

Nguyễn Thị Hải _ Lớp A _ Khoa Công Nghệ Thông Tin _ ĐHSPHN
Luận văn tốt nghiệp _ Mô phỏng thuật toán đệ quy.

TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

* * *

*

Luận văn tốt nghiệp

Đề tài: Mô phỏng thuật toán
ĐỆ QUY

Giáo viên hướng dẫn: PGS.TS Vũ Đình Hòa
Sinh viên: Nguyễn Thị Hải
Lớp A _ K54 _ Khoa Công Nghệ Thông Tin

Năm 2008

MỤC LỤC:

A. Phần 1: Phần mở đầu.

1. Lý do chọn đề tài.
2. Mục tiêu và nhiệm vụ nghiên cứu đề tài.
3. Đối tượng và phạm vi nghiên cứu.
4. Cấu trúc luận văn.

B. Phần 2: Phần nội dung.

1. Mô phỏng thuật toán:

- 1.1. Khái niệm mô phỏng thuật toán.
- 1.2. Lịch sử mô phỏng.
- 1.3. Tác dụng mô phỏng thuật toán.
- 1.4. Kiến trúc của hệ thống mô phỏng.
- 1.5. Một số khó khăn khi thực hiện mô phỏng.
- 1.6. Lựa chọn ngôn ngữ lập trình cài đặt mô phỏng.
- 1.7. Yêu cầu đạt được khi thực hiện mô phỏng.

2. Đề quy:

2.1. Đề quy là gì?

- 2.1.1. Vai trò và định nghĩa của đề quy.
- 2.1.2. Giải thuật đề quy.
- 2.1.3. Thủ tục đề quy.
- 2.1.4. Thiết kế thủ tục đề quy.

2.2. Đệ quy quay lui là gì?

2.3. Cấu trúc và đặc điểm của đệ quy.

2.3.1. Cấu trúc.

2.3.2. Đặc điểm.

2.4. Ưu nhược điểm khi thực hiện đệ quy.

2.4.1. Ưu điểm.

2.4.2. Nhược điểm.

2.5. Đệ quy nên dùng khi nào?

3. Một số bài toán thường gặp trong Đệ quy:

3.1. Bài toán tháp Hà Nội.

3.1.1. Nhận xét.

3.1.2. Phân tích.

3.1.3. Thuật giải.

3.1.4. Giải thuật.

3.1.5. Độ phức tạp thuật toán.

3.2. Bài toán 8 quân hậu.

3.2.1. Bài toán.

3.2.2. Phân tích.

3.2.3. Thuật giải.

3.2.4. Giải thuật.

3.2.5. Nhận xét.

4. Khó khăn trong khi dạy các bài toán Đệ quy:

4.1. Khó khăn chung.

4.2. Bài toán tháp Hà Nội.

4.3. Bài toán 8 quân hậu.

C. Phần 3: Phân tích và thiết kế hệ thống cho bài toán mô phỏng.

I. Lựa chọn ngôn ngữ C#.

1. Ngôn ngữ C#.

2. Đặc điểm của ngôn ngữ C#.

3. Các phương thức và các hàm thư viện.

4. Các lớp xử lý đồ họa.

5. Các bước xây dựng chương trình đồ họa.

II. Thiết kế thuật toán mô phỏng.

D. Code và giao diện chương trình.

1. Code chương trình.

2. Giao diện chương trình.

3. Sử dụng chương trình mô phỏng.

E. Kết luận.

F. Tài liệu tham khảo.

G. Nhận xét của thầy cô.

Phần 1: Phần mở đầu.

1. Lý do chọn đề tài:

Cấu trúc dữ liệu là một chương trình bao gồm các thuật toán như sắp xếp, lựa chọn, đệ quy, ngăn xếp... Mỗi thuật toán đều có một độ khó riêng, đòi hỏi khả năng hiểu rõ thuật toán thật chính xác và có sự liên tưởng thật phong phú để làm sao giúp người học hiểu thật rõ về thuật toán đó. Trong phần này tôi sẽ nghiên cứu về Đệ Quy vì để học và muốn tìm hiểu thật chắc về Đệ Quy thì bạn phải hiểu được cách nó chạy và cách nó thực thi như thế nào. Đã có rất nhiều ý kiến cho rằng học Đệ Quy khá khó và việc áp dụng nó cũng hạn chế vì nó thường hay gây tràn bộ nhớ. Nhưng ngược trở lại nó lại có một vài ứng dụng khá phổ biến trong một vài bài toán mà chỉ có dùng Đệ quy làm được.

Chính điều đó mà việc mô phỏng các thuật toán đang được chú trọng nhiều. Nhờ việc mô phỏng mà việc học một ngôn ngữ hay một thuật toán sẽ dễ dàng hơn. Giúp cho quá trình dạy và học trở nên đơn giản hơn rất nhiều. Chính vì vậy chúng tôi quyết định đi xây dựng thuật toán, cụ thể là mô phỏng thuật toán Đệ Quy.

2. Mục tiêu và nhiệm vụ nghiên cứu đề tài.

Nghiên cứu tổng quan về mô phỏng.

Đưa ra được một quy trình cho việc thiết kế mô phỏng một thuật toán và cách thức cài đặt quá trình mô phỏng.

Giúp cho việc học và hiểu về ngôn ngữ Đệ quy tốt nhất.

Nghiên cứu, phân tích những khó khăn khi học tập, giảng dạy các thuật toán cơ bản trong cấu trúc dữ liệu và một số giải thuật. Từ đó thực hiện xây dựng chương trình mô phỏng cho chúng.

- Ứng dụng chương trình mô phỏng trong giảng dạy để đánh giá và tiến tục điều chỉnh.Đưa ra một thuật giải ưu việt nhất, giúp cho quá trình hiểu bài tốt nhất.

3. Đối tượng và phạm vi nghiên cứu:

Nghiên cứu ngôn ngữ lập trình C#.

Nghiên cứu về Đệ Quy và các vấn đề liên quan.

Nghiên cứu 2 bài toán điển hình nhất về Đệ Quy.

4. Cấu trúc khoá luận.

Chia làm 4 phần:

Phần 1: Phần mở đầu:

Giới thiệu qua đề tài cấu trúc chung của đề tài.

Phần 2: Phần nội dung:

Các lý thuyết về mô phỏng thuật toán và các vấn đề liên quan đến Đệ quy.

Phần 3: Phân tích và thiết kế hệ thống cho bài toán mô phỏng Đệ Quy.

Phần 4: Code chương trình và giao diện.

Đưa ra code và đưa ra được giao diện của bài mô phỏng sau khi chương trình chạy hoàn thành.

Phần 5: Kết luận:

Tổng kết lại những phần đã đạt được, tự đánh giá.

Phần 5: Tài liệu tham khảo.

Phần 5: Lời nhận xét của thầy cô.

Phần 2 : Phần nội dung

1. **Mô phỏng thuật toán:**

1.1. Khái niệm về mô phỏng thuật toán:

Mô phỏng thuật toán là quá trình tách dữ liệu, thao tác, ngữ nghĩa và tạo mô phỏng đồ họa cho quá trình trên [Stasko 1990] (xem [23]). Mô phỏng thuật toán được thiết kế để giúp người dùng có thể hiểu thuật toán, đánh giá chương trình và sửa lỗi chương trình.

Một chương trình máy tính chứa các cấu trúc dữ liệu của thuật toán mà nó thực thi. Trong quá trình thực thi chương trình, các giá trị trong cơ sở dữ liệu được thay đổi. Mô phỏng thuật toán sử dụng biểu diễn đồ họa để biểu diễn cấu trúc dữ liệu và chỉ ra sự thay đổi giá trị trong cơ sở dữ liệu trong mỗi trạng thái. Thông qua đó, người sử dụng có thể xem được từng bước thực thi chương trình và nhờ vậy có thể hiểu chi tiết được thuật toán.

Mô phỏng thuật toán cũng được dùng để đánh giá một chương trình đã có bằng cách cung cấp các mô phỏng cho các thành phần của hệ thống, nhờ đó có thể kiểm tra được hiệu năng của hệ thống.

Bên cạnh việc giúp người sử dụng hiểu hơn về hệ thống, mô phỏng thuật toán còn được dùng để giúp thực hiện quá trình dò lỗi dễ dàng hơn. Để sử dụng mô phỏng thuật toán trong quá trình dò lỗi của một chương trình, người sử dụng chú thích vào các trạng thái của chương trình để tạo ra các lệnh mô phỏng, sau đó chúng sẽ được đưa vào hệ thống mô phỏng thuật toán để tạo mô phỏng. Người sử dụng có thể xem chương trình của họ đã thực hiện như thế nào, các giá trị dữ liệu ở mỗi bước và một bước sẽ ảnh hưởng tới các bước sau như thế nào. Nó sẽ giúp người sử dụng tìm ra tất cả các lỗi có thể xảy ra trong chương trình.

