

ĐẠI HỌC KINH DOANH VÀ CÔNG NGHỆ HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN



GIÁO TRÌNH

TRÍ TUỆ NHÂN TẠO

Chủ biên : TS. Hoàng Xuân Thảo

Biên soạn: TS. Hoàng Xuân Thảo

(Dùng cho chương trình đào tạo hệ đại học

Lưu hành nội bộ

HÀ NỘI - 2009

MỤC LỤC

Chương 1 - TỔNG QUAN VỀ TRÍ TUỆ NHÂN TẠO	2
1.1. Lịch sử hình thành và phát triển.....	2
1.2. Khái niệm về trí tuệ nhân tạo (TTNT)	3
1.3. Lĩnh vực liên quan đến trí tuệ nhân tạo	5
1.4. Một số lĩnh vực ứng dụng của trí tuệ nhân tạo	5
1.5. Những vấn đề cốt lõi của TTNT	9
Chương 2 - KHÔNG GIAN TRẠNG THÁI VÀ CÁC PHƯƠNG PHÁP TÌM KIẾM	10
2.1. Thuật giải heuristics	10
2.2. Khái niệm về không gian trạng thái	15
2.3. Biểu diễn không gian trạng thái	15
2.4. Các chiến lược tìm kiếm mù	16
2.5. Các chiến lược tìm kiếm Heuristics	21
Chương 3 - BIỂU DIỄN TRI THỨC VÀ SUY DIỄN	30
3.1. Tri thức và dữ liệu	30
3.2. Phân loại tri thức	31
3.3. Các phương pháp biểu diễn tri thức	32
3.4. Các kỹ thuật suy diễn.....	46
3.5. Lập trình Prolog	52
Chương 4 - GIỚI THIỆU VỀ MÁY HỌC	60
4.1. Thế nào là máy học?	60
4.2. Học theo cây quyết định	61
4.3. Các ứng dụng	65
TÀI LIỆU THAM KHẢO	69

LỜI NÓI ĐẦU

Trong thời đại công nghiệp 4.0 lĩnh vực nghiên cứu về trí tuệ nhân tạo rất quan trọng đối với sinh viên. Trí tuệ nhân tạo là nỗ lực tìm hiểu những yếu tố trí tuệ. Lý do khác để nghiên cứu lĩnh vực này là cách để ta tự tìm hiểu bản thân chúng ta, là để tạo ra các thực thể thông minh giúp ích cho chúng ta. Trí tuệ nhân tạo có nhiều sản phẩm quan trọng và đáng lưu ý, thậm chí ngay từ lúc sản phẩm mới được hình thành.

Giáo trình cũng được biên soạn dựa trên kinh nghiệm giảng dạy nhiều năm môn Trí tuệ nhân tạo của giảng viên. Mục tiêu của nó nhằm giúp các bạn sinh viên chuyên ngành có một tài liệu cô đọng dùng làm tài liệu học tập. Trong giáo trình này sinh viên được làm quen với một số kiến thức cơ bản nhất về các phương pháp tìm kiếm lời giải và các phương pháp xử lý tri thức.

Ngoài ra, giáo trình cũng giới thiệu một số chương trình cài đặt, nhằm giúp sinh viên có thể hiểu một cách tường tận các giải thuật, đồng thời tin tưởng rằng các giải thuật này có thể áp dụng thực tế và cài đặt được trên máy tính một cách dễ dàng.

Giáo trình gồm các chương sau:

- Chương 1: Tổng quan về trí tuệ nhân tạo.
- Chương 2: Không gian trạng thái và các phương pháp tìm kiếm.
- Chương 3: Biểu diễn tri thức và suy diễn.
- Chương 4: Giới thiệu về máy học.

Mặc dù đã hết sức cố gắng, tuy nhiên giáo trình cũng không tránh khỏi những thiếu sót. Chúng tôi rất mong được sự góp ý của các độc giả, đặc biệt đối với các đồng nghiệp và sinh viên để giáo trình ngày càng hoàn thiện.

Chương 1 - TỔNG QUAN VỀ TRÍ TUỆ NHÂN TẠO

1.1. Lịch sử hình thành và phát triển

Vào năm 1943, Warren McCulloch và Walter Pitts bắt đầu thực hiện nghiên cứu ba cơ sở lý thuyết cơ bản: triết học cơ bản và chức năng của các nơ-ron thần kinh; phân tích các mệnh đề logic, và lý thuyết dự đoán của Turing. Các tác giả đã nghiên cứu đề xuất mô hình nơ-ron nhân tạo, mỗi nơ-ron đặc trưng bởi hai trạng thái "bật", "tắt" và phát hiện mạng nơ-ron có khả năng học.

Thuật ngữ "Trí tuệ nhân tạo" (Artificial Intelligence - AI) được thiết lập bởi John McCarthy tại Hội thảo đầu tiên về chủ đề này vào mùa hè năm 1956. Đồng thời, ông cũng đề xuất ngôn ngữ lập trình Lisp – một trong những ngôn ngữ lập trình hàm tiêu biểu, được sử dụng trong lĩnh vực AI. Sau đó, Alan Turing đưa ra "Turing test" như là một phương pháp kiểm chứng hành vi thông minh.

Thập kỷ 60, 70 Joel Moses viết chương trình Macsyma – chương trình toán học sử dụng cơ sở tri thức đầu tiên thành công. Marvin Minsky và Seymour Papert đưa ra các chứng minh đầu tiên về giới hạn của các mạng nơ-ron đơn giản. Ngôn ngữ lập trình logic Prolog ra đời và được phát triển bởi Alain Colmerauer. Ted Shortliffe xây dựng thành công một số hệ chuyên gia đầu tiên trợ giúp chẩn đoán trong y học, các hệ thống này sử dụng ngôn ngữ luật để biểu diễn tri thức và suy diễn.

Vào đầu những năm 1980, những nghiên cứu thành công liên quan đến AI như các hệ chuyên gia (expert systems) – một dạng của chương AI mô phỏng tri thức và các kỹ năng phân tích của một hoặc nhiều chuyên gia con người.

Vào những năm 1990 và đầu thế kỷ 21, AI đã đạt được những thành tựu to lớn nhất, AI được áp dụng trong logic, khai phá dữ liệu, chẩn đoán y học và nhiều lĩnh vực ứng dụng khác trong công nghiệp.

Những đặc trưng của Trí tuệ nhân tạo

- Trí tuệ nhân tạo xử lý thông tin theo trật tự ký hiệu. Các thông tin gồm: khái niệm, luật, các đối tượng? dùng cho suy lý. Khái niệm cơ bản trong Trí tuệ nhân tạo là sự thể hiện, suy lý, nhận biết, việc học và hệ thống cơ sở tri thức.
- Phương pháp may rủi hay được dùng trong Trí tuệ nhân tạo. Phương pháp này cho phép giải hai lớp bài toán khó. Thứ nhất là những bài toán chưa có thuật giải (bài toán nhận biết, ra quyết định). Thứ hai là các bài toán đã có thuật giải nhưng độ phức tạp lớn (chẳng hạn bài toán chơi cờ).
- Trí tuệ nhân tạo xét đến những thông tin không đầy đủ, không chính xác, có vẻ mâu thuẫn. Tuy vậy, các kết quả của Trí tuệ nhân tạo là cụ thể.
- Việc tương tác người- máy đi đôi với nhận biết tự động là cần thiết trong Trí tuệ nhân tạo. Các bài toán nhận dạng là ví dụ về yêu cầu này.
- Trí tuệ nhân tạo liên quan đến nhiều lĩnh vực, như các kỹ thuật mới, logic học, khoa học nhận biết, ngôn ngữ học, khoa học về tổ chức, thần kinh học. Trí tuệ nhân tạo còn nằm trong các lĩnh vực nghiên cứu nâng cao, các đề án nghiên cứu quan trọng.

1.2. Khái niệm về trí tuệ nhân tạo (TTNT)

Trí tuệ con người (Human Intelligence): Cho đến nay có hai khái niệm về trí tuệ con người được chấp nhận và sử dụng nhiều nhất, đó là:

- **Khái niệm trí tuệ theo quan điểm của Turing**

“Trí tuệ là những gì có thể đánh giá được thông qua các trắc nghiệm thông minh”

- Khái niệm trí tuệ đưa ra trong tự điển bách khoa toàn thư:

“Trí tuệ là khả năng:

Phản ứng một cách thích hợp những tình huống mới thông qua hiệu chỉnh hành vi một cách thích đáng.

Hiểu rõ những mối liên hệ qua lại của các sự kiện của thế giới bên ngoài nhằm đưa ra những hành động phù hợp đạt tới một mục đích nào đó.

Những nghiên cứu các chuyên gia tâm lý học nhận thức chỉ ra rằng quá trình hoạt động trí tuệ của con người bao gồm 4 thao tác cơ bản:

- 1- Xác định tập đích (goals).
- 2- Thu thập các sự kiện (facts) và các luật suy diễn (inference rules) để đạt được đích đặt ra.
- 3- Thu gọn (pruning) quá trình suy luận nhằm xác định tập các suy diễn có thể sử dụng được.
- 4- Áp dụng các cơ chế suy diễn cụ thể (inference mechanisms) để đưa các sự kiện ban đầu đi đến đích.

- **Trí tuệ nhân tạo**

Định nghĩa về AI đã có tới tám cuốn sách đề cập. Những định nghĩa đó đưa ra trên hai nhận định chính:

- Thứ nhất: quan tâm chủ yếu đến quá trình tư duy và lập luận
- Thứ hai: vấn đề ít được quan tâm hơn, đó là hoạt động.

Khái niệm trí tuệ nhân tạo (Artificial Intelligence- viết tắt là AI) được hiểu tùy theo cách nhìn từng người, chưa có định nghĩa nào được thừa nhận chung. Hiểu một cách đơn giản, trí tuệ nhân tạo là một lĩnh vực của khoa học và công nghệ nhằm làm cho máy có những khả năng trí tuệ của con người như: biết suy nghĩ và lập luận để giải quyết vấn đề, biết giao tiếp do hiểu ngôn ngữ và tiếng nói, biết học và tự thích nghi, ...

Một số định nghĩa khác nhau về trí tuệ nhân tạo:

- **Bellman (1978)** định nghĩa: trí tuệ nhân tạo là tự động hoá các hoạt động phù hợp với suy nghĩ con người, chẳng hạn các hoạt động ra quyết định, giải bài toán..
- **Rich và Knight (1991)** cho rằng trí tuệ nhân tạo là lĩnh vực nghiên cứu để làm cho máy tính làm được những việc mà con người đang làm tốt hơn.

- **Winston (1992)** cho rằng trí tuệ nhân tạo là lĩnh vực nghiên cứu các tính toán để máy có thể nhận thức, lập luận và tác động.
- **M. Minsky:** Nghiên cứu nhóm các phương pháp máy tính về hình vi thưng minh của con người

Những định nghĩa về AI được chia thành 4 nhóm:

- Hệ thống tư duy như con người
- Hệ thống tư duy có lập luận
- Hệ thống hoạt động như con người
- Hệ thống hoạt động có lập luận

1.3. Lĩnh vực liên quan đến trí tuệ nhân tạo

- Tâm lý học nhận thức.
- Thần kinh học.
- Lý thuyết về hệ thống (cybernetics).
- Toán Logic và Logic học.
- Sinh học tiến hoá.
- Khoa học về hành vi bầy đàn.
- Tổ chức học.
- Thống kê học.

1.4. Một số lĩnh vực ứng dụng của trí tuệ nhân tạo

- Các phương pháp tìm kiếm lời giải
- Hệ chuyên gia
- Xử lý ngôn ngữ tự nhiên
- Lý thuyết nhận dạng
- Lập kế hoạch và Người máy (Robot)
- Máy học
- Các mô hình thần kinh (Mạng Neuron và giải thuật di truyền)
- Một số ứng dụng cụ thể:

Tổng hợp tiếng nói: Từ một văn bản tự động tổng hợp thành tiếng nói. Thay vì phải tự đọc một cuốn sách hay nội dung một trang web, nó tự động đọc cho chúng ta. Giống như nhận dạng tiếng nói, tổng hợp tiếng nói là sự trợ giúp tốt cho người khiếm thị, nhưng ngược lại nó là bước cuối cùng trong giao tiếp giữa robot với người.

Nhận dạng tiếng nói: Nhận dạng tiếng nói đóng vai trò quan trọng trong giao tiếp giữa người và máy. Nó giúp máy móc hiểu và thực hiện các hiệu lệnh của con người. Một ứng dụng trong lĩnh vực này là hãng sản xuất xe hơi BMW (Đức) đang tiến hành phát triển một công nghệ mới cho phép các tài xế có thể soạn email, tin nhắn bằng giọng nói trong khi đang lái xe. Một ứng dụng khác là phần mềm lồng phụ đề vào các chương trình truyền hình. Đây là một công việc khá buồn tẻ và đòi hỏi phải có những người ghi tốc ký chuyên nghiệp. Nhờ có những tiến bộ trong công nghệ nhận dạng tiếng nói, các nhà cung cấp dịch vụ truyền hình gần đây đã gia tăng đáng kể số lượng các chương trình được lồng phụ đề của họ.

Nhận dạng chữ viết: Nhận dạng chữ viết ứng dụng trong lĩnh vực nhận dạng chữ in hoặc chữ viết tay và lưu thành văn bản điện tử. Ở Việt Nam, phần mềm VnDOCR do Phòng Nhận dạng & Công nghệ tri thức, Viện Công nghệ Thông tin xây dựng có thể nhận dạng trực tiếp tài liệu bằng cách quét thông qua máy scanner thành các tệp ảnh, chuyển đổi thành các tệp có định dạng *.doc, *.xls, *.txt, *.rtf, giúp người sử dụng không phải gõ lại tài liệu vào máy. Tương tự với phần mềm nhận dạng chữ viết trong thư viện, người ta cũng có thể dễ dàng chuyển hàng ngàn đầu sách thành văn bản điện tử một cách nhanh chóng.

Tóm tắt văn bản: Từ một văn bản dài tóm tắt thành một văn bản ngắn hơn theo mong muốn nhưng vẫn chứa những nội dung thiết yếu nhất.

Dịch tự động: Dịch tự động là công việc thực hiện dịch một ngôn ngữ sang một hoặc nhiều ngôn ngữ khác, không có sự can thiệp của con người trong quá trình dịch. Tuy nhiên, để làm cho máy hiểu được ngôn ngữ là một trong những vấn đề khó nhất của trí tuệ nhân tạo. Thí dụ câu: “ông già đi nhanh quá” cũng có nhiều cách

hiểu khác nhau: với cách phân tách từ và cụm từ thành ông già/đi/nhanh quá và ông/già đi/nhanh quá... thì việc dịch câu kiểu như thế này từ tiếng Việt sang tiếng Anh đòi hỏi máy không những phải hiểu đúng nghĩa câu tiếng Việt mà còn phải tạo ra được câu tiếng Anh tương ứng. Các phần mềm dịch tự động hiện nay còn phải tiếp tục nghiên cứu nhiều hơn nữa để có được những hệ dịch tốt.

Tìm kiếm thông tin: Thông tin trên mạng hàng ngày được gia tăng theo cấp số nhân. Việc tìm kiếm thông tin mà người dùng quan tâm bây giờ là tìm đúng thông tin mình cần và phải đáng tin cậy. Theo thống kê, có đến hơn 90% số lượng người Việt Nam lên mạng internet để thực hiện việc tìm kiếm thông tin. Các máy tìm kiếm (search engine) hiện nay chủ yếu thực hiện tìm kiếm dựa theo từ khóa. Thí dụ, Google hay Yahoo chỉ phân tích nội dung một cách đơn giản dựa trên tần suất của từ khoá, thứ hạng của trang và một số tiêu chí đánh giá khác. Kết quả là rất nhiều tìm kiếm không nhận được câu trả lời phù hợp, thậm chí bị dẫn tới một liên kết không liên quan gì do thủ thuật đánh lừa nhằm giới thiệu sản phẩm hoặc lại nhận được quá nhiều tài liệu không phải thứ ta mong muốn, trong khi đó lại không tìm ra tài liệu cần tìm.

Hiện nay, các nhà nghiên cứu đang cải tiến các công cụ tìm kiếm trực tuyến để một ngày nào đó, nó có thể hiểu và trả lời cả những câu hỏi cụ thể, thí dụ như “giá tour du lịch rẻ nhất từ Hà Nội đi Đà Lạt trong ba ngày của tháng này là bao nhiêu?”...Tuy vậy, thực tế cho đến bây giờ chưa có máy tìm kiếm nào có thể làm hài lòng người dùng kiểu như vậy.

