các bài toán con $\{b, c, e, f\}$, tất cả các bài toán con này đều là sơ cấp. Từ các toán tử R_1, R_4 và R_6 ta xây dựng được một cây trong hình 1.11a, cây này được gọi là cây nghiệm. Cây nghiệm được định nghĩa như sau:

Cây nghiệm là một cây, trong đó:

- ☐ Gốc của cây ứng với bài toán cần giải.
- ☐ Tất cả các lá của cây là các đỉnh kết thúc (đỉnh ứng với các bài toán sơ cấp).
- ☐ Nếu u là đỉnh trong của cây, thì các con của u là các đỉnh v theo một toán tử nào đó.

Các đỉnh của đồ thị và/hoặc sẽ được gắn nhãn giải được hoặc không giải được.

Các đỉnh **giải được** được xác định đệ quy như sau:

- ☐ Các đỉnh kết thúc là các đỉnh **giải được**.
- ☐ Nếu u không phải là đỉnh kết thúc, nhưng có một toán tử R sao cho tất cả các đỉnh v theo R đều giải được thì u **giải được**.

Các đỉnh **không giải được** được xác định đệ quy như sau:

- ☐ Các đỉnh không phải là đỉnh kết thúc và không có đỉnh v theo R giải được, là các đỉnh **không giải được**.
- ☐ Nếu u không phải là đỉnh kết thúc và với mọi toán tử R áp dụng được tại u đều có một đỉnh v theo R không giải được, thì u **không giải được**.

Ta có nhận xét rằng, nếu bài toán a **giải được** thì sẽ có một cây nghiệm gốc a , và ngược lại nếu có một cây nghiệm gốc a thì a **giải được**. Hiển nhiên là, một bài toán giải được có thể có nhiều cây nghiệm, mỗi cây nghiệm biểu diễn một cách giải bài toán đó. Chẳng hạn trong ví dụ đã nêu, bài toán a có hai cây nghiệm trong hình 1.11.

Thứ tự giải các bài toán con trong một cây nghiệm là như sau. Bài toán ứng với đỉnh u chỉ được giải sau khi tất cả các bài toán ứng với các đỉnh con của u đã được giải. Chẳng hạn, với cây nghiệm trong hình 1.11a, thứ tự giải các bài toán có thể là b, c, d, j, f, e, a . ta có thể sử dụng thủ tục sắp xếp topo để sắp xếp thứ tự các bài toán trong một cây nghiệm. Đương nhiên ta cũng có thể giải quyết đồng thời các bài toán con ở cùng một mức trong cây nghiệm.

Vấn đề của chúng ta bây giờ là, tìm kiếm trên đồ thị và/hoặc để xác định được đỉnh ứng với bài toán ban đầu là giải được hay không giải được, và nếu nó giải được thì xây dựng một cây nghiệm cho nó.

1.4.3 Tìm kiếm trên đồ thị và/hoặc

Ta sẽ sử dụng kỹ thuật tìm kiếm theo độ sâu trên đồ thị và/hoặc để đánh dấu các đỉnh. Các đỉnh sẽ được đánh dấu giải được hoặc không giải được theo định nghĩa đệ quy về đỉnh giải được và không giải được. Xuất phát từ đỉnh ứng với bài toán ban đầu, đi xuống theo độ sâu, nếu gặp đỉnh u là đỉnh kết thúc thì nó được đánh dấu giải được. Nếu gặp đỉnh u không phải là đỉnh kết thúc và từ u không đi tiếp được, thì u được đánh dấu không giải được. Khi đi tới đỉnh u , thì từ u ta lần lượt đi xuống các đỉnh v theo một toán tử R nào đó. Nếu đánh dấu được một đỉnh v không giải được thì không cần đi tiếp xuống các đỉnh v còn lại. Tiếp tục đi xuống các đỉnh v theo một toán tử khác. Nếu tất cả các đỉnh v theo một toán tử nào đó được đánh dấu giải được thì u sẽ được đánh dấu giải được và quay lên cha của u . Còn nếu từ u đi xuống các đỉnh v theo mọi toán tử đều gặp các đỉnh v được đánh dấu không giải được, thì u được đánh dấu không giải được và quay lên cha của u .

Ta sẽ biểu diễn thủ tục tìm kiếm theo độ sâu và đánh dấu các đỉnh đã trình bày trên bởi hàm đệ quy Solvable(u). Hàm này nhận giá trị true nếu u giải được và nhận giá trị **false** nếu u không giải được. Trong hàm Solvable(u), ta sẽ sử dụng:

- ☐ Với mỗi toán tử R áp dụng được tại u , biến Ok nhận giá trị **true** nếu tất cả các đỉnh v theo R đều giải được, và Ok nhận giá trị **false** nếu có một đỉnh v theo R không giải được.

□ Hàm $\text{Operator}(u)$ ghi lại toán tử áp dụng thành công tại u , tức là $\text{Operator}(u) = R$ nếu mọi đỉnh v kề u theo R đều giải được.

function $\text{Solvable}(u)$;

begin

1. **if** u là đỉnh kết thúc **then**

$\{\text{Solvable} \leftarrow \text{true}; \text{stop}\}$;

2. **if** u không là đỉnh kết thúc và không có đỉnh kề **then**

$\{\text{Solvable}(u) \leftarrow \text{false}; \text{stop}\}$;

3. **for** mỗi toán tử R áp dụng được tại u **do**

$\{Ok \leftarrow \text{true};$

for mỗi v kề u theo R **do**

if $\text{Solvable}(v) = \text{false}$ **then** $\{Ok \leftarrow \text{false}; \text{exit}\}$;

if Ok **then**

$\{\text{Solvable}(u) \leftarrow \text{true}; \text{Operator}(u) \leftarrow R; \text{stop}\}$

4. $\text{Solvable}(u) \leftarrow \text{false}$;

end;

Nhận xét

□ Hoàn toàn tương tự như thuật toán tìm kiếm theo độ sâu trong không gian trạng thái (mục 1.3.2), thuật toán tìm kiếm theo độ sâu trên đồ thị và/hoặc sẽ xác định được bài toán ban đầu là giải được hay không giải được, nếu cây tìm kiếm không có nhánh vô hạn. Nếu cây tìm kiếm có nhánh vô hạn thì chưa chắc thuật toán đã dừng, vì có thể nó bị xa lầy khi đi xuống nhánh vô hạn. Trong trường hợp này ta nên sử dụng thuật toán tìm kiếm sâu lặp (mục 1.3.3).

Nếu bài toán ban đầu giải được, thì bằng cách sử dụng hàm Operator ta sẽ xây dựng được cây nghiệm.

Bài tập chương 1:

Bài 1: Xây dựng không gian trạng thái gồm 8 địa điểm trên địa bàn bạn đang ở. Xác lập các đường đi và áp dụng thuật toán tìm kiếm mù theo chiều sâu, theo chiều rộng vào không gian trạng thái trên.

Bài 2: Xây dựng không gian trạng thái cho trò chơi xếp hình với $n=2$, chỉ ra trạng thái đầu, trạng thái đích, thứ tự ưu tiên. Áp dụng thuật toán tìm kiếm mù theo chiều sâu, theo chiều rộng vào bài toán trên.

Bài 3: Trong bài toán tìm giá trị cho một đại lượng trong tam giác, xác định tập 8 toán tử, bài toán cần giải, các bài toán giải được. Xây dựng đồ thị và hoặc cho bài toán, kiểm tra xem bài toán cần giải có giải được không? Nếu giải được thì giải qua các toán tử nào? Vẽ cây nghiệm tương ứng.