1.2. Lịch sử mô phỏng thuật toán.

Mô phỏng thuật toán đã được xây dựng từ hai thập kỷ gần đây. Nhưng chương trình mô phỏng thuật toán đầu tiên là của Ken Knowlton ở Bell Telephone Laboratories khi mô phỏng ngôn ngữ liên kết danh sách vào năm 1966. Mô phỏng thuật toán phát triển mạnh vào đầu những năm 80 của thế kỷ 20.

Vào năm 1981, video (sorting out sorting) được xây dựng bởi Ronald Baecker ở đại học Toronto được coi là khởi điểm của lĩnh vực mô phỏng thuật toán. Từ đó các nhà giáo dục đã sử dụng mô phỏng thuật toán để trợ giúp quá trình dạy học. Giữa những năm 80 và đầu những năm 90, hai hệ thống có ảnh hưởng mạnh đến sau được phát triển và có ý nghĩa lớn trên tất cả những hệ thống sau này. Hai hệ thống này là BALSA-I (Brown ALgorithm Simulator and Animator)

[Brown 1984] và TANGO (Transition-based Animation GeneratiOn) [Stasko 1990].

BALSA-I là hệ thống mô phỏng thuật toán nổi tiếng rộng khắp đầu tiên. Nó được phát triển bởi Marc Brown và Robert Sedgewick tại trường đại học Brown. BALSA-I là hệ thống mô phỏng thuật toán tương tác mà hỗ trợ đồng thời nhiều cái nhìn của một cấu trúc dữ liệu thuật toán và có thể hiển thị nhiều thuật toán thực thi đồng thời. Sự phát triển của nó là động cơ thúc đẩy các nhà nghiên cứu khác tham gia vào việc phát triển các hệ thống mô phỏng thuật toán khác nữa.

Một hệ thống khác là TANGO, được phát triển bởi John Stasko của trường đại học Brown. Sự nổi bật của TANGO là chỉ ra mô hình path-transition để thiết kế mô phỏng và một framework cho hệ thống mô phỏng thuật toán. Nó đưa ra một khái niệm framework mới mà được chấp nhận bởi một số hệ thống sau này như kiến trúc cơ sở của chúng. Kiến trúc này sẽ được mô tả trong mục tiếp theo.

Từ khi hai hệ thống của BALSA và TANGO được phát triển, các hệ thống đi sau của hai hệ thống đáng chú ý này cũng được phát triển. BALSA-I có một hệ thống đi sau đó là BALSA-II [Brown 1988]. BALSA-II là một hệ thống mô phỏng thuật toán vùng-độc lập thao tác các ảnh với nhiều cái nhìn và cung cấp quá trình tạo ra bộ điều khiển dễ dàng. TANGO thì khác, có nhiều hệ thống đi sau. XTANGO [Stasko 1992] là hệ thống trực tiếp đi sau TANGO. POLKA được thiết kế để xây dựng mô phỏng đồng thời cho các chương trình song song. Nó là một hệ thống mô phỏng thuật toán

hướng đối tượng 2-D và được mở rộng thành hệ thống 3-D, POLKA 3-D. POLKA 3-D cung cấp cái nhìn 3-D và 3-D nguyên thủy, ví dụ như: hình nón, hình cầu, hình lập phương và một số hình khác nữa. Người dùng không bị yêu cầu phải có hiểu biết trước về đồ họa máy tính 3-D để sử dụng POLKA 3-D. Samba cho phép thể hiện mô phỏng tương tác mà đọc các câu lệnh ASCII và thực hiện các hành động mô phỏng tương ứng. Có một phiên bản Java của Samba được gọi là JSamba (xem <http://www.cc.gatech.due/gvu/softviz/paviz/samba.html>).

Các hệ thống mô phỏng thuật toán khác bao gồm: Zeus, Leonardo, CATAI, Mocha. Zeus [Brown 1991] được phát triển tại trường đại học Brown cùng với BALSA và BALSA-II, nó được coi như một trong số các hệ thống phần mềm có ảnh hưởng lớn đến nhau đầu tiên. Zeus được thực thi trong môi trường multi-threaded và multi-processor, vì thế nó có thể làm cho các chương trình song song. CATAI (xem <http://wonderland.dia.unisa.it/catai/>) là một hệ thống mô phỏng các chương trình C++. Nó tin tưởng vào những công nghệ đối tượng phân tán và cho phép một vài người dùng chia sẻ mô phỏng đó thông qua sự trừu tượng hóa lớp học thực tế. Truyền thông và sự đồng bộ hóa giữa các khách hàng mô phỏng và thuật toán được mô phỏng được đảm bảo bởi người phục vụ mô phỏng Java mà sử dụng công nghệ CORBA. Mocha (xem <http://www.cs.brown.edu/people/jib/Mocha.html>) là một mô hình phân tán với kiến trúc client-server nhằm tối ưu phân chia những

thành phần của phần mềm trong một hệ thống mô phỏng thuật toán tiêu biểu. Trong mô hình Mocha, chỉ mã giao diện được xuất tới máy người dùng, trong khi thuật toán được thực hiện trên một server chạy trên máy của nhà cung cấp.

Với việc phát triển của công nghệ mới, tính phổ dụng của mạng toàn cầu và sự tiến hóa của ngôn ngữ lập trình Java, những người phát triển đã xây dựng những hệ thống mô phỏng thuật toán trực tuyến, có lợi thế của những hệ thống mở dễ tiếp cận hơn.

Một số nhà phát triển cũng hợp nhất việc sử dụng đa phương tiện trong các hệ thống của họ. Việc sử dụng các hệ thống mô phỏng thuật toán không còn bị bó hẹp trong các lớp học truyền thống hoặc phòng thí nghiệm giảng dạy nữa mà đã được mở rộng để dạy từ xa.

Trong khoảng hai thập niên gần đây, một số rất lớn các hệ thống mô phỏng thuật toán đã ra đời và phát triển mạnh mẽ. Phần lớn các hệ thống mô phỏng thuật toán đã đề cập trong mục này đều phổ biến hơn và phức tạp hơn các hệ thống đang được sử dụng trong thực tế. Chúng đã được phát triển và sử dụng bởi những nhà chuyên môn, với mục đích giáo dục hoặc nghiên cứu thực nghiệm của họ. Một trong số các hệ thống này có một kiến trúc phức tạp và cần những công nghệ đặc biệt để chạy nó. Chúng ta không có bất kỳ tiện ích nào của các hệ thống này để xây dựng hệ thống mô phỏng các thuật toán đồ thị; thay vào đó, chúng ta đã ước lượng được các hệ thống mô phỏng hiện hữu khác mà kích thước nhỏ hơn và có những kiến trúc đơn giản hơn.

1.3. Tác dụng của mô phỏng thuật toán.

Các hệ thống mô phỏng thuật toán được sử dụng rộng rãi như công cụ hỗ trợ giảng dạy trong ngành giáo dục khoa học máy tính. Một số nghiên cứu thực nghiệm đã ước lượng hiệu quả của chúng trong giáo dục và kết quả nhận được có thay đổi. Cụ thể là:

Brown (1984) đã sử dụng BALSA-I để dạy một khóa giới thiệu lập trình và một khóa “cấu trúc dữ liệu và giải thuật”. Hệ thống được sử dụng như một chương trình trực quan trong khóa giới thiệu, và như một người mô phỏng thuật toán mức cao trong lớp cấu trúc dữ liệu. Ông ta báo cáo rằng việc sử dụng các hoạt cảnh mô phỏng để phụ thêm vào thuyết trình dẫn tới ‘những lợi ích có thể chứng minh được trong việc tăng tốc độ hiểu biết’ qua thuyết trình truyền thống. Stasko (1997) đã sử dụng Samba, chương trình mô phỏng của hệ thống XTango dạy một khóa thuật toán khoa học máy tính. Những sinh viên được yêu cầu sử dụng hệ thống có thêm vào mô phỏng cho các chương trình ấn định của họ. Các kết quả thu được cho biết rằng những sinh viên thích các mô phỏng và những mô phỏng đó có thể làm tăng tính sáng tạo của các sinh viên. Hơn nữa, sự hiểu biết của sinh viên về thuật toán được tăng lên nhờ việc mô phỏng.

Tuy nhiên, sử dụng thuật toán trong việc dạy học không phải lúc nào cũng thành công. Các nhà giáo dục đã làm các thực nghiệm và thu được các kết quả pha trộn. Stasko et al. (1993) đã chỉ ra một thí nghiệm bằng việc dạy hai nhóm sinh viên với hai cách thuyết trình khác nhau. Cả hai nhóm sinh viên này cùng nghiên cứu thuật toán “

Pairing heap” (ghép đôi đống). Một nhóm học thuật toán dựa vào sự mô tả văn bản và nhóm kia cũng nhận các tài liệu đó nhưng có thêm sự trợ giúp bằng các chương trình mô phỏng thuật toán. Mặc dù những kết quả chỉ ra rằng nhóm thứ hai đạt được nhiều điểm hơn nhóm kia, nhưng không có điểm nổi trội nào có thể được kết luận là nhờ sự trợ giúp của mô phỏng.