Khai phá dữ liệu và phát hiện tri thức: Đây là lĩnh vực cho phép xử lý từ rất nhiều dữ liệu khác nhau để phát hiện ra tri thức mới. Ngoài ra, ứng dụng trong lĩnh vực này cũng cần phải biết trả lời câu hỏi của người sử dụng chúng từ việc tổng hợp dữ liệu thay vì máy móc chỉ đáp trả những gì có sẵn trong bộ nhớ. Thực tế để làm được điều này rất khó, nó gần như là mô phỏng quá trình học tập, khám phá khoa học của con người. Ngoài ra, dữ liệu thường có số lượng rất lớn, với nhiều kiểu (số, văn bản, hình ảnh, âm thanh, video,...) và không ngừng thay đổi. Để tìm ra tri

thức thì các chương trình phải đối mặt với vấn đề độ phức tạp tính toán,... Đây là lĩnh vực vẫn còn đang trong giai đoạn đầu phát triển.

Lái xe tự động: Theo Sebastian Thrun, Giáo sư ngành máy tính và kỹ thuật điện của Đại học Carnegie Mellon: ưu điểm lớn nhất của xe tự lái là khả năng loại bỏ sai sót của con người - nguyên nhân dẫn đến 95% số vụ tử vong mỗi năm tại Mỹ do tai nạn giao thông. “Chúng tôi có thể giảm bớt 50% số vụ tai nạn do nguyên nhân này,” ông Sebastian Thrun khẳng định.

Chế tạo được ô tô tự lái và an toàn cao cũng là một mục tiêu được Cục nghiên cứu các dự án công nghệ cao Bộ quốc phòng Mỹ DARPA (Defense Advanced Research Projects Agency) khởi xướng và hỗ trợ dưới dạng một cuộc thi mang tên “thách thức lớn của DARPA” (DARPA grand challenge).

Chúng ta hy vọng sẽ đến một ngày, những chiếc ô tô chạy trên đường không cần người lái. Chỉ nói nơi muốn đến, xe sẽ đưa ta đi và đi an toàn.

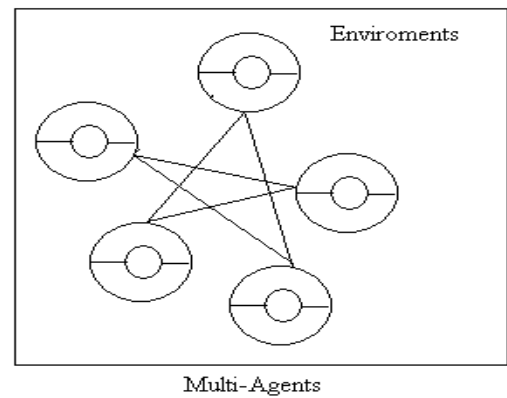
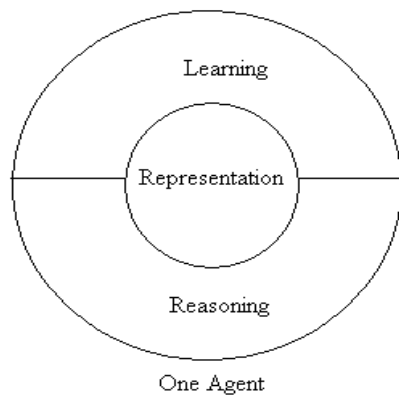
Robot: Nhiều đề án nghiên cứu về robot thông minh và các lĩnh vực liên quan được ứng dụng trong đời sống. Các đề án này hướng đến các sáng tạo công nghệ có nhiều ý nghĩa trong văn hóa, xã hội và công nghiệp, đòi hỏi phải tích hợp nhiều công nghệ, như nguyên lý các tác tử, biểu diễn tri thức về không gian, nhận biết chiến lược, lập luận thời gian thực, nhận dạng và xử lý các chuỗi hình ảnh liên tục trong thời gian thực, ... Một trong những ứng dụng đó là đề án RoboCup: tổ chức thi đấu bóng đá giữa các đội robot. Mục tiêu hướng đến của đề án này là đến năm 2050, sẽ chế tạo được một đội robot có thể thắng đội bóng đá vô địch thế giới.

Ứng dụng quan trọng khác của lĩnh vực này là chế tạo robot đôi phó và dò tìm nạn nhân trong các thảm họa. Trong sự cố hư hỏng tại nhà máy điện hạt nhân xảy ra sau trận động đất và sóng thần ngày 11 tháng 3 ở Nhật vừa qua, người ta gửi robot có tên Quince để hoạt động tại những khu vực khó tiếp cận do độ phóng xạ cao của nhà máy Fukushima. Được điều khiển từ xa, Quince có thể làm việc trong nhiều giờ

đồng hồ để chụp hình và đo độ phóng xạ trong những tòa nhà bị lây nhiễm chất phóng xạ, nơi mà các kỹ thuật viên không thể vào bên trong.

1.5. Những vấn đề cốt lõi của TTNT

- Biểu diễn (*representation*).
- Lập luận (*reasoning*).
- Học (*learning*).
- Tương tác (*interaction*).



Chương 2

KHÔNG GIAN TRẠNG THÁI VÀ CÁC PHƯƠNG PHÁP TÌM KIẾM

2.1. Thuật giải heuristics

2.1.1. Khái niệm

Thuật giải Heuristic thể hiện cách giải bài toán với các đặc tính sau: Giải bài toán theo thuật giải Heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu, vì vậy chi phí thấp hơn. Thường tìm được lời giải tốt (nhưng không chắc là lời giải tốt nhất). Thuật giải Heuristic gần gũi với cách suy nghĩ và hành động của con người.

Một số nguyên lý như sau: để xây dựng thuật giải Heuristic,

• **Nguyên lý vết cận thông minh:** Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách giới hạn lại không gian tìm kiếm hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu.

• **Nguyên lý tham lam (Greedy):** Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải.

• **Nguyên lý thứ tự:** Thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.

• **Hàm Heuristic:** Trong việc xây dựng các thuật giải Heuristic, người ta thường dùng các hàm Heuristic. Đó là các hàm đánh giá thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải. Nhờ giá trị này, ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của thuật giải.

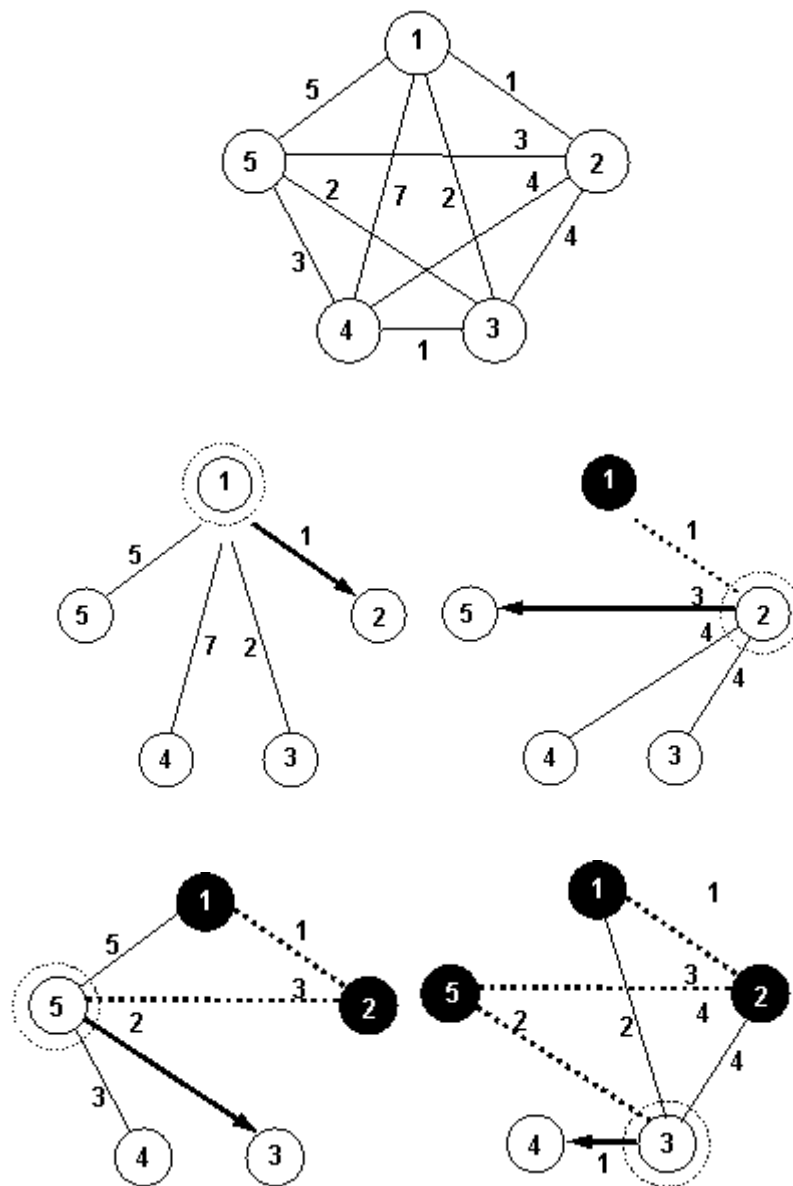
2.1.2. Bài toán hành trình ngắn nhất - ứng dụng nguyên lý tham lam (Greedy)

Bài toán: Có một người đi đưa thư hàng ngày phải qua nhiều địa điểm. Hãy tìm một hành trình cho một người đưa thư đi qua n điểm khác nhau, mỗi điểm đi qua một lần và trở về điểm xuất phát sao cho tổng chiều dài đoạn đường cần đi là ngắn nhất. Giả sử rằng có con đường nối trực tiếp từ giữa hai điểm bất kỳ.

Để giải bài toán trên có nhiều cách giải, có thể giải bằng phương pháp liệt kê tất cả con đường có thể đi, tính chiều dài của mỗi con đường đó rồi tìm con đường có chiều dài ngắn nhất, tuy nhiên giải bằng cách này lời giải khá lâu và mất nhiều thời gian để tìm ra lời giải. Lời giải đơn giản hơn nhiều và thường cho kết quả tương đối tốt là dùng một thuật giải Heuristic ứng dụng nguyên lý Greedy. Tư tưởng của thuật giải như sau:

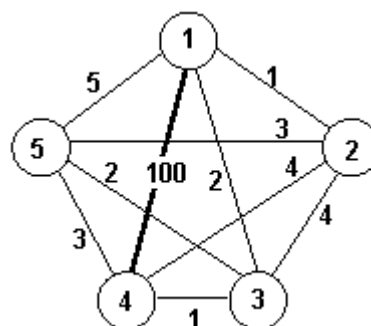
- Từ điểm xuất phát, ta liệt kê tất cả quãng đường từ điểm xuất phát cho đến n đại lý rồi chọn đi theo con đường ngắn nhất.
- Khi đã đi đến một điểm kế tiếp, chọn đi đến điểm kế tiếp cũng theo nguyên tắc trên. Nghĩa là liệt kê tất cả con đường từ điểm đang đứng đến những điểm chưa đi đến. Chọn con đường ngắn nhất. Lặp lại quá trình này cho đến lúc không còn điểm nào để đi.

Theo nguyên lý Greedy, ta lấy tiêu chuẩn hành trình ngắn nhất của bài toán làm tiêu chuẩn cho chọn lựa cục bộ. *Ta hy vọng rằng, khi đi trên n đoạn đường ngắn nhất thì cuối cùng ta sẽ có một hành trình ngắn nhất.* Điều này không phải lúc nào cũng đúng. Với điều kiện trong hình tiếp theo thì thuật giải cho chúng ta một hành trình có chiều dài là 14 trong khi hành trình tối ưu là 13. Kết quả của thuật giải Heuristic trong trường hợp này chỉ lệch 1 đơn vị so với kết quả tối ưu. Trong khi đó, độ phức tạp của thuật giải Heuristic này chỉ là $O(n^2)$.



Hình : Giải bài toán sử dụng nguyên lý Greedy

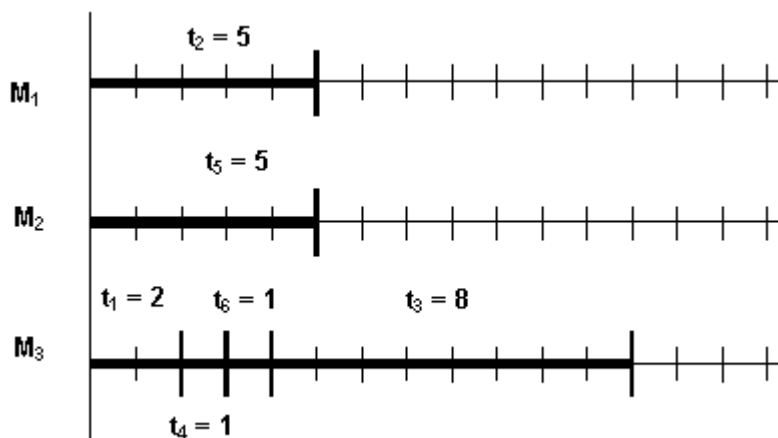
Tất nhiên, thuật giải theo kiểu Heuristic đôi lúc lại đưa ra kết quả không tốt, thậm chí rất tệ như trường hợp ở hình sau.



2.1.3. Bài toán phân việc - ứng dụng của nguyên lý thứ tự

Một công ty nhận được hợp đồng gia công m chi tiết máy J_1, J_2, \dots, J_m . Công ty có n máy gia công lần lượt là P_1, P_2, \dots, P_n . Mọi chi tiết đều có thể được gia công trên bất kỳ máy nào. Một khi đã gia công một chi tiết trên một máy, công việc sẽ tiếp tục cho đến lúc hoàn thành, không thể bị cắt ngang. Để gia công một việc J_1 trên một máy bất kỳ ta cần dùng một thời gian tương ứng là t_1 . Nhiệm vụ của công ty là phải làm sao gia công xong toàn bộ n chi tiết trong thời gian sớm nhất.

Chúng ta xét bài toán trong trường hợp có 3 máy P_1, P_2, P_3 và 6 công việc với thời gian là $t_1=2, t_2=5, t_3=8, t_4=1, t_5=5, t_6=1$. ta có một phương án phân công (L) như hình sau:

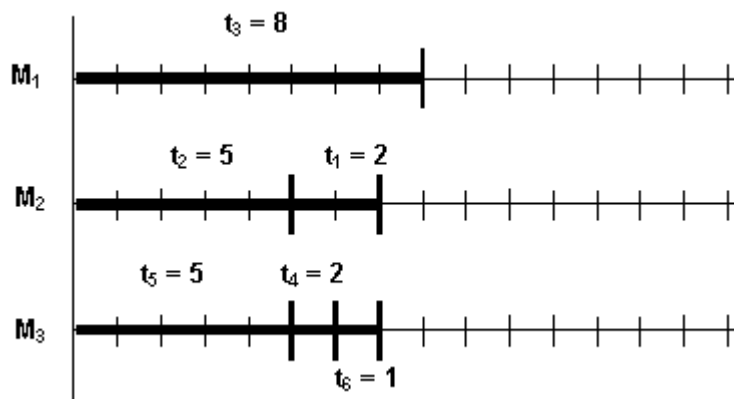


Theo hình này, tại thời điểm $t=0$, ta tiến hành gia công chi tiết J_2 trên máy P_1 , J_5 trên P_2 và J_1 tại P_3 . Tại thời điểm $t=2$, công việc J_1 được hoàn thành, trên máy P_3 ta gia công tiếp chi tiết J_4 . Trong lúc đó, hai máy P_1 và P_2 vẫn đang thực hiện công việc đầu tiên mình ... Sơ đồ phân việc theo hình ở trên được gọi là lược đồ GANTT. Theo lược đồ này, ta thấy thời gian để hoàn thành toàn bộ 6 công việc là 12. Nhận xét một cách cảm tính ta thấy rằng phương án (L) vừa thực hiện là một phương án không tốt. Các máy P_1 và P_2 có quá nhiều thời gian rảnh.

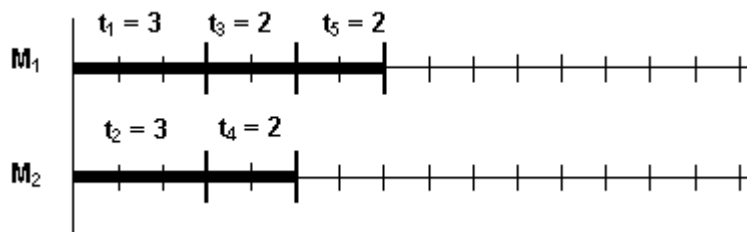
Thuật toán tìm phương án tối ưu L_0 cho bài toán này theo kiểu vét cạn có độ phức tạp cỡ $O(mn)$ (với m là số máy và n là số công việc). Bây giờ ta xét đến một thuật giải Heuristic rất đơn giản (độ phức tạp $O(n)$) để giải bài toán này.

