ỨNG DỤNG THUẬT TOÁN QUAY LUI TRONG GIẢI BÀI TOÁN BẬC THCS TRÊN NGÔN NGỮ LẬP TRÌNH C++

Nguyễn Lê Thông
Trường THCS Diễn Hồng, xã Đức Châu, Nghệ An

Tóm tắt: *Thuật toán quay lui (Backtracking) là một kỹ thuật giải bài toán bằng cách tìm kiếm tất cả các khả năng có thể xảy ra theo từng bước, và loại bỏ (quay lui) những lựa chọn không dẫn đến kết quả đúng. Nghiên cứu này tập trung vào việc ứng dụng thuật toán Quay lui (Backtracking) trong lập trình C++ để bồi dưỡng học sinh giỏi (HSG) môn Tin học cấp THCS. Thực trạng cho thấy nhiều HS gặp khó khăn trong việc tìm ra giải pháp tối ưu, khiến chương trình vượt quá giới hạn thời gian xử lý khi đối mặt với dữ liệu lớn. Mục tiêu chính là giúp HS hiểu rõ khái niệm, bản chất của thuật toán Quay lui và biết cách thiết kế, cài đặt một cách chính xác. Thuật toán Quay lui là một phương pháp vét cạn thông minh, xây dựng bộ nghiệm từng bước và "quay lui" khi không tìm được lời giải, từ đó hạn chế được các nhánh không cần thiết.*

Từ khoá: *Backtracking, HS, HSG, THCS.*

APPLICATION OF THE BACKTRACKING ALGORITHM IN SOLVING JUNIOR HIGH SCHOOL-LEVEL PROBLEMS USING THE C++ PROGRAMMING LANGUAGE

Abstract: *The Backtracking algorithm is a problem-solving technique that involves searching for all possible solutions step-by-step, and discarding (backtracking from) choices that do not lead to the correct result. This study focuses on the application of the Backtracking algorithm in C++ programming to cultivate gifted students (Học sinh giỏi - HSG) in Informatics at the Junior High School (THCS) level. Current practice indicates that many students struggle to find optimal solutions, causing their programs to exceed the time limit when processing large datasets. The main objective is to help students thoroughly understand the concept and the nature of the Backtracking algorithm, and to learn how to design and implement it accurately. The Backtracking algorithm is an intelligent brute-force method that constructs a solution set incrementally and "backtracks" when a solution cannot be found, thereby limiting unnecessary search branches.*

Keywords: *Backtracking, Students, Gifted Students (HSG), Junior High School (THCS).*

Nhận bài: 06/10/2025 *Phản biện: 05/11/2025* *Duyệt đăng: 09/11/2025*

I. ĐẶT VẤN ĐỀ

Bộ môn Tin học, với những đóng góp to lớn đã tạo nên sự thay đổi của nền giáo dục Việt Nam. Bên cạnh đó các cuộc thi HS giỏi môn Tin học dần được chú trọng và đầu tư nhiều hơn, với sự tham gia của nhiều HS tài năng. Kỳ thi HS giỏi môn Tin học là cơ hội quý báu để các HS không chỉ thể hiện sự thông minh của mình mà còn phát triển khả năng tư duy và kỹ năng giải quyết bài toán thông qua các thuật toán phức tạp. Đặc biệt là áp dụng thuật toán quay lui để giải các bài toán từ cơ bản đến nâng cao. Bên cạnh đó, để bồi dưỡng cho các HS giỏi tham gia các cuộc thi lập trình thì GV sẽ yêu cầu HS sử dụng một NNLT cụ thể. Một trong những ngôn ngữ lập trình thường được dùng một cách hiệu quả để viết chương trình giải quyết bài toán là C++.

Với những lý do đó cùng với thực tiễn dạy học bồi dưỡng HS giỏi thì việc ứng dụng thuật toán quay lui trong giải bài toán trên ngôn ngữ lập trình C++ nhằm giúp các GV và các em HS có cái nhìn đúng đắn, hiểu rõ hơn và áp dụng thuật toán đúng cách để giải quyết các bài toán một cách hiệu quả.