Tương tự, Byrne et al. (1996) đã chủ đạo hai thí nghiệm mà trong đó các kết quả chỉ ra rằng lợi ích của mô phỏng không phải là hiển nhiên. Những kết quả pha trộn này đã gây ra chán nản, nhưng đa số các nhà giáo dục đều tin tưởng rằng mô phỏng hỗ trợ việc học.

Tuy nhiên, những kết quả thí nghiệm bất lợi này gợi ý những yếu tố quan trọng khác trong việc sử dụng mô phỏng thuật toán. Các kết quả đã thông báo rằng để đạt được hiệu quả mô phỏng thuật toán đầy đủ thì điều quan trọng là mô phỏng được sử dụng phối hợp với những yếu tố khác. Lawrence et al. (1994) đã sử dụng các hệ thống XTANGO và POLKA để dạy thuật toán cây khung nhỏ nhất Kruskal. Trong số nhóm sinh viên tham dự các thí nghiệm, kết quả của những sinh viên mà tham dự một phiên thí nghiệm tương tác tốt hơn đáng kể so với những sinh viên mà tham dự những phiên thí nghiệm bị động. Các kết quả này đã cho phép các sinh viên điều khiển và tương tác với mô phỏng tốt hơn, chẳng hạn, chương trình mô phỏng cho phép sinh viên đưa vào tập dữ liệu của chính họ và thực hiện mô phỏng trên tập dữ liệu này chứ không chỉ dùng lại ở việc quan sát những tập dữ liệu mẫu.

Hơn nữa, nhiều nghiên cứu gần đây bởi Kehoe et al. (1999) cho thấy có thể sử dụng mô phỏng như một công cụ giáo dục. Thí nghiệm được thực hiện trong một thái độ khác từ các thí nghiệm khác. Những sinh viên được chia thành hai nhóm và cả hai nhóm đều học thuật toán “binomial heap” (đồng nhị thức). Một nhóm học thuật toán bởi sự tương tác với mô phỏng trong khi nhóm còn lại là đọc những hình dạng phẳng về các điểm khóa thao tác của thuật toán. Sự khác nhau trong thí nghiệm này là kịch bản bài tập về nhà. Những sinh viên được đưa cho những câu hỏi trước khi bắt đầu khóa học. Trong suốt thời gian kiểm tra thử, những sinh viên có thể truy cập tới bài dạy và thời gian để hoàn thành bài kiểm tra thử này được cho tương đối nhiều. Các kết quả của thí nghiệm này cho thấy nhóm được trang bị chương trình mô phỏng thuật toán thực hiện bài kiểm tra thử tốt hơn nhóm kia. Các sinh viên của nhóm có sử dụng mô phỏng thuật toán phản hồi rằng mô phỏng đã giúp đỡ họ hiểu thuật toán tốt hơn.

Báo cáo của Kehoe et al (1999) đã trình diễn một cách sử dụng mô phỏng thuật toán trong việc dạy để đạt được giá trị sư phạm cao hơn. Nó đã được thuyết trình rằng mô phỏng thuật toán được sử dụng tốt hơn trong các tình trạng học tương tác và mô phỏng (như một bài tập về nhà). Cũng như vậy, mô phỏng thuật toán có thể có tính sư phạm hơn khi nó được sử dụng trong việc phối hợp với các cách học khác hoặc giúp đỡ những chỉ dẫn khác để giải thích làm thế nào thực hiện một thao tác của thuật toán. Báo cáo cũng nói rằng với mô phỏng thuật toán người ta có thể dễ dàng học các thao tác theo thủ tục của

các thuật toán. Ngoài ra nó có thể làm cho việc học một thuật toán bớt đáng sợ hơn vì nó làm cho thuật toán dễ tiếp cận hơn.

Stasko et al. (1993) đã kết luận từ thí nghiệm của họ một số điều kiện mà mô phỏng thuật toán có thể có lợi nhất. Một trong số những điều kiện này là hỗ trợ mô phỏng thuật toán với những chỉ dẫn thúc đẩy toàn diện. Khi mô phỏng thuật toán đóng vai trò chỉ dẫn này, màn hình mô phỏng phải được bổ sung bởi các mô tả văn bản của các thao tác đang diễn ra. Một điều kiện khác đó là hệ thống mô phỏng thuật toán cần phải bao gồm các chức năng: quay lại hoặc lặp lại những bước thực hiện thuật toán để cho phép những người dùng sao lưu và xem lại những thao tác quan trọng. Một số bài giảng đòi hỏi các trạng thái thực hiện thuật toán cũng cần phải được ghi lại và cung cấp lại được. Sự phản hồi của sinh viên cũng là quý giá trong việc cải thiện chất lượng chỉ dẫn của mô phỏng.

Mặc dù những kết quả được đưa ra từ những nghiên cứu thực nghiệm này không phải luôn có lợi, thì cũng không có nghĩa rằng mô phỏng thuật toán không hiệu quả trong dạy học. Hiện nay đang có nhiều nghiên cứu đang được tiến hành về thiết kế và đánh giá mô phỏng thuật toán. Hansen et al. (1999) tin rằng các kết quả trong các nghiên cứu thực nghiệm trên chưa tốt không phải vì mô phỏng thuật toán là phương pháp dạy học không tốt, mà vì cách thức thực hiện các mô phỏng chưa tốt. Họ đã phát triển một hệ thống trực quan hóa giải thuật siêu phương tiện gọi là HalVis (Hypermedia Algorithm Visualizations). Dựa vào framework của chúng, họ đã thiết kế các

trực quan hóa giải thuật, và họ đã hướng dẫn vài thí nghiệm thực nghiệm bởi việc sử dụng hệ thống này. Tất cả các kết quả thí nghiệm cho thấy trực quan hóa giải thuật bằng đồ họa có hiệu quả hơn so với các phương pháp dạy truyền thống. Những kết quả này cho thấy rằng để mô phỏng thuật toán có hiệu quả và có lợi cho người dùng, thì việc thiết kế cho thích hợp và cách thức mô phỏng là những yếu tố quan trọng.

1.4. Kiến trúc của hệ thống mô phỏng.

Đa số các hệ thống mô phỏng thuật toán có những thư viện hỗ trợ thủ tục mô phỏng và giao diện mô phỏng. Vài hệ thống mô phỏng đòi hỏi phải đưa vào trực tiếp bằng tay những thông điệp gửi tới các thủ tục mô phỏng trong chương trình thực hiện thuật toán. Những hệ thống mô phỏng thuật toán ra đời sớm như: BALSA and TAGO là sự kiện – điều khiển (event-driven), nghĩa là chúng có một chương trình phát sinh những sự kiện trong dạng những thông điệp tới một máy chủ thông điệp. Máy chủ thông điệp chuyển thông điệp tới những cảnh quan tương ứng. Một cảnh quan là một cửa sổ trong một thiết bị màn hình nơi người dùng nhìn những đối tượng mô phỏng. Thông điệp bao gồm thông tin của một đối tượng mô phỏng. Sau khi cảnh quan nhận thông điệp, nó tính toán lại đối tượng và kéo lại nó trên cảnh quan.

Vài hệ thống gần đây được viết bằng Java và tất cả đều có những kiến trúc tương tự nhau. Ví dụ như: JSamba, hệ thống POLKA tiền tiêu (xem <http://www.cc.>

gatech.due/gvu/softviz/parviz/samba.html) và JAWAA (Java và mô phỏng thuật toán trên mạng, xem <http://www.cs.duke.edu/~rodger/tools>), phát triển bởi Pierson và Rodger tại trường đại học Duke vào năm 1996. Những hệ thống này chấp nhận framework của TANGO như kiến trúc của nó. Tất cả các hệ thống sẽ gồm có 3 thành phần, các hàm mô phỏng (animator), kênh mô phỏng (animation interpreter) và trình diễn mô phỏng (animation viewer) như đã chỉ ra trong sơ đồ sau:



Hình 1. Kiến trúc của hệ thống mô phỏng
thuật toán

- Các hàm mô phỏng: Chứa các thư viện để vẽ các đối tượng mô phỏng trên thiết bị màn hình.
- Màn hình trình diễn mô phỏng: Cung cấp một môi trường đồ họa để trình diễn mô phỏng trên thiết bị màn hình tới người dùng cuối.
- Kênh mô phỏng: Đóng vai trò như một kênh truyền thông giữa hệ thống mô phỏng và người dùng cuối. Nó đọc một file kịch bản ASCII được cung cấp bởi người dùng cuối mà trong đó có chứa mô phỏng văn bản cung cấp việc phát sinh những lệnh.
- Kênh mô phỏng dịch các lệnh kịch bản thành các lệnh mô phỏng tương ứng và chuyển qua những tham số điều khiển của đối tượng mô phỏng tới các hàm mô phỏng.