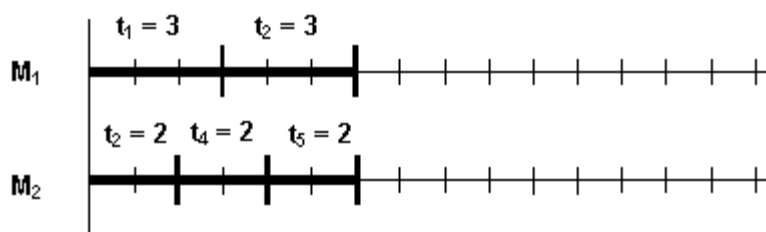
- Sắp xếp các công việc theo thứ tự giảm dần về thời gian gia công.
- Lần lượt sắp xếp các việc theo thứ tự đó vào máy còn dư nhiều thời gian nhất.

Với tư tưởng như vậy, ta sẽ có một phương án L^* như sau:



Rõ ràng phương án L^* vừa thực hiện cũng chính là phương án tối ưu của trường hợp này vì thời gian hoàn thành là 8, đúng bằng thời gian của công việc J_3 . Ta hy vọng rằng một giải Heuristic đơn giản như vậy sẽ là một thuật giải tối ưu. Nhưng tiếc thay, ta dễ dàng đưa ra được một trường hợp mà thuật giải Heuristic không đưa ra được kết quả tối ưu.





2.2. Khái niệm về không gian trạng thái

Không gian trạng thái là tập tất cả các trạng thái có thể có và tập các toán tử của bài toán.

Không gian trạng thái là một bộ bốn, Ký hiệu: $K = (T, S, G, F)$. Trong đó:

T: tập tất cả các trạng thái có thể có của bài toán

S: trạng thái đầu

G: tập các trạng thái đích

F: tập các toán tử

2.3. Biểu diễn không gian trạng thái

Giải bài toán trong không gian trạng thái, trước hết phải xác định dạng mô tả trạng thái bài toán sao cho bài toán trở nên đơn giản hơn, phù hợp bản chất vật lý của bài toán (Có thể sử dụng các xâu ký hiệu, vectơ, mảng hai chiều, cây, danh sách).

Mỗi trạng thái chính là mỗi hình trạng của bài toán, các tình trạng ban đầu và tình trạng cuối của bài toán gọi là trạng thái đầu và trạng thái cuối.

Ví dụ 1. Không gian trạng thái của bài toán đong nước là bộ bốn T, S, G, F xác định như sau:

$$T = \{ (x,y) / 0 \leq x \leq m; 0 \leq y \leq n \}$$

$$S = (0,0)$$

$$G = \{ (x,k) \text{ hoặc } (k,y) / 0 \leq x \leq m; 0 \leq y \leq n \}$$

F = Tập các thao tác đong đầy, đổ ra hoặc đổ sang bình khác thực hiện trên một bình.

Ví dụ 2. Không gian trạng thái của bài toán Tháp Hà nội với $n = 3$:

$$T = \{ (x_1, x_2, x_3) / x_i \in \{1, 2, 3\} \}$$

$$S = (1, 1, 1)$$

$$G = \{(3, 3, 3)\}$$

F = Tập các khả năng có thể chuyển đĩa đã xác định trong phần trước.

Ví dụ 3. Không gian trạng thái của bài toán trò chơi 8 số:

$$T = \{ (a_{ij})_{3 \times 3} / 0 \leq a_{ij} \leq 8 \text{ và } a_{ij} \neq a_{kl} \text{ với } i \neq j \text{ hoặc } k \neq l \}$$

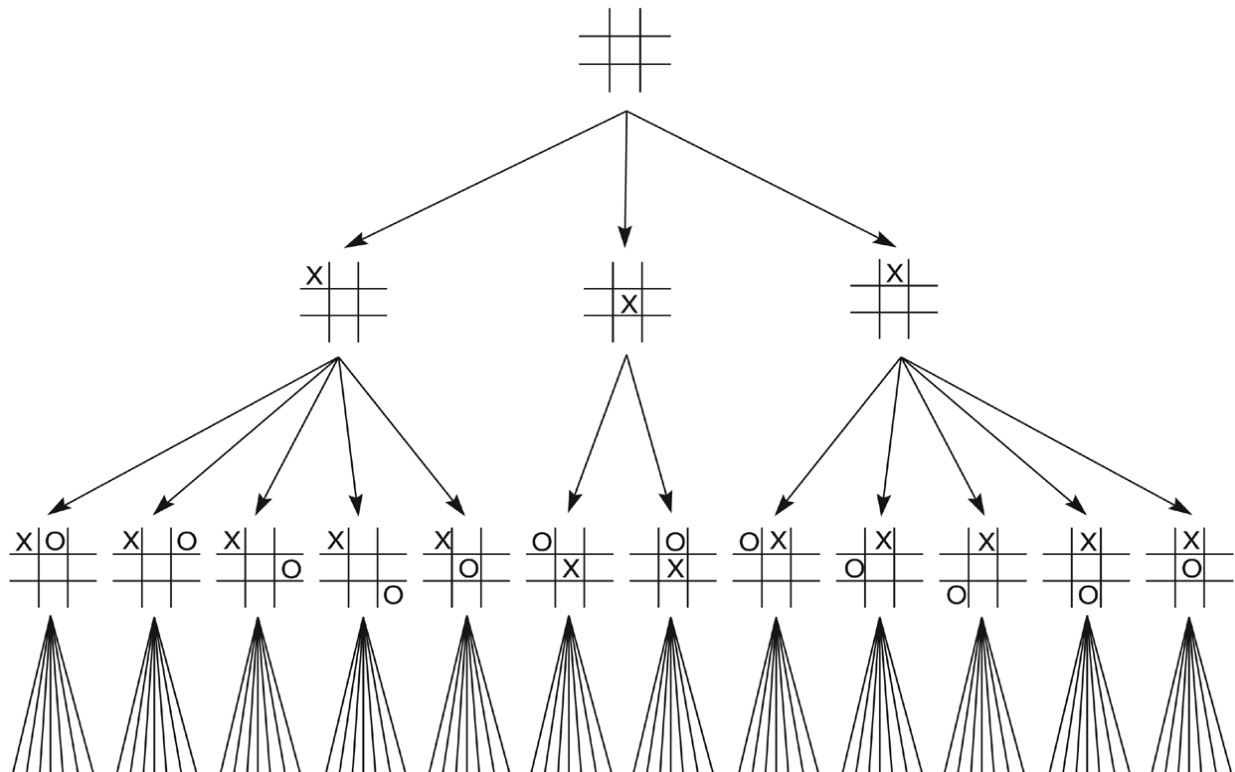
S = Ma trận xuất phát của bài toán,

G = Ma trận cuối cùng của bài toán (các số nằm theo vị trí yêu cầu)

$$F = \{f_l, f_r, f_u, f_d\}$$

Tìm kiếm lời giải trong không gian trạng thái là quá trình tìm kiếm xuất phát từ trạng thái ban đầu, dựa vào toán tử chuyển trạng thái để xác định các trạng thái tiếp theo cho đến khi gặp được trạng thái đích.

Ví dụ 4. Không gian trạng thái triển khai trong Tic-tac-toe



2.4. Các chiến lược tìm kiếm mù

2.4.1. Tìm kiếm theo chiều rộng (BFS): *Breadth-first search*

a. Tư tưởng

Xuất từ một đỉnh v bất kỳ của đồ thị G , chúng ta thực hiện như sau:

Bước 1: đánh dấu đã duyệt cho một đỉnh v bất kỳ.

Bước 2: chọn đỉnh v đã được duyệt nhưng có đỉnh kề chưa được duyệt. Việc chọn đỉnh v được xét ưu tiên cho các đỉnh được đánh dấu duyệt sớm.

Bước 3: thực hiện đánh dấu đã duyệt với tất cả các đỉnh w kề với v ,

Bước 4: làm lại bước 2 cho đến khi tất cả các đỉnh được duyệt.

b. Thuật toán

procedure BFS(v)

(* tìm kiếm theo chiều rộng bắt đầu từ đỉnh v , các biến Chuaxet, Ke là biến cục bộ*)

Begin

QUEUE: = \emptyset ;

QUEUE $\leftarrow v$; (* ket qua nap vao queue*)

Chuaxet[v] := false ;

While QUEUE $\neq \emptyset$ do

Begin

$p \leftarrow \text{QUEUE}$;; (* lay phan tu QUEUE :*)

Tham_dinh(p) ;

For $u \in \text{ke}(v)$ do

If Chuaxet[u] then

Begin

QUEUE $\leftarrow u$;

Chuaxet[u] := false;

End;

End;

End;

Khi đó, tìm kiếm theo chiều rộng trên đồ thị được thực hiện nhờ thuật toán sau:

Begin

(* Initialization*)

For $f \in V$ do Chuaxet[v] :=true ;

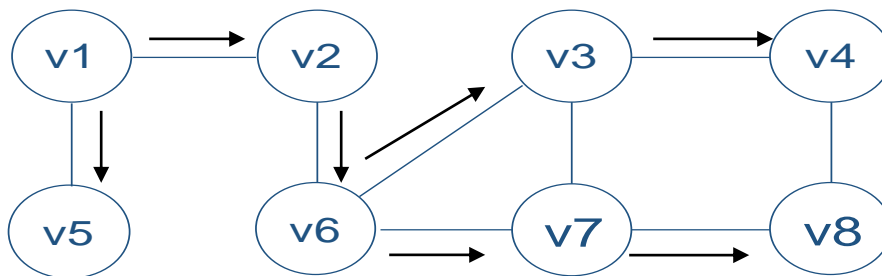
For $v \in V$ do

If Chuaxet[v] then BFS(v) ;

End;

c. Ví dụ 1

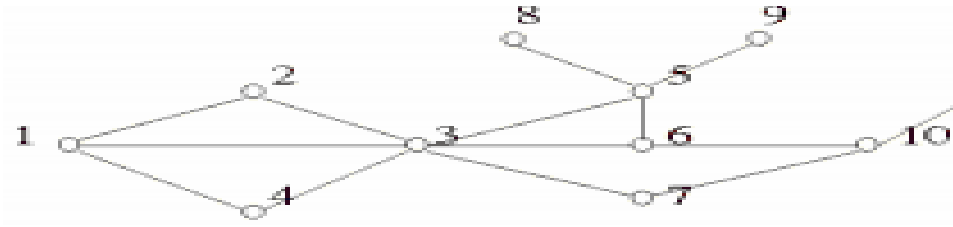
Thực hiện duyệt đồ thị theo chiều rộng trên đồ thị G dưới đây:



Bảng duyệt

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
v1	v2	v5	v6	v3	v7	v4	v8

Ví dụ 2



Bảng duyệt

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	2	4	3	5	6	7	8	9	10

2.4.2. Tìm kiếm theo chiều sâu (DFS): Depth – First Search

a. Tư tưởng

Xuất từ một đỉnh v bất kỳ của đồ thị G , chúng ta thực hiện như sau:

Bước 1: đánh dấu v đã được duyệt.

Bước 2: thực hiện đánh dấu đã duyệt với mỗi đỉnh w chưa duyệt kề với v ,

Bước 3: làm lại bước 2 cho đến khi tất cả các đỉnh được duyệt.

b. Thuật toán

procedure DFS(v)

(* tìm kiếm theo chiều sâu bắt đầu từ đỉnh v , các biến Chuaxet, Ke là biến toàn cục*)

Begin

Tham_dinh(v);

Chuaxet[v]:=false;

For $u \in \text{Ke}(v)$ do

If Chuaxet[u] then DFS(u) ;

End;(*đỉnh v đã duyệt xong*)

Khi đó, tìm kiếm theo chiều sâu trên đồ thị được thực hiện nhờ thuật toán sau:

Begin

(* Initialization*)

For $v \in V$ do Chuaxet[v] :=true ;

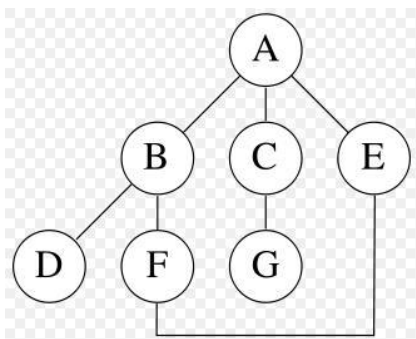
For $v \in V$ do

If Chuaxet[v] then DFS(v) ;

End;

c. Ví dụ

Thực hiện duyệt đồ thị theo chiều sâu trên đồ thị G dưới đây:



Tìm kiếm ưu tiên chiều sâu bắt đầu thăm đỉnh A, đi theo cạnh trái, tiếp tục tìm kiếm xong ở cây con trái mới chuyển sang tìm kiếm ở cây con phải. Thứ tự thăm viếng các đỉnh là: A, B, D, F, E, C, G.

Quá trình viếng thăm các đỉnh diễn ra như sau: Sau khi thăm đỉnh A, vì B chưa được thăm nên theo cạnh AB ta thăm B, tiếp tục theo cạnh BD tới viếng thăm

D. Từ D không thể tiếp tục đi xa hơn, ta quay lại B. Từ B, theo BF đến thăm F, từ F đến thăm E. Từ E vì A đã viếng thăm nên ta quay lại F, rồi quay lại B. Tại B vì tất cả các khả năng từ B đã xem xét nên ta quay lại A. Từ A, quá trình tiếp tục với các đỉnh C và G.

2.5. Các chiến lược tìm kiếm Heuristics

2.5.1. Phương pháp tìm kiếm leo đồi (Hill-climbing)

🚦 Leo đồi sẽ duyệt tất cả các hướng đi có thể và chọn đi theo trạng thái tốt nhất trong số các trạng thái kế tiếp có thể có. (Leo đồi chỉ chọn đi theo trạng thái kế tiếp đầu tiên tốt hơn trạng thái hiện hành mà nó có thể tìm thấy).

🚦 Tư tưởng:

Nếu trạng thái ban đầu cũng là trạng thái đích thì thoát và báo là đã tìm thấy lời giải. Ngược lại, đặt trạng thái hiện hành (T_i) và trạng thái ban đầu là (T_0)

Lặp lại cho đến khi đạt đến trạng thái kết thúc hoặc cho đến khi (T_i) không tồn tại trạng thái kế tiếp (T_k) nào đó tốt hơn trạng thái hiện tại (T_i).

- Đặt S bằng tập tất cả trạng thái kế tiếp có thể có của (T_i) và tốt hơn (T_i)
- Xác định T_k^{\max} là trạng thái tốt nhất trong tập S

🚦 Mã giả

$T_i := T_0;$

Stop: = false;

While stop = false do begin

 If $T_i \equiv T_G$ then begin

(* tìm được kết quả*)

Stop: = true;

End;

Else begin

Best : = h' (T_i);

$T_k^{\max} := T_i;$

While< tồn tại trạng thái kế tiếp hợp lệ của T_i > do begin

T_k : = < một trạng thái kế tiếp của T_i>;

If < h' (T_k) tốt hơn Best> then begin

Best : = h' (T_k);

T_{max} : = T_k;

End;

End;

If (Best > T_i)

T_i := T_{max};

Else begin

< không tìm thấy kết quả>;

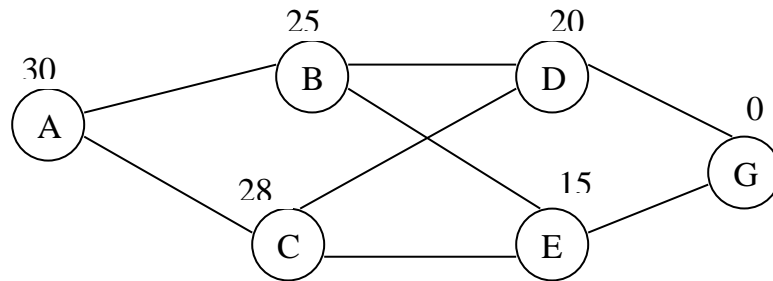
Stop: = true;

End;

End;{ Else if)

End;{ while stop}

🚦 Ví dụ



Giải

$T_0 = \{A\}; T_i = \{A\}$

$T_k\{A\} = \{B, C\}$

$\left. \begin{array}{l} h_B = 25 \\ h_C = 28 \end{array} \right\} \text{ thấy } h_B = 25 < h_A$

chọn $T_i = \{B\}$

$T_k\{B\} = \{D, E\}$

$\left. \begin{array}{l} h_D = 20 \\ h_E = 15 \end{array} \right\} \text{ thấy } h_E = 15 < h_B$

chọn $T_i = \{E\}$

$$T_k\{E\} = \{C, G\}$$

$$\left. \begin{array}{l} h_C = 20 \\ h_G = 0 \end{array} \right\} \begin{array}{l} \text{thấy } h_G = 0 < h_E \\ T_{\max} \equiv T_G \Rightarrow \text{finish} \end{array}$$

vậy ta có đường đi: A -> B -> E -> G

2.5.2. Phương pháp tìm kiếm tốt nhất (Best-first search).

Sử dụng hàm đánh giá để hướng dẫn việc tìm kiếm. 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.