II. NỘI DUNG NGHIÊN CỨU

2.1. Mục đích nghiên cứu

Mục đích chính của bài viết này là nghiên cứu, phân tích và ứng dụng thuật toán Quay lui dành cho HS giỏi cấp THCS nhằm mục đích giúp HS:

Hiểu rõ và mô tả chính xác khái niệm, bản chất và mục đích của thuật toán Quay lui. khơi dậy niềm đam mê khám phá thuật toán của HS.

Biết cách trình bày, thiết kế và cài đặt một cách chính xác thuật toán Quay lui thông qua các bài tập ví dụ cụ thể.

Nâng cao kết quả học tập môn Tin học, đặc biệt đối với các HS có năng khiếu, giúp các em định hướng đúng đắn năng lực bản thân và đạt thành tích cao trong các kỳ thi.

Cung cấp một nguồn tài liệu tham khảo về thuật toán bổ ích và chi tiết, hỗ trợ cho HS và GV giảng dạy môn Tin học ở bậc THCS.

Áp dụng NNLT C++ theo Chương trình giáo dục phổ thông 2018, giúp các em HS làm quen và thành thạo với công cụ này.

2.2. Thực trạng trước khi nghiên cứu

Trong quá trình bồi dưỡng học sinh giỏi môn Tin học lớp 9, trước khi áp dụng thuật toán Quay

lui, nhiều HS thường gặp khó khăn trong việc lựa chọn thuật toán phù hợp. Mặc dù có thể đạt được kết quả chính xác với những bài toán đơn giản, nhưng khi đối mặt với các bộ dữ liệu lớn, thời gian xử lý của chương trình thường vượt quá giới hạn cho phép (thường là dưới 1 giây cho mỗi bộ dữ liệu). HS thường sử dụng các thuật toán quen thuộc mà không dành thời gian phân tích để tìm ra cách tiếp cận tối ưu hơn. Điều này dẫn đến việc chương trình không chỉ tốn thời gian mà còn sử dụng nhiều bộ nhớ hơn so với yêu cầu.

Thực tế này đặt ra vấn đề: Làm thế nào để HS có thể lập trình bằng những giải pháp hiệu quả, đáp ứng yêu cầu cả về thời gian lẫn bộ nhớ khi làm việc với các bài toán có dữ liệu lớn.

2.3. Các biện pháp sử dụng để giải quyết vấn đề

2.3.1. Thế nào là phương pháp Quay lui

Người đầu tiên đề ra thuật ngữ quay lui (backtrack) là nhà toán học người Mỹ D.H. Lehmer vào những năm 1950. Thuật toán quay lui là một thuật toán nâng cao hơn được xây dựng dựa trên thuật toán đệ qui. Đây là một thuật toán giải quyết bài toán bằng cách thử tất cả các lựa chọn có thể, và nếu lựa chọn đó dẫn đến giải pháp không phù hợp, thuật toán sẽ quay lui để thử các lựa chọn khác. Thuật toán này cũng là một phương pháp nhằm giải quyết các bài toán với mục tiêu tối ưu hoá độ phức tạp hoặc ứng dụng trong các bài toán tìm kiếm hoặc liệt kê tất cả các giải pháp có thể.

2.3.2. Bài toán Quay lui vét cạn tổng hợp

Bài toán dạng quay lui vét cạn thường được phát biểu một cách tổng quát như sau: Cần tìm một cấu hình được mô tả bởi một bộ phận gồm n thành phần $X = \{x1, x2, \dots, xn\}$. Mỗi thành phần xi $(i = 1 \dots n)$ có m khả năng chọn từ một tập $S = \{s1, s2, \dots, sn\}$ thỏa mãn một điều kiện nào đó.

Với bài toán trên, cần tìm những thành phần sau:

$X = \{x1, x2, \dots, xn\}$, được gọi là vector nghiệm.

$S = \{s1, s2, \dots, sn\}$, được gọi là tập ứng cử viên.