- Các hàm mô phỏng vẽ đối tượng được mô phỏng theo các tham số điều khiển của đối tượng đó tới Animation viewer.
- Các tham số điều khiển bao gồm tọa độ x và y chỉ rõ nơi đối tượng được mô phỏng xuất hiện trong Animation viewer hoặc màu sắc của đối tượng được mô phỏng.

Người thiết kế mô phỏng cần tạo ra một file văn bản ASCII gồm có mọi lệnh để tạo ra mô phỏng. Những câu giải có thể là một dòng lệnh được đặt vào trong bất kỳ đoạn code nào trong chương trình mà ở đó dữ liệu hoặc cấu trúc dữ liệu sắp được mô phỏng. Các lệnh mô phỏng chuyển đầu vào là văn bản rõ thành đầu ra là một file văn bản ASCII. File kịch bản sau đó được chuyển qua hệ thống tới mô phỏng được tạo ra bởi quá trình thực hiện của người lập trình cuối.

1.5. Một số khó khăn khi thực hiện mô phỏng.

- Chưa thật sự nắm chắc về thuật toán mà ta cần mô phỏng.
- Chưa hiểu rõ về quá trình thực thi của thuật toán khi áp dụng vào một bài toán cụ thể.
- Từ lý thuyết đến thiết kế mô phỏng trên một ngôn ngữ lập trình mà chưa nắm rõ về ngôn ngữ đó thì sẽ gặp nhiều khó khăn trong khi mô phỏng.
- Đệ quy vốn là một ngôn ngữ khó diễn đạt và khó hiểu nên khi mô phỏng thuật toán Đệ Quy ta khó diễn đạt.

1.6. Lựa chọn ngôn ngữ lập trình cài đặt mô phỏng.

1.6.1 Ngôn ngữ lập trình là gì?

Ngôn ngữ lập trình(tiếng anh là Programming language) là một tập con của ngôn ngữ máy tính. Đây là dạng ngôn ngữ chuẩn được chuẩn hóa để miêu tả những tính toán qua máy tính trong một dạng mà cả con người và máy tính đều có thể thay đổi được.

Một ngôn ngữ lập trình phải thoả mãn 2 đặc điểm sau:

- + Nó phải dễ hiểu và dễ sử dụng đối với người lập trình để con người có thể dùng nó để giải quyết các bài toán.
- + Nó phải miêu tả một cách đầy đủ và rõ ràng các tiến trình(process) để có thể chạy được trên máy tính khác.

Chữ **lập trình** dùng để chỉ thao tác của con người nhằm kiến tạo nên các chương trình máy tính thông qua các ngôn ngữ lập trình. Người ta còn gọi quá trình lập trình đó là **quá trình mã hóa** thông tin tự nhiên thành ngôn ngữ máy. Trong các trường hợp xác định thì chữ lập trình còn được viết là "viết mã" (cho chương trình máy tính).

Như vậy, theo định nghĩa, mỗi ngôn ngữ lập trình cũng chính là một chương trình, nhưng có thể được dùng để tạo nên các chương trình khác. Một chương trình máy tính được viết bằng một ngôn ngữ lập trình thì những chỉ thị (của riêng ngôn ngữ ấy) góp phần tạo nên chương trình được gọi là **mã nguồn** của chương trình ấy.

1.7. Yêu cầu cần đạt được khi thực hiện mô phỏng thuật toán.

1.7.1. Mô tả theo đúng thuật toán:

Thuật toán được đưa ra mô phỏng phải chính xác, các bước thực hiện thuật toán phải trực quan và phản ánh đúng theo nội dung thuật toán đã đưa ra để đảm bảo tính đúng đắn của thuật toán.

Để kiểm tra tính đúng đắn của thuật toán, ta có thể cài đặt giải thuật đó trên máy tính rồi đưa vào các bộ dữ liệu xác định, lấy kết quả thu được xác định với kết quả đã biết. Bộ dữ liệu đưa vào phải đảm bảo kết quả thu được phải vét kín các trường hợp nghiêm của bài toán (trường hợp thông thường và các trường hợp đặc biệt). Làm theo cách này thì không chắc chắn, ta chỉ phát hiện được thuật toán sai chứ không khẳng định được luôn đúng. Tính đúng đắn chỉ có th

1.7.2. Hệ thống mô phỏng phải thực hiện từng bước:

Thuật toán thường là trìu tượng, nếu để chương trình chạy tự động thì người dùng sẽ khó hiểu. Vì vậy, cần phải có chế độ thực hiện mô phỏng thuật toán theo từng bước, để người học có thể quan sát, theo dõi sự thay đổi giá trị của từng biến. Nhờ đó, sẽ giúp cho người học hiểu thuật toán rõ hơn và nhanh hơn.

1.7.3. Mô phỏng thuật toán phải có tính động:

Để mô tả trực quan hóa quá trình thực hiện của thuật toán ta nên đưa vào hình ảnh động (có thể có âm thanh) để thể hiện sự thay đổi của dữ liệu trong quá trình thực thi. Ví dụ như, trong thuật toán tìm đường đi ngắn nhất trong đồ thị - thuật toán Dijkstra, ta đưa vào các đối tượng đồ họa là đỉnh, cạnh. Trong

quá trình thực thi thuật toán thì các đỉnh/cạnh được duyệt sẽ sáng nhấp nháy và đổi màu, sau khi được chọn thì các đỉnh/cạnh sẽ được tô màu khác với màu ban đầu. Tương ứng với quan sát đường đi trên đồ thị, ta có thể đưa vào đoạn thuật toán thể hiện tương ứng và song song với quá trình duyệt trên đồ thị. Nhờ đó mà thuật toán được thể hiện một cách rõ nét, sinh động, trực quan. Giúp người đọc dễ theo dõi, dễ hiểu thuật toán hơn.

1.7.4. Thuật toán phải được thử nghiệm trong mọi trường hợp phải đảm bảo thực thi tốt.

Một thuật toán được mô phỏng phải đảm bảo là thuật toán tốt, dễ hiểu và đúng đắn. Muốn vậy ta phải thử nghiệm trong các trường hợp dữ liệu ngẫu nhiên, tốt nhất, xấu nhất. Nếu thuật toán vẫn chạy tốt và trong một thời gian cho phép thì thuật toán mới hiệu quả. Ta không thể chấp nhận một thuật toán đúng mà thời gian chạy quá lớn.

1.7.5. Tạo được sự phân cấp cho người học:

Đối tượng học thuật toán thường là các sinh viên. Họ có trình độ tiếp thu khác nhau, nên ta phải đưa ra nhiều chế độ thao tác khác nhau để người học được phép lựa chọn.

2. Đệ quy:

2.1. Đệ quy là gì?

2.1.1. Vai trò và định nghĩa của đệ quy:

+ Vài trò:

Có vai khá quan trọng trong một số bài toán phức tạp mà

việc dùng các thuật giải khác lại trở nên phức tạp hơn.

Có những bài toán dùng đệ quy thì trở nên thuận lợi hơn nhiều so với lời giải lặp và có những giải thuật đệ quy thực sự cũng có hiệu quả cao hơn nữa, chẳng hạn giải thuật sắp xếp kiểu phân đoạn (Quicksort).

Một điều nữa: Về mặt định nghĩa thì công cụ đệ quy đã cho phép xác định một tập hợp các vô hạn các đối tượng bằng một phát biểu hữu hạn.

+ Định nghĩa:

Đệ quy (tiếng Anh; Recursion) là một phương pháp dùng trong các chương trình máy tính trong đó có một hàm tự gọi.

Một đối tượng là đệ quy nếu nó bao gồm chính nó như một bộ phận hoặc nó được định nghĩa dưới dạng chính nó.

Ví dụ:

```
Procedure Giaithua(n:word):integer;  
begin  
if n=0 then giaithua:=1  
else giaithua:=n*giaithua(n-1);  
end;
```

2.1.2. Giải thuật đệ quy:

Nếu bài toán T được thực hiện bằng lời giải của một bài toán T' khác có dạng giống T thì đó là lời giải đệ quy hay là giải thuật đệ quy.

Ví dụ xét bài toán tìm một từ trong một quyển sách từ điển thì ta có thuật giải như sau:

if từ điển là một trang.

then tìm từ trong trang này

else begin

 Mở từ điển vào trang "giữa";

 Xác định xem nửa nào của từ điển chứa từ cần
tìm;

if từ đó nằm ở nửa trước của từ điển.

then tìm từ đó trong nửa trước

else tìm từ đó trong nửa sau.

end;

Tất nhiên giải thuật trên mới chỉ được nêu dưới dạng "thô" và
còn nhiều chỗ chưa cụ thể, chẳng hạn:

- Tìm từ trong một trang thì làm thế nào

- Thế nào là mở từ điển vào trang giữa

- Làm thế nào để biết từ đó nằm ở nửa nào của từ điển...