Tìm kiếm tốt nhất đầu tiên khác với tìm kiếm theo chiều rộng ở chỗ:

+ Trong tìm kiếm theo chiều rộng ta lần lượt phát triển tất cả các nút ở mức hiện tại để sinh ra các nút ở mức tiếp theo.

+ Còn trong tìm kiếm tốt nhất đầu tiên ta chọn nút để phát triển là nút tốt nhất được xác định bởi hàm đánh giá (tức là nút có trọng số nhỏ (lớn) nhất sẽ được chọn), nút này có thể ở mức hiện tại hoặc ở các mức trên.

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.

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.

🔗Thuật giải BEST-FIRST SEARCH

1. Đặt **OPEN** chứa trạng thái khởi đầu.
2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện :
 - 2.a. Chọn trạng thái **tốt nhất** (T_{max}) trong OPEN (và xóa T_{max} khỏi OPEN)
 - 2.b. Nếu T_{max} là trạng thái kết thúc thì thoát.
 - 2.c. Ngược lại, tạo ra các trạng thái kế tiếp T_k có thể có từ trạng thái T_{max} . Đối với mỗi trạng thái kế tiếp T_k thực hiện:

Tính $f(T_k)$; Thêm T_k vào OPEN

BFS khá đơn giản. Tuy vậy, trên thực tế, cũng như tìm kiếm chiều sâu và chiều rộng, hiếm khi ta dùng BFS một cách trực tiếp. Thông thường, người ta thường dùng các phiên bản của BFS là AT, AKT và A^*

Thuật giải AT là một phương pháp tìm kiếm theo kiểu BFS với độ tốt của nút là giá trị hàm **g** – tổng chiều dài con đường đã đi từ trạng thái bắt đầu đến trạng thái hiện tại.

🔗Thuật giải AT

1. Đặt **OPEN** chứa trạng thái khởi đầu.
2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện :

2.a. Chọn trạng thái (Tmax) có **giá trị g nhỏ nhất** trong OPEN (và xóa Tmax khỏi OPEN)

2.b. Nếu Tmax là trạng thái kết thúc thì thoát.

2.c. Ngược lại, tạo ra các trạng thái kế tiếp Tk có thể có từ trạng thái Tmax. Đối với mỗi trạng thái kế tiếp Tk thực hiện :

$$g(Tk) = g(Tmax) + \text{cost}(Tmax, Tk);$$

Thêm Tk vào OPEN.

* Vì chỉ sử dụng hàm g (mà không dùng hàm ước lượng h') để đánh giá độ tốt của một trạng thái nên ta cũng có thể xem AT chỉ là một thuật toán.

Thuật giải AKT

(Algorithm for Knowledgeable Tree Search)

Thuật giải AKT mở rộng AT bằng cách sử dụng thêm thông tin ước lượng h'. Độ tốt của một trạng thái f là tổng của hai hàm g và h'.

🔗Thuật giải AKT

1. Đặt **OPEN** chứa trạng thái khởi đầu.

2. Cho đến khi tìm được trạng thái đích hoặc không còn nút nào trong OPEN, thực hiện :

2.a. Chọn trạng thái (Tmax) có **giá trị f nhỏ nhất** trong OPEN (và xóa Tmax khỏi OPEN)

2.b. Nếu Tmax là trạng thái kết thúc thì thoát.

2.c. Ngược lại, tạo ra các trạng thái kế tiếp Tk có thể có từ trạng thái Tmax. Đối với mỗi trạng thái kế tiếp Tk thực hiện :

$$g(Tk) = g(Tmax) + \text{cost}(Tmax, Tk);$$

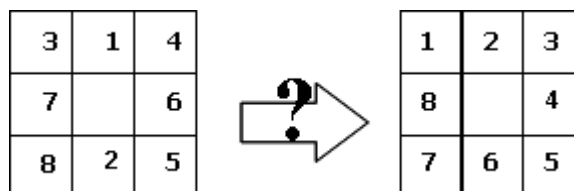
Tính $h'(Tk)$

$$f(Tk) = g(Tk) + h'(Tk);$$

Thêm Tk vào OPEN.

2.5.3. Ứng dụng A* để giải bài toán Tac-ci

Bài toán Ta-can-h đã từng là một trò chơi khá phổ biến, đôi lúc người ta còn gọi đây là bài toán 9-puzzle. Trò chơi bao gồm một hình vuông kích thước 3x3 ô. Có 8 ô có số, mỗi ô có một số từ 1 đến 8. Một ô còn trống. Mỗi lần di chuyển chỉ được di chuyển một ô nằm cạnh ô trống về phía ô trống. Vấn đề là từ một trạng thái ban đầu bất kỳ, làm sao đưa được về trạng thái cuối là trạng thái mà các ô được sắp lần lượt từ 1 đến 8 theo thứ tự từ trái sang phải, từ trên xuống dưới, ô cuối cùng là ô trống.



Cho đến nay, ngoại trừ 2 giải pháp vét cạn và tìm kiếm Heuristic, người ta vẫn chưa tìm được một thuật toán chính xác, tối ưu để giải bài toán này. Tuy nhiên, cách giải theo thuật giải A* lại khá đơn giản và thường tìm được lời giải (nhưng không phải lúc nào cũng tìm được lời giải). Nhận xét rằng: Tại mỗi thời điểm ta chỉ có tối đa 4 ô có thể di chuyển. Vấn đề là tại thời điểm đó, ta sẽ chọn lựa di chuyển ô nào? Chẳng hạn ở hình trên, ta nên di chuyển (1), (2), (6), hay (7) ? Bài toán này hoàn

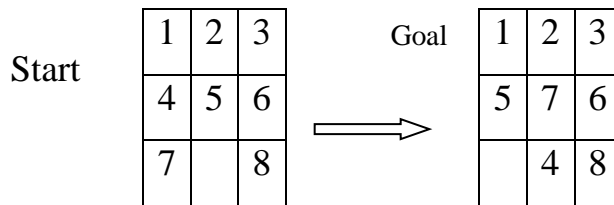
toàn có cấu trúc thích hợp để có thể giải bằng A* (tổng số trạng thái có thể có của bàn cờ là $n^2!$ với n là kích thước bàn cờ vì mỗi trạng thái là một hoán vị của tập n^2 con số).

Có hai cách xác định hàm heuristic:

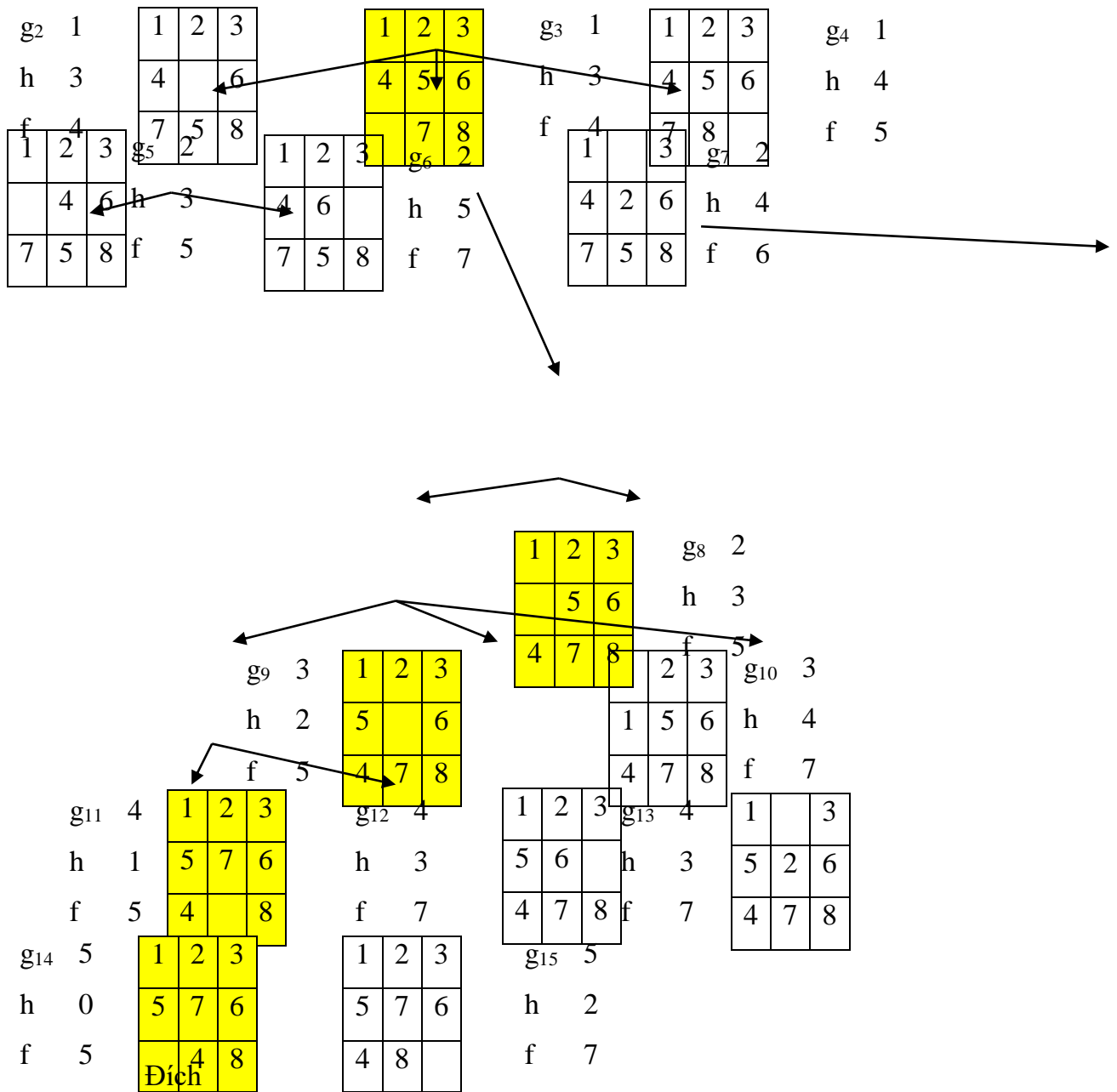
Cách 1: $h1$ = số lượng ô số mà ở sai vị trí. Đối với hình trên 7 trong số 8 quân cờ ở sai vị trí, vì vậy trạng thái đầu sẽ có $h1 = 7$. $h1$ là một hàm heuristic chấp nhận được, bởi vì rõ ràng là bất cứ ô số nào mà đang ở sai vị trí phải di chuyển ít nhất một lần.

Cách 2: $h2$ = tổng số khoảng cách của các ô số so với vị trí mục tiêu. Bởi vì các quân cờ không thể đi theo các đường chéo, khoảng cách mà chúng ta tính tổng sẽ là tổng của các khoảng cách theo chiều ngang và theo chiều dọc.

Ví dụ:



Cách 1:



KẾT LUẬN: Tiến trình qua 5 bước: g1->g3->g8->g9->g11->g15 từ trạng thái ban đầu đạt tới đích.

Chương 3 - BIỂU DIỄN TRI THỨC VÀ SUY DIỄN

3.1. Tri thức và dữ liệu

Thành phần trung tâm của agent *dựa trên tri thức* (knowledge-based agent), còn gọi là *hệ dựa trên tri thức* (knowledge-based system) hoặc đơn giản là hệ tri thức trong đó chứa cơ sở tri thức (Knowledge Base: viết tắt tiếng Anh: KB; viết tắt tiếng Việt: CSTT).

Cơ sở tri thức là một tập hợp các tri thức được biểu diễn dưới dạng nào đó. Mỗi khi nhận được các thông tin đưa vào, Agent cần có khả năng suy diễn để đưa ra các câu trả lời, đưa ra các hành động hợp lý. Nhiệm vụ này được thực hiện bởi bộ suy diễn-thành phần cơ bản khác của các hệ tri thức. Như vậy, hệ tri thức bao hàm một CSTT và được trang bị một thủ tục suy diễn. Mỗi khi tiếp nhận các sự kiện từ môi trường, thủ tục suy diễn thực hiện quá trình liên kết các sự kiện với các tri thức trong CSTT để rút ra các câu trả lời, hoặc các hành động hợp lý mà Agent cần thực hiện. Khi thiết kế một Agent giải quyết vấn đề nào đó thì CSTT sẽ chứa các tri thức về đối tượng cụ thể đó. Để máy tính có thể sử dụng, xử lý tri thức, cần biểu diễn tri thức dưới dạng thuận tiện. Đó là mục tiêu của biểu diễn tri thức.

Tri thức là một khái niệm trừu tượng. Chúng ta không cố gắng đưa ra một định nghĩa chính xác ở đây mà muốn so sánh nó với hai khái niệm có liên quan là thông

tin và dữ liệu. Karan Sing đã phát biểu: “Chúng ta ngập chìm trong thông biển thông tin nhưng lại khát tri thức”.

Trong ngữ cảnh của khoa học máy tính “dữ liệu là nguyên liệu thô để xử lý” là các con số, chữ cái, hình ảnh, âm thanh... Thông tin là tất cả những gì con người có thể cảm nhận qua các giác quan (chính xác, xem khái niệm Entropy là độ đo thông tin, độ đo về các tin tức mới đối với một người nào đó). Nếu so về số lượng: dữ liệu nhiều hơn thông tin; thông tin nhiều hơn tri thức. Chúng ta có thể mô tả chúng theo dạng hình chóp.

3.2. Phân loại tri thức

Tri thức sự kiện : là các khẳng định về một sự kiện, khái niệm nào đó (trong một phạm vi xác định). Các định luật vật lý, toán học, ... thường được xếp vào loại này. (Chẳng hạn: mặt trời mọc ở đằng đông, tam giác đều có 3 góc 60^0 , ...)

➡ **Tri thức thủ tục** : thường dùng để diễn tả phương pháp, các bước cần tiến hành, trình tự hay ngắn gọn là cách giải quyết một vấn đề. Thuật toán, thuật giải là một dạng của tri thức thủ tục.

➡ **Tri thức mô tả** : cho biết một đối tượng, sự kiện, vấn đề, khái niệm, ... được thấy, cảm nhận, cấu tạo như thế nào (một cái bàn thường có 4 chân, con người có 2 tay, 2 mắt,...)

➡ **Tri thức Heuristic** : là một dạng tri thức cảm tính. Các tri thức thuộc loại này thường có dạng ước lượng, phỏng đoán, và thường được hình thành thông qua kinh nghiệm.

Tri thức Meta: Diên t_q tri thức v_o tri thức. Lo¹i tri thức này giúp l^êy ra tri thức th¹ch h¹p nh¹t \otimes ó gi¹i quy¹t v¹n \otimes o

3.3. Các phương pháp biểu diễn tri thức

3.3.1. Biểu diễn tri thức bằng luật sinh

Cho hai bình rỗng X và Y có thể tích lần lượt là VX và VY, hãy dùng hai bình này để đo ra z lít nước ($z \leq \min(VX, VY)$). Vấn đề này được giải quyết bằng cách sử dụng các luật dẫn xuất (luật sinh).

Ví dụ: Xét một trường hợp cụ thể, có 2 bình rỗng X và Y, trong đó bình X có thể tích là 7 lít, bình Y có thể tích là 9 lít nước. Trong trường hợp này có thể ký hiệu cụ thể như $VX = 7$ và $VY = 9$ và $z = 4$. Sau một thời gian tính toán, có thể sẽ đưa ra một quy trình đổ nước đại loại như:

- M¹c đầy nước vào bình Y bình, $Vx=0, Vy=9$
- Đ¹ hết nước từ bình Y sang bình X đến khi bình X đầy: $Vx=7, Vy=2$
- Đ¹ hết nước trong bình X đi: $Vx=0, Vy=2$
- Đ¹ hết nước còn lại từ bình Y sang bình X, $Vx=2, Vy=0$
- M¹c đầy bình Y, $Vx=2, Vy=9$
- Đ¹ hết qua từ bình Y sang bình X đến khi bình X đầy, $Vx=7, Vy=5$
- L¹ợng nước còn lại chính là số nước cần đo.

Tuy nhiên, với những số liệu khác, bạn phải "mày mò" lại từ đầu để tìm ra quy trình đổ nước. Cứ thế, mỗi một trường hợp sẽ có một cách đổ nước hoàn toàn khác nhau. Như vậy, nếu có một ai đó yêu cầu bạn đưa ra một cách làm tổng quát thì

chính bạn cũng sẽ lúng túng (dĩ nhiên, ngoại trừ trường hợp bạn đã biết trước cách giải theo tri thức mà chúng ta sắp sửa tìm hiểu ở đây!).

Sau khi phân tích bài toán và thử nghiệm các kết quả rút ra quy luận, người dùng thử "truyền" những kinh nghiệm này cho máy tính và để cho máy tính "mày mò" tìm các thao tác. Điều này hoàn toàn có lợi, vì máy tính có khả năng "mày mò" hơn hẳn chúng ta! Nếu những "kinh nghiệm" mà chúng ta cung cấp cho máy tính không giúp chúng ta tìm được lời giải, chúng ta sẽ thay thế nó bằng những kinh nghiệm khác và lại tiếp tục để máy tính tìm kiếm lời giải!