Điều kiện ràng buộc của vector nghiệm.

Tùy theo từng bài toán mà vector nghiệm có thể là nghiệm của bài toán hoặc có thể là một vector cấu hình để từ đó có thể tìm ra nghiệm của bài toán. Đối với điều kiện ràng buộc của vector nghiệm cũng tùy theo từng bài toán mà chúng có thể khác nhau. Điều kiện ràng buộc của vector nghiệm có thể liên quan đến từng thành phần trong vector nghiệm hoặc liên quan đến toàn thể cấu hình vector nghiệm.

2.3.3.Ý tưởng của phương pháp Quay lui

Trong các kỹ thuật cơ bản để thiết kế thuật toán, quay lui là một trong những kỹ thuật quan trọng nhất vì nó cho phép giải một lớp các bài toán khá lớn có dạng tổng quát như sau:

Tìm một (hoặc tất cả) bộ nghiệm ($x1, x2, \dots, xn$) thỏa mãn điều kiện F nào đó, trong đó các thành phần xi được chọn từ một tập hữu hạn Di với mọi $i = 1, 2, \dots, n$.

Tư tưởng của phương pháp quay lui như sau :

- Ta xây dựng bộ nghiệm từng bước, bắt đầu từ thành phần nghiệm $x1$ được chọn trong các giá trị có thể $S1 = D1$.

- Giả sử đã chọn được các thành phần $x1, x2, \dots, xi-1$, từ các điều kiện của bài toán ta sẽ xác định được tập Si gồm các giá trị được chọn cho thành phần nghiệm xi. Tập Si là tập con của Di và phụ thuộc vào các thành phần $x1, x2, \dots, xi-1$ đã chọn. Chọn một phần tử

xi thuộc Si như một thành phần của bộ nghiệm. Từ bộ $x1, x2, \dots, xi$) lặp lại quá trình trên để tiếp tục mở rộng nghiệm cho thành phần xi+1. Nếu không chọn được thành phần nào của xi+1. (do Si+1.rỗng) thì ta quay lại chọn một phần tử khác của Si làm thành phần nghiệm xi (ý nghĩa của quay lui là ở bước này).

- Trong quá trình mở rộng nghiệm ta luôn kiểm tra nghiệm thành phần có phải là nghiệm của bài toán hay không. Nếu chỉ cần một nghiệm thì ta dừng lại, nếu cần tìm tất cả các nghiệm thì quá trình tìm nghiệm chỉ dừng khi tất cả các khả năng chọn các thành phần nghiệm đã vét cạn.

Quay lui là một trong các phương pháp vét cạn trong đó các giá trị nghiệm được chọn không thực hiện được bằng cách duyệt tuần tự. Điểm tốt của thuật toán quay lui so với vét cạn tuần tự là hạn chế bớt các nhánh phải duyệt mà theo những nhánh đó không tìm được lời giải thể hiện ở việc xây dựng các tập giá trị có thể Si khi tìm thành phần nghiệm xi và quay lui khi không mở rộng thành phần nghiệm xi+1.

Hạn chế của phương pháp này là phải duyệt qua nhiều khả năng nên độ phức tạp của chương trình thường ở mức giai thừa hay hàm mũ nên tốc độ tính toán khá lâu trong trường hợp kích thước của dữ liệu vào khá lớn. Để khắc phục hạn chế này người ta tìm cách hạn chế các khả năng không đưa đến kết quả bằng phương pháp nhánh cận, tuy nhiên lớp bài toán dùng được phương pháp này không nhiều.

Để giải bài toán bằng thuật toán quay lui cần thực hiện các bước sau:

Tìm cách biểu diễn nghiệm của bài toán dưới dạng một dãy các đối tượng được chọn dần từng bước $(x1, x2, \dots, xn)$. Xác định được tập Si các khả năng được chọn làm thành phần thứ i của nghiệm. Chọn cách thích hợp để biểu diễn Si.

Tìm các điều kiện để các nghiệm đã chọn là các nghiệm của bài toán.