Để trả lời rõ những câu hỏi trên không phải là khó, nhưng ta sẽ
không sa vào các chi tiết này mà muốn tập trung vào việc xét
"chiến thuật" lời giải. Có thể hình dung chiến thuật tìm kiếm này
một cách khái quát như sau:

Ta thấy có hai điểm chính cần lưu ý:

1. Sau mỗi lần từ điển được tách đôi thì một nửa thích hợp sẽ lại
được tìm kiếm bằng một "chiến thuật" như đã dùng trước đó.

2. Có một trường hợp đặc biệt, khác với mọi trường hợp trước,
sẽ đạt được sau nhiều lần tách đôi, đó là trường hợp tự điển chỉ
còn duy nhất một trang. Lúc đó việc tách đôi ngừng lại và bài toán

trở thành đủ nhỏ để ta có thể giải quyết trực tiếp bằng cách tìm từ mong muốn trên trang đó chẵng hạn, bằng cách tìm tuần tự.

Trường hợp đặc biệt này được gọi là trường hợp suy biến.

Có thể coi đây là một "chiến thuật" kiểu "chia để trị". Bài toán được tách thành bài toán nhỏ hơn và bài toán nhỏ hơn lại được giải quyết với thuật chia để trị như trước, cho tới khi xuất hiện trường hợp suy biến.

2.1.3. Thủ tục đệ quy:

Một thủ tục gọi là đệ quy nếu trong quá trình thực hiện nó phải gọi đến chính nó nhưng với kích thước nhỏ hơn của tham số.

Như bài ví dụ cụ thể trên ta thấy:

Có thể coi đây là một "chiến thuật" kiểu "chia để trị". Bài toán được tách thành bài toán nhỏ hơn và bài toán nhỏ hơn lại được giải quyết với thuật chia để trị như trước, cho tới khi xuất hiện trường hợp suy biến.

Ta thể hiện giải thuật tìm kiếm này dưới dạng một thủ tục:

Procedure TIMKiem (TD,Tu)

{TD được coi là đầu mối để truy nhập được vào tự điển đang xét,
Tu chỉ từ cần tìm}

1. if Tự điển chỉ còn là một trang

then Tìm từ Tu trong trang này

else begin

2. Mở tự điển vào trang giữa

Xác định xem nửa nào của tự điển chứa từ Tu

```
if Tu nằm ở nửa trước của từ điển  
    then call TIMKIEM (TD 1,Tu)  
    else call TIMKIEM (TD 2,Tu)  
end;
```

{TD 1 và TD 2 là đầu mối để truy nhập được vào nửa trước và nửa sau của từ điển}

3. Return

Thủ tục như trên được gọi là thủ tục đệ quy.

2.1.4. Thiết kế giải thuật đệ quy:

Khi bài toán đang xét hoặc dữ liệu đang xử lý được định nghĩa dưới dạng đệ quy thì việc thiết kế các giải thuật đệ quy tỏ ra rất thuận lợi. Hầu như nó phản ánh rất sát nội dung của định nghĩa đó. Các bạn có thể thấy điều này qua bài toán sau:

* Bài toán Dãy số FIBONACCI

Dãy số Fibonacci bắt nguồn từ bài toán cổ về việc sinh sản của các cặp thỏ. Bài toán được đặt ra như sau:

- 1) Các con thỏ không bao giờ chết.
- 2) Hai tháng sau khi ra đời một cặp thỏ mới sẽ sinh ra một cặp thỏ con (một đực và một cái).
- 3) Khi đã sinh con rồi thì cứ mỗi tháng tiếp theo chúng lại sinh được một cặp con mới.

Giả sử bắt đầu từ một cặp mới ra đời thì đến tháng thứ n sẽ có bao nhiêu cặp?

Ví dụ: n=6, ta thấy:

Tháng thứ 1: 1 cặp (cặp ban đầu)

Tháng thứ 2: 1 cặp (cặp ban đầu vẫn chưa đẻ)

Tháng thứ 3: 2 cặp (đã có thêm 1 cặp con)

Tháng thứ 4: 3 cặp (cặp đầu vẫn đẻ thêm)

Tháng thứ 5: 5 cặp (cặp con bắt đầu đẻ)

Tháng thứ 6: 8 cặp (cặp con vẫn đẻ tiếp)

Bây giờ ta xét tới việc tính số cặp thỏ ở tháng thứ n: $F(n)$

- Nếu mỗi cặp thỏ ở tháng thứ $(n-1)$ đều sinh con thì $F(n) = 2(n-1)$.

Nhưng không phải như vậy. Trong các cặp thỏ ở tháng thứ $(n-1)$ chỉ có những cặp đã có ở tháng thứ $(n-2)$ mới sinh con ở tháng thứ n được thôi.

Do đó: $F(n) = F(n-2) + F(n-1)$

Vì vậy có thể tính $F(n)$ theo:

$$F(n) = \begin{cases} 1 \text{ nếu } n \leq 2 \\ F(n-2) + F(n-1) \text{ nếu } n > 2. \end{cases}$$

Dãy số thể hiện $F(n)$ ứng với các giá trị của $n= 1, 2, 3, \dots$ có dạng:

1 1 2 3 5 8 13 21 34 55 ...

được gọi là dãy số Fibonacci. Dãy số Fibonacci còn là mô hình của rất nhiều hiện tượng tự nhiên và cũng được sử dụng nhiều trong tin học. Sau đây là thủ tục đệ quy thể hiện giải thuật tính $F(n)$:
Function $F(n)$ 1. if $n \leq 2$ then $F:=1$ else $F:=F(n-2) + F(n-1)$ 2.

Return ở đây chỉ có một chi tiết hơi khác là trường hợp suy biến ứng với hai giá trị $F(1) = 1$ và $F(2)=1$.

2.2. Đệ quy quay lui là gì?

Thuật toán quay lui dùng để giải các bài toán liệt kê cấu hình. Mỗi cấu hình được xây dựng bằng cách xây dựng từng phần tử, mỗi phần tử được chọn bằng cách thử tất cả các khả năng.

Giả thiết cấu hình cần liệt kê có dạng (x_1, x_2, \dots, x_n) . Khi đó thuật toán quay lui thực hiện theo các bước như sau:

- Xét tất cả các giá trị x_1 có thể nhận, thử cho x_1 nhận lần lượt các giá trị đó. Với mỗi giá trị thử gán cho x_1 ta có.
- Xét lần lượt các giá trị x_2 có thể nhận, lại thử x_2 nhận lần lượt các giá trị đó. Với mỗi giá trị thử gán cho x_2 lại xét tiếp các khả năng chọn $x_3 \dots$ Cứ như vậy làm tiếp..
- Xét tất cả các giá trị của x_n có thể nhận, cho x_n nhận lần lượt các giá trị đó, thông báo cấu hình tìm được (x_1, x_2, \dots, x_n) .

Trên phương diện quy nạp có thể nói rằng thuật toán quay lui liệt kê các cấu hình n phần tử dạng (x_1, x_2, \dots, x_n) bằng cách thử cho x_1 nhận các giá trị có thể. Với mỗi giá trị thử gán cho x_1 lại liệt kê tiếp cấu hình $n - 1$ phần tử (x_2, \dots, x_n) .

Mô hình thuật toán quay lui thường mô tả bằng thủ tục TRY:

Procedure Try (i : integer);

begin

for (mỗi giá trị V có thể cho x_i) do

begin

<Thử cho $x_i := V$ >;

```
if < $x_i$  là phần tử cuối cùng trong câu hình> then
    <thông báo câu hình tìm được>
else
    begin
        <ghi nhận việc  $x_i$  nhận giá trị  $V$ >;
        Try( $i + 1$ );
        <Nếu cần, bỏ ghi nhận việc thứ  $x_i := V$ , để thử
        giá trị khác>;
    end;
end;
end;
```

Đệ quy quay lui có một số bài toán điển hình như liệt kê các dãy nhị phân độ dài n, liệt kê cá tập con k phần tử hay bài toán phân tích số.Nhưng điển hình nhất đó là bài toán xếp các quân hậu trên một bàn cờ(ta thường xét vị trí của 8 quân hậu).

2.3. Cấu trúc và đặc điểm của đệ quy.

2.3.1. Cấu trúc: Gồm 2 phần

- *Phần cơ sở*: chứa các tác động của hàm hoặc thủ tục với một số giá trị cụ thể ban đầu của tham số.
- *Phần đệ quy*: định nghĩa tác động cần được thực hiện cho giá trị hiện thời của các tham số bằng các tác động đã được định nghĩa trước đây với kích thước tham số nhỏ hơn.

2.3.2. Đặc điểm:

Trong thủ tục đệ quy có lời gọi đến chính nó.

Mỗi lần có lời gọi lại thủ tục thì kích thước của bài toán thu nhỏ hơn trước.

Trường hợp đặc biệt là xảy ra trường hợp suy biến thì bài toán sẽ dùng đệ quy để kết thúc chương trình. Ở đây có sử dụng thuật toán “chia để trị”.