Phát biểu lại bài toán một cách tổng quát như sau: giả sử rằng $VX < VY$.

Gọi lượng nước chứa trong bình X là x ($0 \leq x \leq VX$)

Gọi lượng nước chứa trong bình Y là y ($0 \leq y \leq VY$)

Như vậy, điều kiện kết thúc của bài toán sẽ là :

$$x = z \text{ hoặc } y = z$$

Điều kiện đầu của bài toán là : $x = 0$ và $y = 0$

Quá trình giải được thực hiện bằng cách xét lần lượt các luật sau, luật nào thỏa mãn thì sẽ được áp dụng. Lúc này, các luật chính là các "kinh nghiệm" hay tri thức mà ta đã chuyển giao cho máy tính. Sau khi áp dụng luật, trạng thái của bài toán sẽ thay đổi, ta lại tiếp tục xét các luật kế tiếp, nếu hết luật, quay trở lại luật đầu tiên. Quá trình tiếp diễn cho đến khi đạt được điều kiện kết thúc của bài toán.

Ba luật này được mô tả như sau :

(L1) Nếu bình X đầy thì đổ hết nước trong bình X đi.

(L2) Nếu bình Y rỗng thì đổ đầy nước vào bình Y.

(L3) Nếu bình X không đầy và bình Y không rỗng thì hãy trút nước từ bình Y sang bình X (cho đến khi bình X đầy hoặc bình Y hết nước).

Trên thực tế, lúc đầu để giải trường hợp tổng quát của bài toán này, người ta đã dùng đến hơn 15 luật (kinh nghiệm) khác nhau. Tuy nhiên, sau này, người ta đã rút gọn lại chỉ còn 3 luật như trên.

Bạn có thể dễ dàng chuyển đổi cách giải này thành chương trình như sau:

```
x := 0; y := 0;
```

```
WHILE ( (x <> z) AND (y <> 0) ) DO BEGIN
```

```
IF (x = Vx) THEN x := 0;
```

```
IF (y = 0) THEN (y := Vy);
```

```
IF (y > 0) THEN BEGIN
```

```
k := min(Vx - x, y);
```

```
x := x + k;
```

```
y := y - k;
```

```
END;
```

```
END;
```

```
...
```

Thử "chạy" chương trình trên với số liệu cụ thể là :

$V_x = 3$, $V_y = 4$ và $z = 2$

Ban đầu : $x = 0, y = 0$

Luật (L2) $\rightarrow x = 0, y = 4$

Luật (L3) $\rightarrow x = 3, y = 1$

Luật (L1) $\rightarrow x = 0, y = 1$

Luật (L3) $\rightarrow x = 1, y = 0$

Luật (L2) $\rightarrow x = 1, y = 4$

Luật (L3) $\rightarrow x = 3, y = 2$

3 luật mà chúng ta đã cài đặt trong chương trình ở trên được gọi là **cơ sở tri thức**. Còn cách thức tìm kiếm lời giải bằng cách duyệt tuần tự từng luật và áp dụng nó được gọi là **động cơ suy diễn**.

🔴➡️ *Trạng thái cơ bản :*

Bình X đầy, Bình X rỗng, Bình X không rỗng, Bình X có n lít nước.

🔴➡️ *Thao tác*

Đổ hết nước trong bình, đổ đầy nước trong bình, đổ nước từ bình A sang bình B cho đến khi B đầy hoặc A rỗng.

Lưu ý rằng ta không thể có thao tác "Đổ n lít nước từ A sang B" vì bài toán đã giả định rằng các bình đều không có vạch chia, hơn nữa nếu ta biết cách đổ n lít nước từ A sang B thì lời giải bài toán trở thành quá đơn giản.

"Mức đầy X"

"Đổ z lít nước từ X sang Y"

Vì đây là một bài toán đơn giản nên bạn có thể dễ nhận thấy rằng, các trạng thái cơ bản và thao tác chẳng có gì khác so với các điều kiện mà chúng ta đã đưa ra.

Chương trình minh họa bằng ngôn ngữ lập trình C++

```
#include<stdio.h>
#include<conio.h>
#include<iostream>
#include<math.h>
using namespace std;
int minXY(int A,int B)
{
return A>B?B:A;
}
int main()
{
int X,Y,Vx,Vy,Z,K,min,d=0;
X=0;
Y=0;
Vx=X;
Vy=Y;
cout<<"\nNhập Vx: "; cin>>Vx;
cout<<"\nNhập Vy: "; cin>>Vy;
cout<<"\nNhập Z: "; cin>>Z;
cout<<"\nKET QUA: ";
cout<<"\n";
while ((X!=Z) && (Y!=Z))
{
if (X==Vx)
{
```

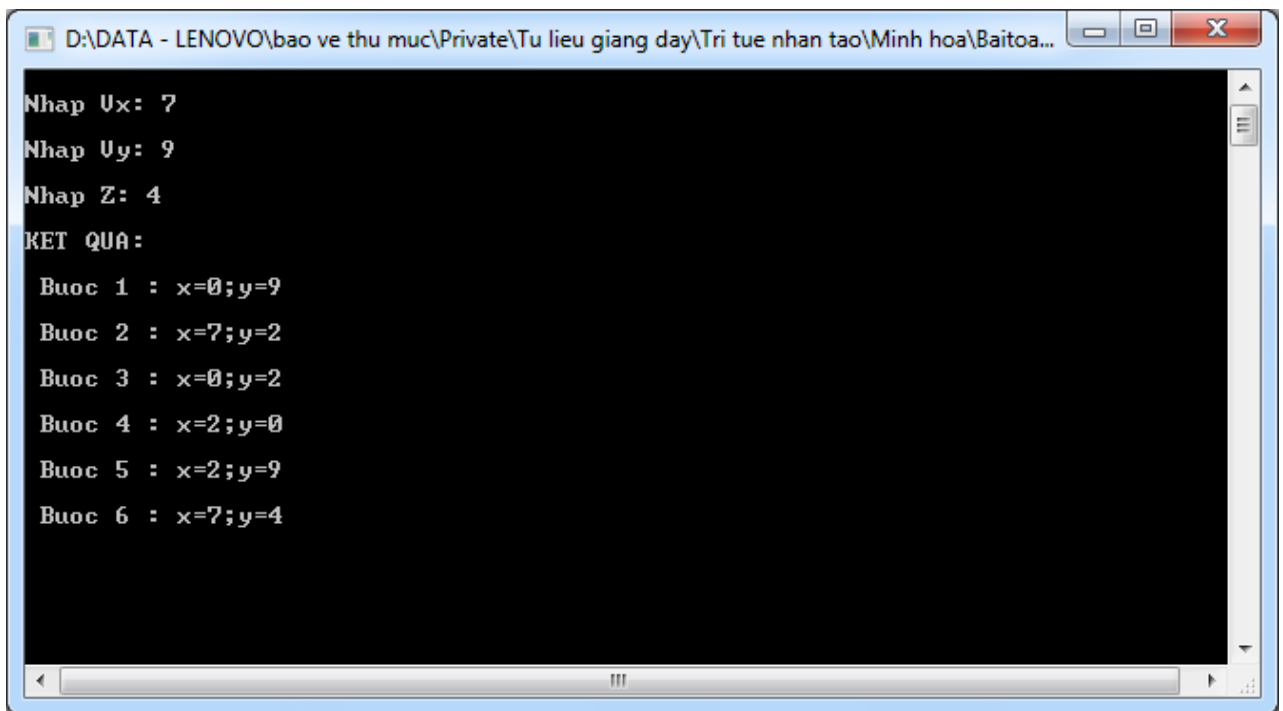
```

X=0;
d=d+1;
cout<<"\n Buoc " << d << " : x="<<X<<"<<y="<<Y<<"\n";
}
if (Y==0)
{
Y=Vy;
d=d+1;
cout<<"\n Buoc " << d << " : x="<<X<<"<<y="<<Y<<"\n";
}
if (Y>0)
{
K=minXY(Vx-X,Y);
X=X+K;
Y=Y-K;
d=d+1;
cout<<"\n Buoc " << d << " : x="<< X<<"<<y="<<Y<<"\n";
}
}

getch();
return 0;
}

```

Kết quả chạy chương trình



```
D:\DATA - LENOVO\bao ve thu muc\Private\Tu lieu giang day\Tri tue nhan tao\Minh hoa\Baitoa...
Nhap Ux: 7
Nhap Uy: 9
Nhap Z: 4
KET QUA:
Buoc 1 : x=0;y=9
Buoc 2 : x=7;y=2
Buoc 3 : x=0;y=2
Buoc 4 : x=2;y=0
Buoc 5 : x=2;y=9
Buoc 6 : x=7;y=4
```

Với tư tưởng thuật giải trên sinh viên có thể viết bằng nhiều ngôn ngữ lập trình khác nhau.

3.3.2 Biểu diễn tri thức và suy diễn với logic mệnh đề.

a. Biểu diễn tri thức với logic mệnh đề.

Định nghĩa: Logic mệnh đề là công cụ toán logic, trong đó các mệnh đề được gán cho một biến, hoặc hằng; còn các biểu thức là sự liên kết có nghĩa giữa các biến hằng với một số toán tử nhất định.

Ví dụ: Mệnh đề “Nếu trời mưa (A) thì đất ướt (B) “ được mô tả: $A \Rightarrow B$

Ngôn ngữ biểu diễn tri thức = Cú pháp + Ngữ nghĩa + Cơ chế suy diễn.

- **Cú pháp:** cú pháp của logic mệnh đề gồm tập các ký hiệu và tập các luật xây dựng công thức.

- Các ký hiệu

Hai hằng logic: True và False.

Các ký hiệu mệnh đề: P, Q,...

Các phép kết nối logic: \wedge , \vee , \neg , \Rightarrow , \Leftrightarrow .

Các dấu mở ngoặc và đóng ngoặc: (...)

- Các quy tắc xây dựng các công thức:

Nếu A và B là công thức thì:

$(A \wedge B)$ (đọc “A hội B” hoặc “A và B”)

$(A \vee B)$ (đọc “A tuyển B” hoặc “A hoặc B”)

$(\neg A)$ (đọc “phủ định A”)

$(A \Rightarrow B)$ (đọc “A kéo theo B” hoặc “nếu A thì B”)

$(A \Leftrightarrow B)$ (đọc “A và B kéo theo nhau”)

là các công thức.

Để ngắn gọn, ta bỏ đi các cặp dấu ngoặc khi không cần thiết. Chẳng hạn, thay cho $((A \vee B) \wedge C)$ ta sẽ viết là $(A \vee B) \wedge C$.

- Ngữ nghĩa:

Ngữ nghĩa của logic mệnh đề cho phép ta xác định ý nghĩa của các công thức trong thế giới hiện thực nào đó. Điều đó được thực hiện bằng cách kết hợp mỗi ký hiệu mệnh đề với sự kiện nào đó trong thế giới hiện thực.

Ví dụ: ký hiệu mệnh đề P là một sự kiện “Hà Nội là thủ đô nước Việt Nam”.

Một sự kiện chỉ có thể đúng hoặc sai gán cho mỗi ký hiệu mệnh đề một giá trị chân lý True hoặc False.

Ý nghĩa của các kết nối logic trong các bảng chân lý:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

Bảng chân lý của các kết nối logic

- Sự tương đương của các công thức:

Hai công thức A và B được xem là tương đương nếu chúng có cùng một giá trị chân lý. Để chỉ A tương đương với B ta viết $A \equiv B$ bằng phương pháp bảng chân lý, dễ dàng chứng minh được sự tương đương của các công thức sau đây :

$$\begin{aligned}
 A \Rightarrow B &\equiv \neg A \vee B \\
 A \Leftrightarrow B &\equiv (A \Rightarrow B) \wedge (B \Rightarrow A) \\
 \neg(\neg A) &\equiv A
 \end{aligned}$$

Luật De Morgan:

$$\begin{aligned}
 \neg(A \vee B) &\equiv \neg A \wedge \neg B \\
 \neg(A \wedge B) &\equiv \neg A \vee \neg B
 \end{aligned}$$

Luật giao hoán:

$$\begin{aligned}
 A \vee B &\equiv B \vee A \\
 A \wedge B &\equiv B \wedge A
 \end{aligned}$$

Luật kết hợp:

$$\begin{aligned}
 (A \vee B) \vee C &\equiv A \vee (B \vee C) \\
 (A \wedge B) \wedge C &\equiv A \wedge (B \wedge C)
 \end{aligned}$$

Luật phân phối:

$$\begin{aligned}
 A \wedge (B \vee C) &\equiv (A \wedge B) \vee (A \wedge C) \\
 A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C)
 \end{aligned}$$

- **Dạng chuẩn tắc:**

Để dễ dàng viết các chương trình máy tính thao tác trên các công thức, chúng ta sẽ chuẩn hóa các công thức, đưa chúng về dạng biểu diễn chuẩn được gọi là dạng chuẩn hội. Chúng ta có thể biến đổi một công thức bất kỳ về công thức ở dạng chuẩn hội bằng cách áp dụng các thủ tục sau:

- Bỏ các dấu kéo theo (\Rightarrow) bằng cách thay $(A \Rightarrow B)$ bởi $(\neg A \vee B)$.
- Chuyển các dấu phủ định (\neg) vào sát các kết hiệu mệnh đề bằng cách áp dụng luật De Morgan và thay $\neg(\neg A)$ bởi A .
- Áp dụng luật phân phối, thay các công thức có dạng $A \vee (B \wedge C)$ bởi $(A \vee B) \wedge (A \vee C)$.

Ví dụ : Ta chuẩn hóa công thức $(P \Rightarrow Q) \vee \neg(R \vee \neg S)$:

$$(P \Rightarrow Q) \vee \neg(R \vee \neg S) \equiv (\neg P \vee Q) \vee (\neg R \wedge S)$$

$$\equiv ((IP \vee Q) \vee IR) \wedge ((IP \vee Q) \vee S)$$

$$\equiv (IP \vee Q \vee IR) \wedge (IP \vee Q \vee S).$$

. Như vậy công thức $(P \Rightarrow Q) \vee I(R \vee IS)$ được đưa về dạng chuẩn hội $(IP \vee Q \vee IR) \wedge (IP \vee Q \vee S)$.

b. Luật suy diễn.

Một công thức H được xem là hệ quả logic của một tập công thức $G = \{G_1, \dots, G_m\}$ nếu $\{G_1, \dots, G_m\}$ đúng thì H cũng đúng. Nói cách khác bất kỳ mô hình nào của G cũng là mô hình của H.

Khi có một cơ sở tri thức, ta muốn sử dụng các tri thức trong cơ sở này để suy ra tri thức mới mà nó là hệ quả logic của các công thức trong cơ sở tri thức. Điều đó được thực hiện bằng cách sử dụng các luật suy diễn. Luật suy diễn giống như một thủ tục mà chúng ta sử dụng để sinh ra một công thức mới từ các công thức đã có. Một luật suy diễn gồm hai phần: một tập các điều kiện và một kết luận. Chúng ta sẽ biểu diễn các luật suy diễn dưới dạng “phân số”, trong đó tử số là danh sách các điều kiện, còn mẫu số là kết luận của luật, tức là mẫu số là công thức mới được suy ra từ các công thức ở tử số.

Một số luật suy diễn quan trọng trong logic mệnh đề. Trong các luật này $\alpha, \alpha_i, \beta, \gamma$ là các công thức:

- Luật Modus Ponens:

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

- Luật Modus Tollens:

$$\frac{\alpha \Rightarrow \beta, \neg \beta}{\neg \alpha}$$

- Luật bắc cầu:

$$\frac{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{\alpha \Rightarrow \gamma}$$

- Luật loại bỏ hội:

$$\frac{\alpha_1 \wedge \dots \wedge \alpha_i \wedge \dots \wedge \alpha_m}{\alpha_i}$$

- Luật đưa vào hội:

$$\frac{\alpha_1, \dots, \alpha_i, \dots, \alpha_m}{\alpha_1 \wedge \dots \wedge \alpha_i \wedge \dots \wedge \alpha_m}$$

- Luật đưa vào tuyển:

$$\frac{\alpha_i}{\alpha_1 \vee \dots \vee \alpha_i \vee \dots \vee \alpha_m}$$

- Luật phân giải:

$$\frac{\alpha \vee \beta, \neg \beta \vee \gamma}{\alpha \vee \gamma}$$

Tiên đề, định lý, chứng minh.

Các công thức đã cho được gọi là các tiên đề. Các công thức được suy ra được gọi là các định lý. Dãy các luật được áp dụng để dẫn tới định lý được gọi là một chứng minh của định lý.