* Một số lưu ý khi xây dựng thuật toán quay lui:

- Phải duyệt qua mọi phương án của bài toán có thể chứa nghiệm (vét cạn).

- Tránh trường hợp duyệt trùng lặp các khả năng đã duyệt.

Từ ý tưởng của thuật toán quay lui đã nêu trên ta đưa ra được mã giả của thuật toán quay lui bằng ngôn ngữ C++ như sau:

```
void
backtracking(i)
{ if (i > n){
if (nghiem ThoaMan()) { print(X);
}
return;
}
for (j in D) {
if (check(j, i)) {
// Co the dat gia tri j vao vi tri i X[i] = j; // Dat j
backtracking(i+1); X[i] = 0; // Bo chon j
 }
        }
                }
```

2.3.4. Một số ví dụ cài đặt chương trình Quay lui

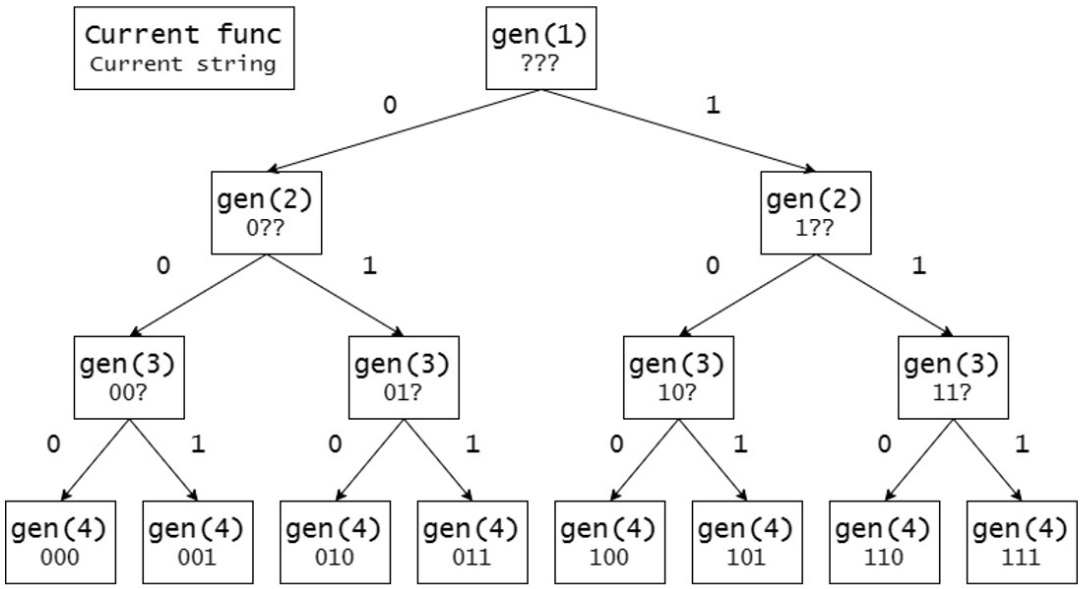
Ví dụ 1: Sinh các dãy nhị phân

Bài toán: Liệt kê tất cả các dãy nhị phân độ dài n, là dãy có tất cả n ký tự và gồm toàn các ký tự 0 và 1.

INPUT	OUTPUT
3	000, 001, 010, 011, 100, 101, 110, 111

Mô tả thuật toán bằng sơ đồ sau:

Sơ đồ mô tả thuật toán Sinh các dãy nhị phân



Tại hàm $gen(1)$, ta xét từng giá trị của ký tự hiện tại, sau đó gọi đến $gen(2)$ với từng ký tự đó. Tương tự như vậy, ta gọi đến $gen(3)$ từ các ký tự ở $gen(2)$ và rồi gọi đến

$gen(4)$ với các ký tự ở $gen(3)$. Tới $gen(4)$ ta đã duyệt hết các vị trí và không thể thử thêm nữa nên có thể in ra kết quả.