Mỗi một lần hàm tự gọi đệ quy đến nó thì máy tính sẽ tự tạo ra một biến cục bộ mới.

Có bao nhiêu lần hàm gọi đệ quy thì sẽ có bấy nhiêu lần thoát ra khỏi hàm. (Kiểu như lặp hàm).

Khi thoát ra ngoài hàm đệ quy thì một loạt các biến cục bộ tạo ra do dùng đệ quy lúc này mới được giải phóng, và chúng sẽ giải phóng trước các biến cục bộ (sinh ra do đệ quy) tạo ra sau.

Sử dụng đệ quy là một phương pháp làm cho chương trình ngắn gọn, dễ hiểu nhưng nó sẽ làm tốn bộ nhớ và thời gian nếu như cấu trúc hàm đệ quy “phức tạp”.

2.4. Ưu nhược điểm khi sử dụng đệ quy.

2.4.1. Ưu điểm:

- + Đệ quy mạnh ở chỗ có thể định nghĩa một tập hợp rất lớn các tác động bởi một số hữu hạn các mệnh đề.

- + Làm cho chương trình ngắn gọn, trong sáng, dễ hiểu nỗi bật được bản chất của vấn đề.

2.4.2. Nhược điểm:

- + Làm tốn bộ nhớ nếu như hàm đệ quy quá phức tạp.

+ Nếu bài toán không suy biến thì sử dụng đệ quy không hợp lý, làm cho bài toán phức tạp lên.

2.5. Đệ quy nên dùng khi nào?

Chỉ sử dụng hàm đệ quy khi bài toán xảy ra trường hợp suy biến.

Trong một trường hợp tổng quát bài toán có thể đưa về cùng dạng. Nhưng giá trị thay đổi, sau một loạt thay đổi nó phải đưa về trường hợp suy biến.

3. Một số bài toán thường gặp trong Đệ quy:

Thực tế có rất nhiều bài toán có sử dụng giải thuật đệ quy như bài toán tính giai thừa của $n!$ hay bài toán tính giá trị của dãy Fibonacci.. Nhưng chúng ta sẽ chỉ đi sâu vào một vài bài toán điển hình nhất trong đó đã đưa ra được bản chất nổi bật nhất của đệ quy.
Đó là:

3.1. Bài toán Tháp Hà Nội.

3.1.1. Nhận xét:

Bài toán này mang tính chất một trò chơi có nội dung:

Có n đĩa, kích thước nhỏ dần, đĩa có lỗ ở giữa (như đĩa hát).

Có thể chồng chúng lên nhau xuyên qua một cái cọc, to dưới nhỏ trên để cuối cùng có một chồng đĩa dạng như hình một chiếc tháp.

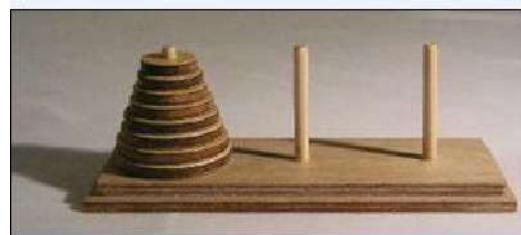
3.1.2. Phân tích:

Giả sử ta gọi các cọc là A, B, C.

Chuyển chồng đĩa từ cọc A sang cọc khác, chẳng hạn sang

cọc C, theo những quy tắc sau:

- + Mỗi lần chỉ được chuyển một đĩa.
- + Không được xảy ra trường hợp đĩa to ở trên đĩa nhỏ dù chỉ là tạm thời.
- + Được phép sử dụng cọc trung gian, chẳng hạn đây là cọc B để tạm đặt đĩa (gọi là đĩa trung gian) khi chuyển từ cọc A sang cọc C.



3.1.3. Thuật giải: Ta sẽ thử tìm hiểu xem trong Pascal thì bài toán này được giải theo hướng ra sao để từ đó áp dụng vào phần thiết kế.

- + Trường hợp có 1 đĩa:
 - Chuyển từ cọc A sang cọc C.
- + Trường hợp có 2 đĩa:
 - Chuyển từ đĩa thứ nhất từ cọc A sang cọc B.
 - Chuyển từ đĩa thứ hai từ cọc A sang cọc C.
 - Chuyển từ đĩa thứ nhất từ cọc B sang cọc C.
- + ...
- + Trường hợp có n đĩa ($n > 2$) và nếu coi (n-1) đĩa ở trên, đóng vai trò như đĩa thứ nhất thì có thể xử lý tương tự như 2 trường hợp trên. Nghĩa là:
 - Chuyển (n-1) đĩa từ cọc A sang cọc B.

- Chuyển đĩa thứ n cọc A sang cọc C.
- Chuyển (n-1) đĩa từ cọc B sang cọc C.

Như vậy bài toán trên được chia ra làm các bài toán con và hướng giải các bài toán này như nhau. Cụ thể với n đĩa thì kết quả chỉ là chuyển 2 đĩa bất kỳ và ta gọi nó là trường hợp suy biến.

3.1.4. Giải thuật:

Ta sử dụng thủ tục đệ quy: Với những gì bài toán yêu cầu và hướng giải quyết như trên thì việc dùng giải thuật đệ quy là hợp lý nhất. Ta có giải thuật đệ quy sau:

Procedure dich_chuyen (n, A, B, C);

1- **if** n=1 **then** chuyển đĩa từ A sang C

2- **else begin**

 calldich_chuyen(n-1, A, C, B);

 calldich_chuyen(1, A, B, C);

 calldich_chuyen(n-1, B, A, C)

end

3- **return**

Ngoài ra ta có thể giải bằng phương pháp biểu diễn nhị phân:

Các vị trí đĩa có thể xác định được trực tiếp từ biểu diễn nhị phân của số thứ tự di chuyển (cơ số 2 với một chữ số cho mỗi đĩa) trong đó các dãy 1 và các dãy 0 tượng trưng cho các dãy các đĩa liền nhau trên cùng cọc, và mỗi khi chữ số có thay đổi thì đĩa kế tiếp sẽ đổi sang trái hay phải một cọc (hay chuyển sang cọc ngoài cùng phía đối diện). Chữ số ở đầu đại diện cho đĩa lớn nhất và nếu là chữ số 0 thì có nghĩa là đĩa lớn nhất không đổi khỏi cọc xuất phát và ngược lại. Đặt các chữ số 1 và 0 luân phiên bên dưới các chữ số của một

bước chuyển cho phép biết được di chuyển theo một chiều khi nó hợp với chữ số của bước chuyển tại nơi chữ số thay đổi và theo chiều kia khi nó không hợp. Do đó bước chuyển 00000000... có nghĩa là đặt 8 đĩa lớn nhất lên cọc ban đầu, bước chuyển 11111111... có nghĩa là đặt chúng lên cọc cuối cùng, và bước chuyển 11011000... có hai đĩa lớn nhất trên cọc đích, đĩa tiếp theo trên cọc xuất phát, hai đĩa tiếp theo ở cọc trung gian, và ba đĩa tiếp theo nữa trên cọc xuất phát, bắt kể có thêm bao nhiêu chữ số đại diện các đĩa nhỏ hơn. Ta có thể dễ dàng tính được các vị trí của các đĩa trong một bộ tám mươi đĩa sau một số các bước tiến, nếu giới hạn đủ lớn để chứa nó. Việc dùng phương pháp đệ quy cho trường hợp tám mươi đĩa như thế này có thể không thực tế.

3.1.5. Độ phức tạp của thuật toán:

Ta xét xem n đĩa thì số lần di chuyển đĩa sẽ bao nhiêu?

Giả sử gọi dichuyen(n) là số đó thì ta có: dichuyen(1) = 1;

Khi n tăng lên ($n > 1$) thì ta sẽ tính được dichuyen(n) như sau:

$$\text{Dichuyen}(n) = \text{dichuyen}(n-1) + \text{dichuyen}(1) + \text{dichuyen}(n-1)$$

Vậy ta đã xác định được mối quan hệ truy hồi của cách tính:

$$\text{Dichuyen}(1) = 1$$

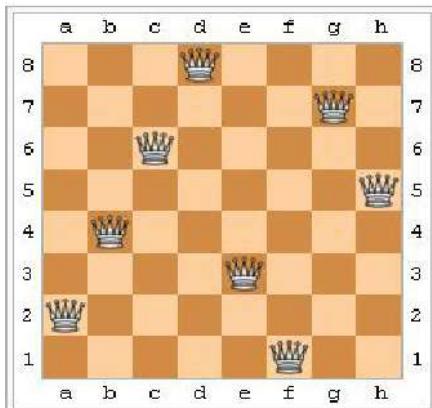
$$\text{Dichuyen}(n) = 2 * \text{dichuyen}(n - 1) + 1, \text{ nếu } n > 1$$

Vậy $\text{dichuyen}(n) = 2^n - 1$ là độ đánh giá giải thuật của bài toán tháp Hà Nội.