Giả sử ta có các công thức sau:

$$Q \wedge S \Rightarrow G \vee H \quad (1)$$

$$P \Rightarrow Q \quad (2)$$

$$R \Rightarrow S \quad (3)$$

$$P \quad (4)$$

$$R \quad (5)$$

Giả sử ta cần chứng minh công thức GVH. Từ công thức (2) và (4), ta suy ra Q (Luật Modus Ponens). Lại áp dụng luật Modus Ponens, từ (3) và (5) ta suy ra S. Từ Q, S ta suy ra QAS (luật đưa vào hội). Từ (1) và QAS ta suy ra GVH. Công thức GVH đã được chứng minh.

3.3.3. Biểu diễn tri thức và suy diễn với logic vị từ.

a. Biểu diễn tri thức với logic vị từ.

Logic mệnh đề cho phép ta biểu diễn các sự kiện. Logic vị từ cho phép ta biểu diễn tri thức về thế giới với các đối tượng, các thuộc tính của đối tượng và các quan hệ của đối tượng.

Ví dụ: cho mệnh đề “trời đã mưa vào thứ ba”.

Cách biểu diễn mệnh đề này dùng logic mệnh đề: $R = \text{trời đã mưa vào thứ ba}$.

Với cách biểu diễn này ta chỉ xác minh được giá trị chân lý của ký hiệu mệnh đề R nhưng không thể truy cập các thành phần đối tượng bên trong của mệnh đề như “đã mưa” và “thứ ba”. Đó là tình huống thời tiết và thời gian.

Cách biểu diễn mệnh đề trên dùng logic vị từ : $\text{thoitiet}(\text{thuba}, \text{mua})$. Với cách biểu diễn này, ta có thể truy cập các thành phần đối tượng trong mệnh đề như thuba, mua . Trong cách biểu diễn này, ta cũng có thể thay thế biến X biểu diễn lớp của các đối tượng trong tuần với đặc trưng thuba điển hình là: $\text{weather}(X, \text{mua})$.

- Cú pháp: gồm có các ký hiệu chân lý, ký hiệu vị từ và các phép toán logic.

Ký hiệu vị từ gồm có hằng vị từ, biến vị từ, hàm vị từ, vị từ và vị từ định lượng.

- Hằng vị từ: là chuỗi các chữ cái in thường dùng để biểu diễn tên riêng hoặc thuộc tính riêng của đối tượng.

Ví dụ: Nam, cao, xanh là các ký hiệu hằng vị từ hợp lệ.

- Biến vị từ: chuỗi các chữ cái với ít nhất chữ cái đầu tiên của chuỗi phải là chữ cái in hoa dùng để biểu diễn lớp của các đối tượng.

Ví dụ: X là biến vị từ dùng để biểu diễn lớp của các đối tượng ngày trong tuần.

- Hàm vị từ: là ánh xạ từ một hoặc nhiều phần tử của tập hợp này đến một phần tử duy nhất trong một tập hợp khác. Hàm phải có tên riêng và các đối số của nó. Tên của hàm vị từ được qui ước là chuỗi các chữ cái in thường. Cú pháp tổng quát của hàm: $\text{Tên_hàm}(\text{các đối số vào của hàm})$.

Hàm nhận các đối số vào từ một tập hợp này và trả về duy nhất một đối số ra trong một tập hợp khác.

Ví dụ: Linh là bố của Nam.

Mệnh đề này có thể biểu diễn bằng hàm vị từ “bố”: $bo(Nam)$. Hàm trả về giá trị ra của nó là Linh.

Các đối số của hàm vị từ có thể là hằng vị từ hoặc biến vị từ.

- Vị từ: dạng đặc biệt của hàm vị từ. Vị từ cũng phải có tên riêng và các đối số vào của nó. Tên vị từ thường là mối quan hệ giữa hai hoặc nhiều đối tượng trong một mệnh đề. Qui ước đặt tên cho vị từ cũng giống như hàm vị từ. Vị từ nhận các đối số vào từ một tập hợp và trả về đối số ra trong một tập hợp giá trị chân lý đúng hoặc sai.

Ví dụ: cho mệnh đề Linh thích Trang.

Mệnh đề này có thể được biểu diễn bằng vị từ thích: $thich(Linh, Trang)$.

- Vị từ định lượng: để xác định phạm vi giá trị của biến vị từ, hai đại lượng đứng trước biến là \forall và \exists , hai vị từ này gọi là vị từ định lượng.

Vị từ định lượng \forall đứng trước biến để xác định biểu thức là đúng cho mọi giá trị của biến.

Vị từ định lượng \exists đứng trước biến để xác định biểu thức là đúng cho một vài giá trị của biến.

b. Các luật suy diễn

- Luật thay thế phổ dụng:

Giả sử G là một câu, câu $\forall x G$ là đúng trong một minh hoạ nào đó nếu và chỉ nếu G đúng đối với tất cả các đối tượng nằm trong miền đối tượng của minh hoạ đó. Mỗi hạng thức t ứng với một đối tượng vì thế nếu câu $\forall x G$ đúng thì khi thay tất cả các xuất hiện của biến x bởi hạng thức t ta nhận được câu đúng. Công thức nhận được từ công thức G bằng cách thay tất cả các xuất hiện của x bởi t được kí hiệu là $G[x/t]$. Luật thay thế phổ dụng phát biểu rằng, từ công thức $\forall x G$ suy ra công thức $G[x/t]$.

$$\frac{\forall x G}{G[x/t]}$$

Ví dụ: từ câu $\forall x \text{ thích}(x, \text{bongda})$ (mọi người đều thích bóng đá), bằng cách thay x bởi An ta suy ra câu $\text{thích}(\text{An}, \text{bongda})$ (An thích bóng đá) .

- Luật Modus Ponens tổng quát.

Giả sử P_i, P_i' ($i=1, \dots, n$) và Q là các công thức phân tử sao cho tất cả các cặp câu P_i, P_i' hợp nhất được bởi phép thế θ , tức là $P_i\theta = P_i'$ ($i=1, \dots, n$). Khi đó ta có luật:

$$\frac{(P_1 \wedge \dots \wedge P_n \Rightarrow Q), P_1', \dots, P_n'}{Q'}$$

Ví dụ: Giả sử ta có các câu $(\text{sinhvien}(x) \wedge \text{nam}(x) \Rightarrow \text{thích}(x, \text{bongda}))$ và $\text{sinhvien}(\text{Anh}), \text{nam}(\text{Anh})$. Với phép thế $\theta = [x/\text{Anh}]$, các cặp câu $\text{sinhvien}(x), \text{nam}(\text{Anh})$ hợp nhất được. Do đó ta suy ra câu $\text{thích}(\text{Anh}, \text{bongda})$.

- Luật hợp nhất

Ví dụ: cho hai câu

$\text{ban}(x, \text{Ba}) \Rightarrow \text{tot}(x)$ (Mọi bạn của Ba đều là người tốt)

$\text{ban}(\text{Lan}, y)$ (Lan là bạn của tất cả mọi người)

Ta có thể hợp nhất hai câu : $\text{ban}(x, \text{Ba}) \Rightarrow \text{tot}(x)$ và $\text{ban}(\text{Lan}, y)$ bởi phép thay thế $[x/\text{Lan}, y/\text{Ba}]$. áp dụng luật thay thế phổ dụng với phép thay thế này ta nhận được hai câu:

$\text{ban}(\text{Lan}, \text{Ba}) \Rightarrow \text{tot}(\text{Lan})$

$\text{ban}(\text{Lan}, \text{Ba})$

Từ hai câu này, theo luật Modus Ponens, ta suy ra câu $\text{tot}(\text{Lan})$ (Lan là người tốt).

Một cách tổng quát, một phép thế θ là một dãy các cặp x_i/t_i , $\theta = [x_1/t_1 \ x_2/t_2 \dots \ x_n/t_n]$ trong đó các x_i là các biến khác nhau, các t_i là các hạng thức và các x_i không có mặt trong t_i ($i=1, \dots, n$). Áp dụng phép thế θ vào công thức G , ta nhận được công thức $G\theta$, đó là công thức nhận được từ công thức G bằng cách thay mỗi sự xuất hiện của các x_i bởi t_i . Chẳng hạn, nếu $G = P(x, y, f(a, x))$ và $\theta = [x/b, y/g(z)]$ thì $G\theta = P(b, g(z), f(a, b))$.

Với hai câu phân tử G và H mà tồn tại phép thế θ sao cho $G\theta$ và $H\theta$ trở thành đồng nhất ($G\theta=H\theta$) thì G và H được gọi là hợp nhất được, phép thế θ được gọi là hợp nhất tử của G và H . Chẳng hạn, hai câu thích(A_n,y) và thích($x,bongda$) là hợp nhất được bởi hợp nhất tử $[x/A_n,y/bongda]$.

3.4. Các kỹ thuật suy diễn

3.4.1 Suy diễn tiến.

Thủ tục tìm kiếm suy diễn tiến được mô tả như sau:

- Bắt đầu tìm kiếm từ dữ liệu ban đầu của bài toán.
- Chọn tất cả các luật hợp với dữ liệu ban đầu của bài toán để vẽ kết luận sinh ra các dữ liệu mới.
- Tại mỗi điểm dữ liệu mới, chọn tất cả các luật phù hợp với dữ liệu mới để vẽ kết luận phát sinh ra các dữ liệu mới hơn.
- Thủ tục này được lặp lại cho tất cả các dữ liệu mới cho đến khi dữ liệu đích được tìm thấy.

Ví dụ 1:

Sự kiện ban đầu : H, K

$R3 : H \rightarrow A \{ A, H, K \}$

$R1 : A \rightarrow E \{ A, E, H, H \}$

$R5 : E \wedge K \rightarrow B \{ A, B, E, H, K \}$

$R2 : B \rightarrow D \{ A, B, D, E, H, K \}$

$R6 : D \wedge E \wedge K \rightarrow C \{ A, B, C, D, E, H, K \}$

Ví dụ 2: Giả sử cơ sở luật (cơ sở luật về các động vật trong sở thú) gồm các luật sau

Luật 1: nếu động vật có lông mao

thì động vật là loài có vú

Luật 2: nếu động vật có lông vũ

thì động vật là chim

Luật 3: nếu 1. động vật biết bay, và

2. động vật đẻ trứng

thì động vật là chim

Luật 4: nếu 1. động vật là loài có vú, và

2. động vật ăn thịt

thì động vật là thú ăn thịt

Luật 5: nếu 1. động vật là loài có vú, và

2. động vật có răng nhọn, và

3. động vật có móng vuốt

thì động vật là thú ăn thịt

Luật 6: nếu 1. động vật là thú ăn thịt, và

2. động vật có màu lông vàng hung, và

3. động vật có đốm sẫm

thì động vật là báo Châu Phi

Luật 7: nếu 1. động vật là thú ăn thịt, và

2. động vật có màu lông vàng hung, và

3. động vật có vằn đen

thì động vật là hổ

Luật 8: nếu 1. động vật là chim, và

2. động vật không biết bay, và

3. động vật có chân dài, và

4. động vật có cổ dài

thì động vật là đà điểu

Luật 9: nếu 1. động vật là chim, và

2. động vật không biết bay, và
 3. động vật biết bơi, và
 4. động vật có lông đen và trắng
- thì động vật là chim cánh cụt

Giả sử một em bé quan sát một con vật có tên là Ki trong sở thú, em thấy nó có các đặc điểm sau:

Ki có lông mao

Ki ăn thịt

Ki có màu lông vàng hung

Ki có đốm sẫm

Lúc này cơ sở sự kiện sẽ bao gồm các sự kiện trên.

Thủ tục lập luận tiến xem xét luật 1. Khi biến “động vật” trong luật này được thay bởi Ki, điều kiện của luật trở thành “Ki có lông mao”, đây là một sự kiện có trong bộ nhớ làm việc, do đó ta suy ra “Ki là loài có vú”. Đây là sự kiện mới, do đó nó được thêm vào bộ nhớ làm việc. Xét luật 4, thế biến “động vật” bởi Ki, thì hai điều kiện của luật trở thành:

Ki là loài có vú, và

Ki ăn thịt

Cả hai sự kiện này đều có trong bộ nhớ làm việc, do đó từ luật 4 ta suy ra “Ki là thú ăn thịt”. Sự kiện mới này lại được thêm vào bộ nhớ làm việc. Ta xét tiếp luật 6, thế biến “động vật” bởi Ki, các điều kiện của luật trở thành:

Ki là loài thú ăn thịt, và

Ki có màu lông vàng hung, và

Ki có đốm sẫm

Tất cả các điều kiện này đều đúng, do đó từ luật 6, ta suy ra “Ki là báo Châu Phi”. Như vậy từ các sự kiện đã biết về Ki, lập luận tiến đã suy ra các sự kiện mới sau :

Ki là loài có vú.

Ki là thú ăn thịt.

Ki là báo Châu Phi.

- Ưu điểm của suy diễn tiến :

- Suy diễn tiến làm việc hiệu quả khi cần thu thập thêm thông tin và thấy được điều cần suy diễn.

- Cho khối lượng thông tin mới từ những thông tin đã có.

- Phù hợp với các bài toán lập lịch

- Nhược điểm của suy diễn tiến :

- Không phân biệt được độ quan trọng khác nhau của thông tin thu được.

- Thông tin thu được có thể không liên quan đến chủ đề mà người dùng quan tâm.

3.4.2 Suy diễn lùi.

Thủ tục suy diễn lùi được mô tả như sau :

- Thủ tục bắt đầu suy diễn từ kết luận của bài toán.

- Chọn tất cả các luật phù hợp với kết luận đích, thiết lập dữ liệu ở về điều kiện phát sinh ra đích làm kết luận mới.

- Tại mỗi điểm kết luận mới, chọn tất cả các luật phù hợp với kết luận mới thiết lập dữ liệu ở về điều kiện phát sinh ra kết luận làm kết luận mới hơn.

- Thủ tục này lặp lại cho tất cả các đích mới cho đến khi nào dữ liệu ban đầu của bài toán được tìm thấy.

Ví dụ 1 : Giả sử bộ nhớ làm việc chứa các sự kiện sau :

Bibi có lông vũ

Bibi có chân dài

Bibi có cổ dài

Bibi không biết bay

Ta đưa ra giả thuyết sau đây

Bibi là đà điểu

Đối sánh giả thuyết này với phần kết luận của các luật, ta thấy nó khớp với kết luận của luật 8 nếu thể biến “động vật” bởi Bibi. Từ luật 8, ta suy ra rằng, giả thuyết “Bibi là đà điểu” là đúng, nếu các điều kiện sau là đúng

1. Bibi là chim
2. Bibi không biết bay
3. Bibi có chân dài
4. Bibi có cổ dài

Đây là 4 giả thuyết mới. Việc đánh giá giả thuyết “Bibi là đà điểu” được quy về việc đánh giá bốn giả thuyết mới này. Các giả thuyết 2, 3 và 4 đều có trong bộ nhớ làm việc, ta chỉ cần đánh giá giả thuyết “Bibi là chim”. Lại đối sánh giả thuyết này với phần kết luận của các luật. Ta thấy nó khớp với kết luận của luật 2 và luật 3. Xét luật 3, đi lùi lại phần điều kiện của luật này, ta nhận được các giả thuyết mới là:

Bibi biết bay

Bibi đẻ trứng

Cả hai giả thuyết này đều không có trong bộ nhớ làm việc và cũng không khớp với phần kết luận của luật nào cả. Do đó, ta không thể phát triển tiếp các giả thuyết này được nữa. Chuyển sang xét luật 2, để “Bibi là chim” luật này đòi hỏi điều kiện “Bibi có lông vũ”. Điều kiện này có trong bộ nhớ làm việc. Vậy giả thuyết đã đưa ra “Bibi là đà điểu” là đúng.