Ứng dụng:

Dùng trong các bài toán dạy các ngôn ngữ lập trình cơ bản hay trong nghiên cứu tâm lí cách giải quyết vấn đề.

3.2. Bài toán bàn cờ về các quân hậu.



3.2.1 Bài toán:

Xét bàn cờ hình vuông 8 hàng 8 cột. Quân hậu là một quân cờ có thể ăn được bất kỳ quân nào nằm trên cùng một hàng, cùng một cột hay cùng một đường chéo.

Bài toán đặt ra: Hãy xếp 8 quân hậu trên bàn cờ sao cho không có quân hậu nào ăn được quân hậu nào, có nghĩa trên mỗi hàng mỗi cột chỉ có thể có một quân hậu mà thôi.

3.2.2. Phân tích:

Ta không nên tìm lời giải cho bài toán bằng cách xét từng trường hợp với mọi vị trí của 8 quân hậu trên bàn cờ rồi lọc các trường hợp chấp nhận được. Phương pháp thử từng bước này tuy không hay lăm nhưng lại có thể đưa ra được tất cả các cách sắp xếp vị trí cho các quân hậu.

Phương pháp này được gọi là thuật toán đệ quy quay lui. Nó được áp dụng trong cách giải bài toán 8 quân hậu như sau:

Do mỗi cột chỉ có một quân hậu nên lựa chọn đối với quân hậu thứ j, ứng với cột j, là đặt nó vào hàng nào để đảm bảo “an toàn” nghĩa là không cùng hàng, cùng đường chéo với (j-1) quân

hậu đã được xếp trước đó. Vậy để đi đến các lời giải ta phải thử tất cả các trường hợp sắp xếp quân hậu đầu tiên tại cột 1. Với mỗi vị trí như vậy ta lại phải giải quyết bài toán 7 quân hậu với phần còn lại của bàn cờ, nghĩa là ta đã “quay lui bài toán cũ”.

3.2.3. Thuật giải:

Ta dùng một thủ tục là TRY để minh họa cho giải pháp giải bài này. Cụ thể:

Procedure TRY(j)

B1: Khởi phát việc chọn vị trí cho quân hậu thứ j.

B2: repeat thực hiện việc chọn tiếp theo

If an toàn then begin

 đặt quân hậu;

 if $j < 8$ then begin

 call TRY (j+1)

 if không thành công.

 then cất quân hậu

end

end

until thành công hay hết chõ

B3: Return

Cụ thể bài toán được phân tích như sau:

Đối với quân hậu thứ j, vị trí của nó chỉ chọn trong cột thứ j. Vậy tham biến j trở thành chỉ số cột và việc chọn lựa để tiến hành trên 8 giá trị của chỉ số hàng i.

Để lựa chọn i được chấp nhận, thì hàng i và 2 đường chéo ô (i,j) phải không có quân hậu nào ở trên đó.

Chú ý là trong hai đường chéo thì đường chéo theo chiều đi lên có các ô (i,j) mà tổng $i + j$ không đổi, còn đường chéo theo đường đi xuống có các ô (i,j) mà $i - j$ không đổi.

Do đó ta sẽ chọn các mảng một chiều Boolean để biểu diễn các tình trạng này là:

$a[i] = \text{true}$ có nghĩa là không có quân hậu nào chứa hàng i.

$b[i + j] = \text{true}$ có nghĩa không có quân hậu nào chiếm được đường chéo $i + j$.

$c[i - j] = \text{true}$ có nghĩa là không có quân hậu nào chiếm được đường chéo $i - j$.

Điều kiện: $1 \leq i, j \leq 8$ nên suy ra $1 \leq j \leq 8$

Vậy: $2 \leq i + j \leq 16$ và $-7 \leq i - j \leq 7$

Như vậy điều kiện để lựa chọn i được chấp nhận là $a[i]$ and $b[i + j]$ and $c[i - j]$ có giá trị true.

Quân hậu được đặt theo nguyên tắc :

$x[j] = i$; $a[i] := \text{false}$; $b[i+j] := \text{false}$; $c[i - j] := \text{false}$;

Quân hậu được cất theo nguyên tắc:

$a[i] = \text{true}$; $b[i + j] := \text{true}$; $c[i - j] := \text{true}$;

Ở đây thuật toán quay lui được sử dụng để kiểm soát sự tiến lùi của quân hậu.

3.2.4. Giải thuật:

Đầu tiên ta xây dựng một thủ tục TRY:

Procedure TRY (j, q)

1. $i := 0;$
2. repeat $i := i + 1$; $q := \text{false};$
 If $a[i]$ and $b[i + j]$ and $c[i - j]$ then begin
 $x[j] := i;$
 $a[i] := \text{false};$
 $b[i + j] := \text{false};$
 $c[i - j] := \text{false};$
 if $j < 8$ then begin
 call TRY ($j + 1, q$)
 if not q then begin
 $a[i] := \text{true};$
 $b[i + j] := \text{true};$
 $c[i - j] := \text{true};$
 end
 end
 else $q := \text{true};$
 end
until $q \vee (i = 8)$

3. return

Với thủ tục TRY trên ta sẽ có một giải thuật cho lời giải bài toán 8 con hậu theo ngôn ngữ Pascal như sau:

Program TAMHAU;

1. {Khởi tạo tình trạng ban đầu}

 For $i := 1$ to 8 do $a[i] := \text{true};$

 For $i := 2$ to 16 do $b[i] := \text{true};$

For i:=-7 to 7 do c[i] := true;

2. {Tìm một lời giải}

Call TRY (1, q);

3. {In kết quả}

If q then for i:= 1 do 8 write (x[i]);

end

3.2.5. Nhận xét:

Bài toán là một mô hình thuật toán điển hình của giải thuật đệ quy quay lui.Cụ thể:

Nét đặc trưng để giải bài này là ở mỗi lời giải là một bước thử. Nếu có một bước thử được chấp nhận thì ghi nhớ các thông tin cần thiết và tiến hành bước thử tiếp theo. Nếu trái lại không có một lựa chọn nào thích hợp thì làm lại bước trước, xoá bớt các ghi nhớ và quay về các lựa chọn còn lại.Hoạt động trên là quay lui.

Ứng dụng: Dùng để kiểm soát hoạt động của các quân hậu trên bàn cờ.

4. **Những khó khăn trong khi dạy các ví dụ về Đệ Quy:**

4.1. Khó khăn chung:

Đệ quy là một giải thuật khá khó so với các ngôn ngữ lập trình khác cả về lý thuyết và sự ứng dụng của nó trong thực tế.

Thực chất chính học sinh còn khá mơ hồ khi tiếp cận với đệ quy nên mỗi khi giáo viên càng đi sâu vào một vấn đề khó hơn thì càng làm cho việc truyền đạt khó khăn hơn.

Có ít tài liệu mô phỏng thực tế về sự hoạt động của đê quy. Giáo viên cũng có ít thời gian để cho học sinh đi tìm hiểu kỹ do thời gian chương trình phân phối danh cho tìm hiểu đê quy ít.

4.2. Bài toán tháp Hà Nội:

- a. Học sinh chưa nắm rõ về yêu cầu cũng như kết quả thu được mà bài toán yêu cầu.
- b. Mô phỏng bước chuyển khi số đĩa ít từ 2 tăng dần cũng đã làm cho học sinh hơi khó hiểu. Vì quá trình số đĩa chuyển càng nhiều thì càng làm cho học sinh rối.
- c. Học sinh chưa phân biệt rõ các vai trò các đĩa như thế nào? Khi nào một đĩa là đĩa chung gian.
- d. Giải thích các mô tả bằng tay về sự di chuyển của các đĩa rất khó khi mà chính học sinh còn chưa hiểu được khi số đĩa chỉ là nhỏ nhất.
- e. Giải thích về thuật giải của bài toán này cho học sinh hiểu là khá khó do khó khi giải thích sự hoạt động của đê quy có sử dụng trong bài toán này.

4.3. Bài toán 8 quân hậu:

- a. Khó khăn trong cách dạy về đê quy quay lui, học sinh không biết quay lui là gì?
- b. Bài toán này có khá nhiều lời giải nên giáo viên cần tìm một cách giải hợp lý nhất mà học sinh lại dễ hiểu.
- c. Môi trường mô phỏng còn hạn chế do chưa có nhiều đê tài nghiên cứu về vấn đề này.

Phần 3 : Phân tích và thiết kế hệ thống cho bài toán mô phỏng Đệ Quy.

I. Lựa chọn ngôn ngữ lập trình _ Csharp:

1. Ngôn ngữ Csharp(C#):

Ngôn ngữ Csharp(C#) là ngôn ngữ được thiết kế bởi Anders Hejlsberg của Microsoft. C# là ngôn ngữ lập trình hướng đối tượng hiện đại có thể tương tác với các ngôn ngữ lập trình cơ bản khác như C, C++, Pascal, Java hay cả Basic. C# là bộ sản phẩm của .Net.

C# có trình biên dịch là môi trường Framework Software Development Kit (SDK).

2. Đặc điểm của ngôn ngữ C#:

C# là ngôn ngữ dẫn xuất từ ngôn ngữ C và C++, nhưng nó được tạo từ nền tảng phát triển hơn và có tính hiện đại tiên tiến hơn cả về các chức năng chính và giao diện cho người dùng. Các đặc tính nổi bật của C# bao gồm:

2.1. C# là ngôn ngữ đơn giản:

C# là ngôn ngữ đơn giản vì nó gần gũi với ngôn ngữ C và C++ về cú pháp, diện mạo, biểu thức, toán tử,... Nhưng nó được cải thiện làm cho đơn giản hơn. Các câu lệnh hay việc thiết kế một form cũng đơn giản cho người bắt đầu làm quen với thiết kế.

2.2. C# là ngôn ngữ hiện đại:

Tính hiện đại ở đây thể hiện ở khả năng tương thích phong phú, có các đặc tính về kiểm soát đón bắt ngoại lệ (checked và

unchecked), về các khả năng kiểm tra tràn số hay không (overflow)? khả năng đón bắt lỗi với try và catch, khả năng đón bắt ngoại lệ...

2.3. C# là ngôn ngữ hướng đối tượng:

Nó sử dụng các phương thức, các lớp và các từ khoá để định nghĩa dữ liệu của từng đối tượng. Đặc trưng nhất ở đây là tính kế thừa tức là ta có thể xây dựng một lớp khác dựa trên lớp khác, chỉ cần thay đổi ứng dụng này chứ không cần thay đổi mã nguồn bên trong của đối tượng. Ngoài ra nó còn có sự đóng gói và sự đa hình.

2.4. C# là ngôn ngữ mạnh mẽ và mềm dẻo:

Khả năng hỗ trợ của ngôn ngữ này rất rộng trong nhiều công việc khác nhau từ dễ đến khó. Có sử dụng các ràng buộc trong các liên kết dữ liệu.

2.5. C# là ngôn ngữ có ít từ khoá:

Các ngôn ngữ khác có sử dụng các từ khoá khá nhiều nhưng C# lại sử dụng rất hạn chế các từ khoá, chủ yếu dùng trong trường hợp mô tả thông tin.

2.6. C# là ngôn ngữ hướng modul:

C# có sử dụng các lớp(class), cấu trúc (structures) và phương thức static. Ngoài ra có phương thức private và public...Những lớp và phương thức có thể sử dụng lại cho nhau trong các ứng dụng của đối tượng.

2.7. C# sẽ trở nên phổ biến:

Là một ngôn ngữ lập trình hướng đối tượng, được phát triển trên nền tảng tiên tiến của các ngôn ngữ lập trình cơ bản khác và tích hợp được nhiều tính năng mà các ngôn ngữ khác chưa có.

C# ngày càng trở nên phổ biến do tính hỗ trợ rõ rệt của nó khi lập trình, giao diện đẹp và dễ sử dụng. Khả năng tương tác với các ngôn ngữ khác khá cao.

3. Các phương thức, các hàm thư viện cơ bản:

- Phương thức tĩnh Static: Đây là khái niệm vô cùng quan trọng của C#. Khi được gọi thì phát biểu khởi đầu là tên của lớp.
- Phương thức Onpaint: Là phương thức kế thừa của form trong khi thiết kế đồ họa cho form.
- Phương thức Drawstring: Là phương thức dùng trong vẽ đồ thị, đưa ra một xâu cụ thể trong font.
- Lớp thư viện:

Được tổ chức thành các không gian tên, mỗi không gian tên là một nhóm cục bộ bao gồm các nhóm và được cài đặt như một DLL.

Bao gồm: class, struct, interface, enumeration, delegate...

4. Các lớp xử lý đồ họa và các phương thức trong thiết kế form:

Ở đây chúng ta chỉ tìm hiểu về vẽ đồ họa hướng đối tượng gọi là đồ họa Vector.

- Xây dựng lớp đồ họa:

Khai báo một đối tượng trong lớp Graphics

Graphics grfx = new Graphics;

- Lớp Pen và Brush: Là bút để vẽ và chổi tô.

Pen pen = new Pen (color);

Brush brush = new SolidBrush (clr);

- Phương thức DrawLine, DrawRectangles...: Dùng để vẽ các đường, các hình như tròn, đa giác...

Void DrawLine (Pen pen , PointF point1, PointF point2);

- Phương thức Fill... :Dùng để vẽ màu cho đường, cho hình hoạ...

Void FillRectangle (Brush brush, Rectangle [] rect);

Void FillRectangle (Brush brush, Rectangle [] rectf);

- Phương thức DrawString: Dùng để vẽ một xâu:

Void DrawString (string str, Font font, Brush brush,

PointF ptf);

- Phương thức DrawImage:

Graphic g = this.CreateGraphic();

g.DrawImage (<tên Bitmap>, 1, 1);

5. Các bước xây dựng một chương trình đồ họa:

Ban đầu chúng ta cần cài bộ cài Visual Studio .NET để bắt đầu thực hiện việc thiết kế.

Chúng ta sẽ viết các chương trình hiển thị ở chế độ đồ họa với chương trình trên Window Application. Chúng ta đang nghiên cứu về đồ họa trong .NET vì thế chúng ta sẽ thực hiện các bước xây dựng một ứng dụng trên Windows với Visual Studio .NET. Sau đây là các bước tạo một ứng dụng Window.

- ✓ B1: Chạy Visual Studio .NET
- ✓ B2: Tạo ra một dự án mới :
 - Chọn File/ New/ Project.
 - Trong mục Project Types chọn Visual C# project.
 - Trong Templates chọn Windows Application .Viết tên và đường dẫn rồi nhấn OK.Thấy một form hiện
- ✓ B3: Thiết kế giao diện trên form:

Có 2 cách thiết kế:

- C1: Bạn thiết kế trực tiếp trên form với việc sử dụng các control như các textbox,button,label... và sử dụng các thuộc tính trong bảng Properties bên phải màn hình thiết kế.

Để lấy được bảng thuộc tính này ta kích chuột phải vào form rồi kích Properties.

Cách này tuy nhanh nhưng không thực hiện được các chức năng hay các sự kiện của form do bảng thuộc tính không hỗ trợ.

- C2: Thiết kế bằng cách viết Code(View Code).

Một số sự kiện thiết kế trên form không thực hiện được vì vậy bạn phải viết sự kiện cho nó.

Bạn kích chuột phải vào form và chọn View Code và bắt đầu viết các sự kiện cho form.

Cách này tuy hơi chậm nhưng ta có thể thực hiện hoàn chỉnh tạo một form bằng việc viết Code.

- ✓ B4: Dịch và chạy chương trình:

Nhấn Ctrl + F5 để dịch chương trình và sửa các lỗi trên form.

Nhấn F5 để chạy chương trình.

II. Thiết kế thuật toán mô phỏng Đệ Quy:

1. Tổng quan về hệ thống:
2. Bài toán tháp Hà Nội:
3. Bài toán 8 quân hậu:
4. Giải mã hệ thống:
5. Khó khăn và thuận lợi khi thiết kế từng thuật toán:

Phần 4: Code chương trình và giao diện minh họa.

1. Code của chương trình:
2. Giao diện chương trình.
3. Hướng dẫn sử dụng mô phỏng thuật toán Đệ Quy.

Phần 5: Kết luận.

1. Kết quả đạt được:
2. Những vấn đề chưa làm được:
3. Một số hướng đề xuất phát triển cho bài toán.

Phần 6 : Tài liệu tham khảo.

1. Tài liệu sách:

- *Cấu trúc dữ liệu và giải thuật _ Đỗ Xuân Lôi.*
- *Tài liệu bồi dưỡng đội tuyển Olympic Tin _ Lê Minh Hoàng.*

- *Kỹ thuật lập trình ứng dụng C#.net toàn tập _ Phạm Hữu Khang, Phương Lan _ NXBLĐXH.*
- *Ngôn ngữ lập trình Pascal _ Quách Tuấn Ngọc.*

2. Một vài trang web liên quan:

- <http://www.cshap-home.com>

Phần 7 : Nhận xét của thầy (cô) giáo.

.....

TÂM KẾT

Đây chỉ là một phần hoàn chỉnh của đề tài. Tuy còn nhiều thiếu xót nhiều mong các thầy cô cho thêm sự góp ý để tôi có thể hoàn thành tốt phần này và các phần demo sau này.

Tuy chỉ là những bước nền tảng cho việc mô phỏng một thuật toán xong tôi cũng đã cố gắng rất nhiều dù biết còn thiếu xót nhiều.

Báo cáo khoa học lần này là cơ hội tốt để tôi có thể kiểm tra được những gì mình đã làm được và chưa làm được. Rất mong sự góp ý của thầy cô.