- Ưu điểm của suy diễn lùi:
 - Phù hợp với bài toán đưa ra giả thiết rồi kiểm tra kết luận có đúng không?
 - Tập trung vào đích đã cho suy ra luôn đúng với vấn đề mà người dùng quan tâm.
 - Chỉ liên quan đến tri thức đã có (không mở rộng được tập tri thức) nên không bị lệch chủ đề.
- Nhược điểm
 - Tiếp tục quá trình suy diễn trong khi có thể dừng lại.
 - Ví dụ 2:

Tập các sự kiện :

- Ổ cứng là "hồng" hay "hoạt động bình thường"
- Hồng màn hình.
- Lỏng cáp màn hình.
- Tình trạng đèn ổ cứng là "tắt" hoặc "sáng"
- Có âm thanh đọc ổ cứng.
- Tình trạng đèn màn hình "xanh" hoặc "chớp đỏ"
- Không sử dụng được máy tính.
- Điện vào máy tính "có" hay "không"

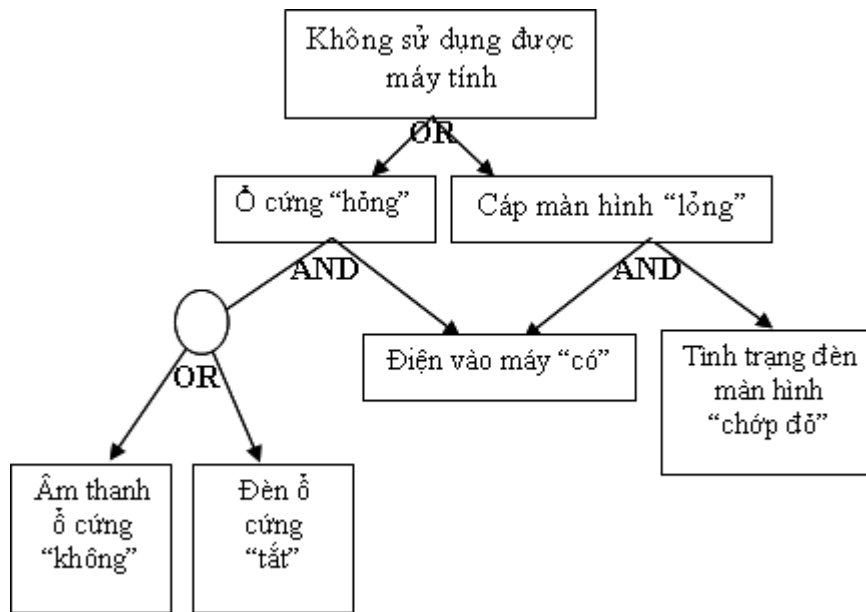
Tập các luật :

R1. Nếu (ổ cứng "hồng") hoặc (cáp màn hình "lỏng")) thì không sử dụng được máy tính.

R2. Nếu (điện vào máy là "có") và (âm thanh đọc ổ cứng là "không") hoặc tình trạng đèn ổ cứng là "tắt")) thì (ổ cứng "hồng").

R3. Nếu (điện vào máy là "có") và (tình trạng đèn màn hình là "chớp đỏ") thì (cáp màn hình "lỏng").

Để xác định được các nguyên nhân gây ra sự kiện "không sử dụng được máy tính", ta phải xây dựng một cấu trúc đồ thị gọi là đồ thị AND/OR như sau :



Như vậy là để xác định được nguyên nhân gây ra hỏng hóc là do ổ cứng hỏng hay cáp màn hình lỏng, hệ thống phải lần lượt đi vào các nhánh để kiểm tra các điều kiện như điện vào máy "có", âm thanh ổ cứng "không"... Tại một bước, nếu giá trị cần xác định không thể được suy ra từ bất kỳ một luật nào, hệ thống sẽ yêu cầu người dùng trực tiếp nhập vào. Chẳng hạn như để biết máy tính có điện không, hệ thống sẽ hiện ra màn hình câu hỏi "*Bạn kiểm tra xem có điện vào máy tính không (kiểm tra đèn nguồn)? (C/K)*". Để thực hiện được cơ chế suy luận lùi, người ta thường sử dụng ngăn xếp (để ghi nhận lại những nhánh chưa kiểm tra).

3.5. Lập trình Prolog

3.5.1. Tổng Quan

Prolog là ngôn ngữ lập trình logic do GS. A. Colmerauer đưa ra lần đầu tiên năm 1972, đến năm 1980 phát triển rộng rãi và được người nhật chọn làm ngôn ngữ phát triển máy tính thế hệ 5. Prolog đã được cài đặt trên hầu hết các hệ điều hành Unix, Linux, Windows.

Prolog còn được gọi là ngôn ngữ lập trình ý hiệu. Nguyên lý lập trình logic dựa trên phép suy diễn logic, liên quan đến những khái niệm toán học như logic Horn, logic vị từ bậc một, ...

Một mệnh đề Horn biểu diễn một sự kiện hay một sự việc nào đó là đúng hoặc không đúng, xảy ra hoặc không xảy ra (có hoặc không có,...).

Ví dụ mệnh đề Horn:

- “Tất cả mọi người đều chết” hoặc “Nếu ai là người thì ai đó phải chết”.(luật_rule)
- Hoa là người.(sự kiện_fact)

Một chương trình logic có thể được xem như là một cơ sở dữ liệu gồm các mệnh đề Horn, hoặc dạng luật, hoặc dạng sự kiện.

Người sử dụng gọi chạy một chương trình bằng cách đặt câu hỏi truy vấn trên cơ sở dữ liệu này:

- Hoa có chết không?(Tương đương khẳng định Hoa chết đúng hay sai?)

Prolog rất thích hợp để giải quyết bài toán liên quan đến các đối tượng và mối quan hệ giữa chúng. Prolog được ứng dụng chủ yếu trong lĩnh vực trí tuệ nhân tạo như công nghệ xử lý tri thức, hệ chuyên gia, máy học, xử lý ngôn ngữ, trò chơi,...

3.5.2. Môi trường lập trình Prolog(Môi trường SWI – Prolog)

a. Phép toán trong SWI-Prolog

- Phép toán số học : +, -, *, /, mod, //, **
- Biểu thức số học: được xây dựng nhờ vị từ **is**

Number **is** Expr

Number: đối tượng cơ bản

Expr: biểu thức số học được xây dựng từ các phép toán số học, các số và các biến.

- Phép so sánh số học : >, <, ==, >=, <=, <=

- Các hàm số học: sin, cos, tan, arctan, ln, log, exp, sqrt, round, trunc,...
- Các ghi chú trong Prolog nằm trong cặp /* */ hoặc sau dấu %

b. Trong SWI-Prolog, có các vị từ xác định kiểu dữ liệu:

var(V)	V là một biến?
nonvar(X)	X không phải là một biến?
atom(A)	A là một nguyên tố?
integer(I)	I là một số nguyên?
float(R)	R là một số thực(dấu chấm động)?
number(N)	N là một số (nguyên hoặc thực)?
atomic(A)	A là một nguyên tố hoặc một số?
compound(X)	X là một term có cấu trúc?

3.5.3.Cấu trúc Prolog

Chương trình Prolog gồm các phần:

domains

...

predicates

...

clauses

...

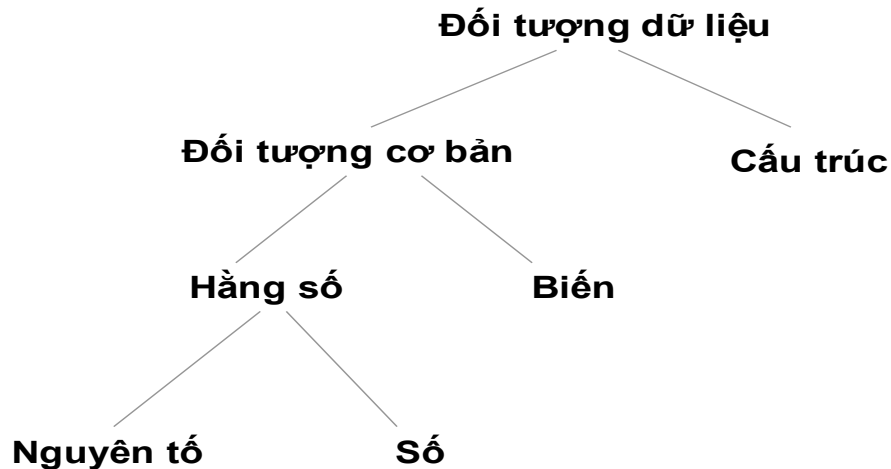
goal

...

(Không nhất thiết có đầy đủ các phần)

a. domains:

- ❖ phần định nghĩa các kiểu dữ liệu (Trong SWI-Prolog, người sử dụng không cần khai báo kiểu dữ liệu, cấu trúc chương trình không cần các từ khóa trên).
- ❖ Các kiểu dữ liệu trong prolog:



- Kiểu nguyên tố

Là chuỗi các ký tự

Chữ hoa từ A đến Z

Chữ thường từ a đến z

Chữ số 0,...,9

Ký tự đặc biệt : + - * / < > = : . & _ ~

Dựa trên quy tắc

Chuỗi các chữ cái, chữ số, dấu _ ,bắt đầu bằng một chữ thường
(VD : anna, x25, x__y, alpha_beta,...)

Chuỗi các ký tự đặc biệt (VD : <---> , === > , ...)

Chuỗi các ký tự nằm trong cặp dấu ‘ ’ (VD : ‘Hoàng’, ‘Hoa’,...)

- Kiểu số

Gồm số nguyên và số thực

Ví dụ :

Số nguyên : -3, -100, 1, 5, 2, ...

Số thập phân : 1.25, -3.25, ...

Số thực ít được sử dụng trong lập trình Prolog

- Kiểu biến

Biến

Chuỗi các chữ cái, chữ số và dấu _, bắt đầu bằng một chữ viết HOA hoặc dấu _.

Ví dụ :

X, Result, Object2, _23

Biến vô danh

Ký hiệu : _

Nếu biến chỉ xuất hiện một lần, không cần đặt tên. Sử dụng biến vô danh

Ví dụ :

haschild(X) :- parent(X,Y). %Biên Y chỉ xuất hiện một lần

haschild(X) :- parent(X,_). %Thay thế bằng biên vô danh

b. predicates: phần khai báo các quan hệ(vị từ).

Đối với Prolog, một chương trình có thể hiểu như là các tri thức được người lập trình cung cấp cho hệ thống Prolog, nhờ đó hệ thống có thể trả lời được những câu hỏi đặt ra. Đơn vị kiến thức mà người lập trình cung cấp cho Prolog gọi là các vị từ (predicate).

Các predicate có thể được xem như các hàm, giá trị trả về là các giá trị luận lý đúng hoặc sai. Giá trị này chỉ được dùng để suy luận.

Công việc đầu tiên khi lập trình Prolog là định nghĩa các vị từ.

Ví dụ:

Dữ kiện: Mọi người đều phải chết. Hoa là người.

Yêu cầu: Hệ thống phải trả lời được các câu hỏi: ai là người? ai không là người? ai phải chết? ai không phải chết?... Hệ thống sẽ tự động suy luận rằng Hoa phải chết.

Như vậy với bài toán đã nêu, chúng ta đặt ra 2 vị từ là:

nguoi(symbol)

chet(symbol)

c. clauses: phần thể hiện các mệnh đề(có thể) của vị từ.

Các mệnh đề clauses là lời giải thích để suy luận và trả lời các câu hỏi mà chúng ta đưa ra. Có 2 dạng mệnh đề là sự kiện(fact) và luật(rule). Sự kiện là điều mà chúng ta công nhận là đúng, luật là những quy tắc mà chúng ta xác định điều kiện đúng cho chúng.

Ví dụ: Như vậy với bài toán đã nêu

nguoi(Hoa). /* Sự kiện Hoa là người là điều chắc chắn đúng*/

chet(X): - nguoi(X). /* ta dùng luật khi dữ kiện ban đầu chưa chắc chắn, X phải chết nếu X là người*/

Mệnh đề rule sẽ bao gồm 2 phần, nằm 2 bên cặp ký hiệu “: - “. Phần bên trái cho biết vị từ đang được đề cập và các thông số tương ứng. Phần bên phải xác định điều kiện trả lời đúng cho luật trên, bao gồm các lời gọi các vị từ khác, được ngăn cách bởi ký hiệu ‘,’ gọi là mệnh đề con (sub-clause). Một luật chỉ trả lời đúng nếu tất cả sub-clause bên vế phải đều trả lời đúng. Ở ví dụ trên chỉ có một sub-clause xác định xem X có phải là người không. Như vậy chúng ta đã diễn tả được khái niệm một symbol sẽ phải chết nếu symbol đó là người.

d. goal: đích cần đạt được của chương trình.

Để thực thi chương trình, người sử dụng nhập yêu cầu(câu hỏi) của mình cho hệ thống. Yêu cầu này gọi là goal. Có 2 loại goal: goal nội và goal ngoại.

- **Để nhập goal ngoại** (external goal), sau khi đã hoàn tất việc soạn thảo hương trình, chúng ta dùng bàn phím gõ phím tắt Alt+R để chuyển sang vùng giao tiếp của

chương trình. Câu hỏi chúng ta đặt ra chỉ dựa vào tri thức chúng ta đã cung cấp cho hệ thống.

Ví dụ: chúng ta cung cấp cho hệ thống các khái niệm *nguoi* và *chet*, như vậy chúng ta chỉ có thể đặt câu hỏi liên quan đến 2 khái niệm này.

Nhập vào goal như sau:

chet(“Hoa”). /*câu trả lời là : Yes, vì ở đây chúng ta đã có khái niệm Hoa là người rồi*/

chet(“Huy”). /*câu trả lời là : No, ta chưa có khái niệm về Huy*/

chet(“X”). /* câu trả lời là:X= Hoa vì Hoa là người*/

- ***Goal nội***(internal goal) là khi chúng ta thêm phần goal này hẳn vào trong phần soạn thảo chương trình. Khi thực thi hệ thống sẽ không hỏi goal nữa, mà tự động thực hiện các yêu cầu trong phần goal. Tuy nhiên khi thực hiện goal nội, hệ thống sẽ không tự động in kết quả, chúng ta phải gọi vị từ *write* để làm điều này.

Ví dụ : chương trình được viết như sau :

predicates

nguoi(symbol)

chet(symbol)

clauses

nguoi(“Hoa”).

chet(X) : - nguoi(X).

goal

nguoi(X),write(X)

3.5.4. Bài toán tính giá trị giai thừa của một số nguyên bất kỳ

Chúng ta cung cấp dữ kiện cho hệ thống là : giai thừa của 0 là 1 và giai thừa của một số lớn hơn 0 là giai thừa của một số liền trước nó nhân với chính nó. Với cách đặt vấn đề này chúng ta chỉ còn một khái niệm phải biểu diễn là : giai thừa của một số là gì ?

Ở đây ta không được khai báo : `giaithua(integer)` vì một vị từ chỉ có thể trả lời là đúng hoặc sai, trong khi chúng ta đang mong muốn kết quả trả về là một số. Ngôn ngữ Prolog không có sự chồng chất hàm, nghĩa là kết quả của hàm(vị từ) không thể dùng làm thông số cho một vị từ khác, trong khi chúng ta đang định dùng kết quả này để tính giai thừa của $n-1$, để nhân tiếp cho n ra kết quả cuối cùng.

Vị từ thích hợp sẽ được khai báo như sau :

`giaithua(integer,integer)`

Điều này hiểu theo ngôn ngữ thủ tục, nghĩa là chúng ta khai báo một hàm có thông số là 2 số nguyên(`integer`) và kết quả trả về sẽ là đúng hoặc sai. Điều chúng ta muốn diễn tả có nghĩa là : giai thừa của một số nguyên sẽ là một số nguyên khác.

Chương trình hoàn chỉnh như sau :

`predicates`

`giaithua(integer,integer)`

`clauses`

`giaithua(0,1).`

`giaithua(X,Y) :- X1=X-1,giaithua(X1,Y1),Y=X*Y1.`

Chúng ta có thể viết ở goal dạng nghi vấn sau :

`giaithua(3,6)`

Hiểu theo ngôn ngữ tự nhiên : có phải giai thừa của 3 là 6 không ? Câu trả lời là : Yes

Hay : `giaithua(3,8)`

Hiểu theo ngôn ngữ tự nhiên : có phải giai thừa của 3 là 8 không ? Câu trả lời là : No.

Chương 4 - GIỚI THIỆU VỀ MÁY HỌC

4.1. Thế nào là máy học?

Thuật ngữ “học” theo nghĩa thông thường là **tiếp thu tri thức** để biết cách vận dụng. Quá trình học diễn ra dưới nhiều hình thức khác nhau như học thuộc lòng (học vẹt), học theo kinh nghiệm (học dựa theo trường hợp), học theo kiểu nghe nhìn... Trên máy tính cũng có nhiều thuật toán học khác nhau. Trong phạm vi của giáo trình này, chúng ta chỉ khảo sát phương pháp học dựa theo trường hợp. Theo phương pháp này, hệ thống sẽ được cung cấp một số các trường hợp “mẫu”, dựa trên tập mẫu này, hệ thống sẽ tiến hành phân tích và rút ra các quy luật (biểu diễn bằng luật sinh). Sau đó, hệ thống sẽ dựa trên các luật này để “đánh giá” các trường hợp khác (thường không giống như các trường hợp “mẫu”). Ngay cả chỉ với kiểu học này, chúng ta cũng đã có nhiều thuật toán học khác nhau. Một lần nữa, với mục đích giới thiệu, chúng ta chỉ khảo sát một trường hợp đơn giản. Có thể khái quát quá trình *học theo trường hợp* dưới dạng hình thức như sau:

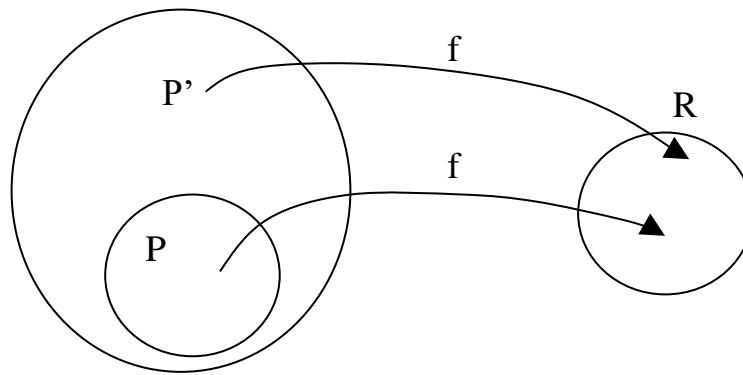
Dữ liệu cung cấp cho hệ thống là một ánh xạ f trong đó ứng một trường hợp p trong tập hợp P với một “lớp” r trong tập R .

$$f : P \mapsto R$$

$$p \rightarrow r$$

Tuy nhiên, tập P thường nhỏ (và hữu hạn) so với tập tất cả các trường hợp cần quan tâm P' ($P \subset P'$). Mục tiêu của chúng ta là xây dựng ánh xạ f' sao cho có thể ứng mọi trường hợp p' trong tập P' với một “lớp” r trong tập R . Hơn nữa, f' phải bảo toàn f , nghĩa là:

$$\text{Với mọi } p \in P \text{ thì } f(p) \equiv f'(p)$$



Hình 4.1. Học theo trường hợp là tìm cách xây dựng ánh xạ f' dựa theo ánh xạ f . f được gọi là **tập mẫu**.

Phương pháp học theo trường hợp là một phương pháp phổ biến trong cả nghiên cứu khoa học và mê tín dị đoan. Cả hai đều dựa trên các dữ liệu quan sát, thống kê để từ đó rút ra các quy luật. Tuy nhiên, khác với khoa học, mê tín dị đoan thường dựa trên tập mẫu không đặc trưng, cục bộ, thiếu cơ sở khoa học.

4.2. Học theo cây quyết định

Cho trước một CSDL dạng bảng trong đó các cột của bảng là giá trị các thuộc tính, còn các hàng của bảng là phần tử mang các thuộc tính trong các cột. Mục đích xây dựng cây định danh là xây dựng tập luật để xác định phần tử X_i có các giá trị thuộc tính $\{A_j\}$ thì có thuộc tính mục tiêu là T hay E, $X_i\{A\} \rightarrow T/F$. Để xây dựng được tập luật có nhiều phương pháp. Phương pháp đơn giản nhất là xây dựng cây định danh.

Cây định danh là cây mà các nút của cây là thuộc tính dẫn xuất, các cung của cây là giá trị thuộc tính sản xuất, còn lá của cây là các phần tử có cùng thuộc tính mục tiêu.

Bảng dữ liệu $\xrightarrow{\text{Thử}}$ Cây định danh $\xrightarrow{\text{Xây dựng}}$ Luật nhân quả

a) Thuật toán Quinlan

Cho một bảng quan sát là tập hợp các mẫu với các thuộc tính nhất định của các đối tượng nào đó. Sử dụng một độ đo để định lượng và đề ra một tiêu chuẩn

nhằm chọn lựa một thuộc tính mang tính chất “phân loại” để phân bảng này thành các bảng con nhỏ hơn sao cho từ mỗi bảng con này dễ dàng phân tích tìm ra quy luật chung (bảng đệ quy). Từ đó thiết lập cây quyết định cho thấy thứ tự các thuộc tính được xét.

b) Phương pháp phân hoạch theo Quinlan để xây dựng cây định danh

Quinlan chọn thuộc tính phân hoạch bằng cách xây dựng các vector đặc trưng cho mỗi giá trị của từng thuộc tính dẫn xuất và thuộc tính mục tiêu. Cách tính như sau:

Với mỗi thuộc tính dẫn xuất A còn có thể sử dụng để phân hoạch tính:

$$V_A(j) = (T(j,r_1), (T(j,r_2), \dots, (T(j,r_n))$$

Trong đó:

$T(j,r_i)$ = (tổng số phần tử trong phân hoạch có giá trị thuộc tính dẫn xuất A là j và có giá trị thuộc tính mục tiêu là r_i / (tổng số phần tử trong phân hoạch có giá trị thuộc tính dẫn xuất A là j)

+ A là thuộc tính dẫn xuất

+ r_1, r_2, r_n là các giá trị của thuộc tính mục tiêu.

$$\sum_j T(j, r_i) = 1$$

Một vector $V_A(j)$ được gọi là vector đơn vị nếu nó chỉ có duy nhất một thành phần có giá trị 1 và các thành phần khác có giá trị 0. Thuộc tính được chọn để phân hoạch là thuộc tính là thuộc tính có nhiều vector đơn vị nhất. Sau khi phân hoạch xong, ta xây dựng cây định danh. Căn cứ vào cây định danh, phát sinh và tối ưu tập luật.

+ Khi phân hoạch ở mỗi gốc ta tính các vector đặc trưng cho các thuộc tính dẫn xuất sau đó chọn thuộc tính dẫn xuất có nhiều vector đơn vị nhất để phân hoạch.

+ Nếu có nhiều thuộc tính dẫn xuất có tỉ số: *[Số vector đơn vị / tổng số các vector lớn hơn là thuộc tính phân hoạch]*

Bài toán áp dụng:

Hãy xây dựng cây định danh và tìm luật theo phương pháp vector đặc trưng của Quinlan để xác định một loại quả độc hay không độc theo bảng số liệu sau.

Tên	Vị	Màu	Vỏ	Độc
A	Ngọt	Đỏ	Nhẵn	Không
B	Cay	Đỏ	Nhẵn	Không
C	Chua	Vàng	Có gai	Không
D	Cay	Vàng	Có gai	Độc
E	Ngọt	Tím	Có gai	Không
F	Chua	Vàng	Nhẵn	Không
G	Ngọt	Tím	Nhẵn	Không
H	Cay	Tím	Có gai	Độc

Thuộc tính mục tiêu của bài toán là quả có độc ta chọn thuộc tính dẫn xuất để phân hoạch:

– Thuộc tính vị:

$$V_{\text{vị}}(\text{Ngọt}) = (T(\text{ngọt, độc}), T(\text{ngọt, không độc})) = (0/3, 3/3).$$

$$V_{\text{vị}}(\text{Cay}) = (T(\text{cay, độc}), T(\text{cay, không độc})) = (2/3, 1/3).$$

$$V_{\text{vị}}(\text{Chua}) = (T(\text{chua, độc}), T(\text{chua, không độc})) = (0/2, 2/2).$$

– Thuộc tính màu:

$$V_{\text{Màu}}(\text{đỏ}) = (T(\text{đỏ, độc}), T(\text{đỏ, không độc})) = (0/2, 2/2).$$

$$V_{\text{Màu}}(\text{vàng}) = (T(\text{vàng, độc}), T(\text{vàng, không độc})) = (1/3, 2/3).$$

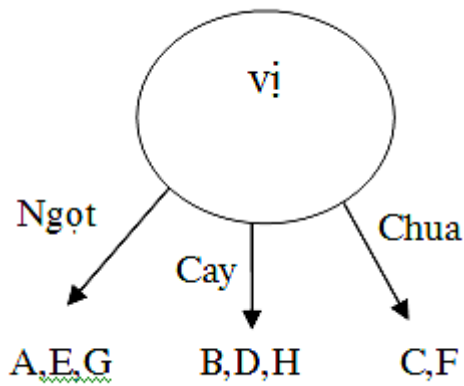
$$V_{\text{Màu}}(\text{tím}) = (T(\text{tím, độc}), T(\text{tím, không độc})) = (2/3, 1/3).$$

– Thuộc tính vỏ:

$$V_{\text{vỏ}}(\text{nhẵn}) = (T(\text{nhẵn, độc}), T(\text{nhẵn, không độc})) = (0/4, 4/4).$$

$$V_{\text{vỏ}}(\text{gai}) = (T(\text{gai, độc}), T(\text{gai, không độc})) = (3/4, 1/4).$$

Các thuộc tính vỏ, thuộc tính màu có 1 vector đơn vị, thuộc tính vị có 2 vector đơn vị vậy ta chọn thuộc tính vị là thuộc tính phân hoạch, quả gạch chân là quả có độc.



Còn lại tập P_{Cay} còn lẫn lộn quả độc và không độc, tiếp tục phân hoạch thành các tập con theo hai thuộc tính màu và vỏ:

– Thuộc tính màu:

$$V_{\text{Màu}}(\text{đỏ}) = (T(\text{đỏ, độc}), T(\text{đỏ, không độc})) = (0/1, 1/1)$$

$$V_{\text{Màu}}(\text{vàng}) = (T(\text{vàng, độc}), T(\text{vàng, không độc})) = (1/1, 0/1)$$

$$V_{\text{Màu}}(\text{tím}) = (T(\text{tím, độc}), T(\text{tím, không độc})) = (1/1, 0/1).$$

– Thuộc tính vỏ:

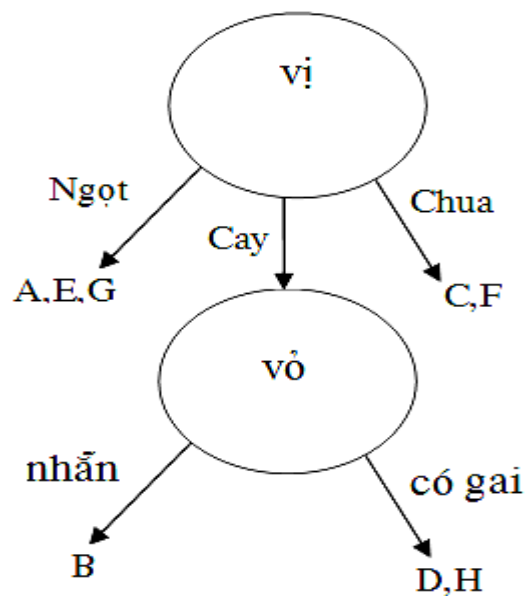
$$V_{\text{vỏ}}(\text{Có gai}) = (T(\text{Có gai, độc}), T(\text{Có gai, không độc})) = (2/2, 0/2)$$

$$V_{\text{vỏ}}(\text{Nhẵn}) = (T(\text{Nhẵn, độc}), T(\text{Nhẵn, không độc})) = (0/1, 1/1).$$

Thuộc tính vỏ có 2 vector đơn vị, vậy ta chọn thuộc tính này làm vector phân hoạch tiếp được cây định danh

Từ cây định danh ta có thể suy ra hệ luật như sau:

1. Quả có vị ngọt và quả có vị chua không độc;
2. Quả có vị cay vỏ nhẵn không độc;
3. Quả có vị cay và có gai độc.



Quá trình này tiếp tục cho đến khi tất cả các nút lá của cây không còn lẫn lộn giữa độc và không độc nữa. Ta thấy rằng qua mỗi bước phân hoạch cây phân hoạch ngày càng “phình” ra. Chính vì vậy mà quá trình này được gọi là quá trình “đâm chồi”. Cây mà chúng ta xây dựng được gọi là cây định danh.

4.3. Các ứng dụng

4.3.1. Phân loại văn bản

Phân loại văn bản là một bài toán xử lý văn bản cổ điển, đó là ánh xạ một văn bản vào một chủ đề đã biết trong một tập hữu hạn các chủ đề dựa trên ngữ nghĩa của văn bản. Ví dụ một bài viết trong một tờ báo có thể thuộc một (hoặc một vài) chủ đề

nào đó (như *thể thao, sức khỏe, công nghệ thông tin,...*). Việc tự động phân loại văn bản vào một chủ đề nào đó giúp cho việc sắp xếp, lưu trữ và truy vấn tài liệu dễ dàng hơn về sau.

Sử dụng phương pháp cây quyết định để áp dụng vào bài toán phân loại văn bản. Dựa vào tập các văn bản huấn luyện, xây dựng một cây quyết định. Cây quyết định có dạng là cây nhị phân, mỗi nút trong tương ứng với việc phân hoạch tập văn bản dựa trên một thuộc tính nào đó (một từ). Việc xây dựng cây quyết định phụ thuộc vào việc lựa chọn thuộc tính để phân hoạch.

Cây quyết định được xây dựng và dùng cho phân loại văn bản. Văn bản mới (cần được phân loại) được coi như là một tập hợp các đặc trưng (các từ). Ta sẽ tiến hành duyệt cây quyết định để gán nhãn phân loại chủ đề cho văn bản đó. Việc duyệt cây quyết định hơi giống với duyệt và tìm kiếm trên

cây nhị phân tìm kiếm:

- Nếu từ thuộc văn bản và giá trị của từ nhỏ hơn giá trị phân tách tại nút, hoặc từ không thuộc văn bản thì ta sẽ duyệt tiếp cây con trái của cây quyết định.
- Nếu từ thuộc văn bản và giá trị của từ lớn hơn giá trị phân tách tại nút thì ta sẽ duyệt cây con phải của cây quyết định.
- Quá trình này dừng khi gặp nút hiện tại là nút lá, gán nhãn cho văn bản là nhãn của nút lá đó.

4.3.2. Nhận dạng mặt người

Dựa vào diện mạo (appearance-based) phương pháp này thường dùng một mô hình máy học nên còn được gọi là phương pháp dựa trên máy học (machine learning-based). Hướng tiếp cận này áp dụng các kỹ thuật theo hướng xác suất thống kê và máy học để tìm những đặc tính liên quan của khuôn mặt và không phải là khuôn mặt. Các đặc tính đã được học ở trong hình thái các mô hình phân bố hay các hàm biệt số nên dùng có thể dùng các đặc tính này để xác định khuôn mặt người.

Đồng thời, bài toán giảm số chiều thường được quan tâm để tăng hiệu quả tính toán cũng như hiệu quả xác định. Có nhiều phương pháp áp dụng xác suất thống kê để giải quyết. Một ảnh hay một vector đặc trưng xuất phát từ một ảnh được xem như một biến ngẫu nhiên x , và biến ngẫu nhiên có đặc tính là khuôn mặt hay không phải khuôn mặt bởi công thức tính theo các hàm mật độ phân lớp theo điều kiện.

$$P(x \mid \text{khuôn mặt}) \text{ và } P(x \mid \sim \text{khuôn mặt})$$

Xác định khuôn mặt người là một kỹ thuật máy tính để xác định các vị trí và kích thước của các khuôn mặt người trong các ảnh bất kì. Kỹ thuật này nhận biết các đặc trưng của khuôn mặt và bỏ qua những thứ khác như: tòa nhà, cây cối, cơ thể ...

Việc xác định khuôn mặt người có những khó khăn nhất định như:

- Hướng của khuôn mặt đối với máy ảnh, như: nhìn thẳng, nhìn nghiêng hay nhìn từ trên xuống. Cùng trong một ảnh có thể có nhiều khuôn mặt ở những tư thế khác nhau.
- Sự có mặt của các chi tiết không phải là đặc trưng riêng của khuôn mặt người, như: râu quai nón, mắt kính,
- Các nét mặt khác nhau trên khuôn mặt, như: vui, buồn, ngạc nhiên,
- Mặt người bị che khuất bởi các đối tượng khác có trong ảnh. rất khác khi người đang cười, tức giận hay sợ hãi ...

4.3.3. Phân tích dữ liệu

Một số hệ thống máy học nỗ lực loại trừ giác khách quan của con người trong việc phân tích dữ liệu, trong khi các hệ thống khác hướng đến việc tăng sự cộng tác giữa người và máy. Không thể loại bỏ hoàn toàn tác động của con người vì các nhà thiết kế hệ thống phải chỉ định cách biểu diễn của dữ liệu và những cơ chế nào sẽ được dùng để tìm kiếm các đặc tính đặc trưng của dữ liệu. Học máy có thể được xem là một nỗ lực để tự động hóa một số phần của phương pháp khoa học.

Học máy là một phương pháp để tạo ra các chương trình máy tính bằng việc phân tích các tập dữ liệu. Học máy có liên quan lớn đến thống kê, nghiên cứu việc phân tích dữ liệu, nhưng khác với thống kê, máy học tập trung vào sự phức tạp của các giải thuật trong việc thực thi tính toán.

TÀI LIỆU THAM KHẢO

- [1] Nguyễn Thanh Thủy. *Trí tuệ nhân tạo: Các phương pháp giải quyết vấn đề và xử lý tri thức*. Nhà xuất bản Giáo dục, 1995-1999.
- [2] GS. Hoàng Kiếm. *Nhập môn trí tuệ nhân tạo*, Nhà xuất bản ĐHQG TP.HCM, 2008.
- [3] Th.S Nguyễn Việt Cường, Bài giảng Nhập môn Trí tuệ nhân tạo, HV Công nghệ Bưu Chính Viễn Thông, 2002.