Viết chương trình giải quyết bài toán bằng ngôn ngữ C++ như sau:

```
int n;
string curString;
void genString(int pos){ if (pos > n){
cout << curString << "\n";
return;
}
for (char i = '0'; i <= '1'; i++){
curString.push_back(i); // Them ky tu moi vao
day genString(pos + 1);
curString.pop_back(); // Bo ky tu nay di
}
}
int main(){
```

```
cin >> n; curString = ""; genString(1); return 0;
}
```

Ví dụ 2: Liệt kê tập hợp

Bài toán: Liệt kê tất cả tập hợp khác rỗng chứa các số từ 1 đến n, theo thứ tự từ điển.

INPUT	OUTPUT
3	1 1 2 1 2 3 1 3 2 2 3 3

Phân tích bài toán: Việc liệt kê tất cả các tập hợp giống như việc liệt kê tất cả các dãy nhị phân, trong đó bit thứ i trong dãy nhị phân là 0/1 tương ứng với không chọn/chọn số thứ i vào tập hợp.

Ví dụ với $n = 3$ ta sẽ có bảng minh họa xâu nhị phân và các tập hợp như sau:

STT	XÂU NHỊ PHÂN	TẬP HỢP	GIẢI THÍCH
1	000	{}	Tập hợp rỗng
2	001	{3}	Chỉ lấy 3
3	010	{2}	Chỉ lấy 2
4	011	{2, 3}	Lấy 2 và 3
5	100	{1}	Chỉ lấy 1
6	101	{1, 3}	Lấy 1 và 3
7	110	{1, 2}	Lấy 1 và 2
8	111	{1, 2, 3}	Lấy 1, 2 và 3

Đối với bài toán này ta cần duy trì một vector để lưu tập hợp tạo được. Với mỗi phần tử có $X[i] = 1$, tức là chọn phần tử i thì ta sẽ thêm phần tử i vào cuối vector và duyệt đến vị trí kế tiếp. Sau khi duyệt xong thì lại bỏ phần tử cuối của vector đi.

Viết chương trình giải quyết bài toán bằng ngôn ngữ C++ như sau:

```
int n;
vector <vector<int>> res; vector<int> a;
void backtrack(int i) { if(i>n){
res.push_back(a); return;
}
// Khong chon phan tu i, vector giu nguyen
backtrack(i + 1);
// Chon phan tu i, vector them phan tu i vao
cuoi
a.push_back(i); backtrack(i + 1); a.pop_back();
}
int main(){
cin >> n; backtrack(1);
sort(res.begin(), res.end()); for(auto i: res) {
for(auto j: i){
cout << j << " ";
}
cout << "\n";
}
}
```

Ví dụ 3: Bài toán xếp hậu

Bài toán: Tìm tất cả các cách xếp n ($n \leq 12$) quân Hậu lên một bàn cờ $n \times n$ sao cho không có hai quân Hậu nào có thể ăn được nhau. Nếu có hai cách là hoán vị của nhau (về vị trí) thì chỉ tính là một, ví dụ hai tập hợp {(1, 2), (3, 4), (6, 6)} và {(1, 2), (5, 6), (3, 4)} chỉ lấy 1. Hai quân Hậu được gọi là có thể ăn được nhau nếu chúng nằm cùng hàng, cột hoặc đường chéo của bàn cờ.

Phân tích bài toán:

Giả sử quân Hậu thứ i nằm ở hàng xi và cột yi. Cách gọi này hơi ngược một chút so với hệ tọa độ Decartes thông thường nhưng nó sẽ rất hợp cho những bài toán liên quan đến bảng:

Khi nào thì hai quân Hậu A và B ăn nhau?

- Khi A và B nằm cùng hàng, tức là $xA = xB$
- Khi A và B nằm cùng cột, tức là $yA = yB$
- Khi A và B nằm trên cùng đường chéo. Lúc này có hai trường hợp:

$+ xA + yA = xB + yB$

$+ xA - yA = xB - yB$

Đối với đường chéo, bạn đọc có thể tự kiểm chứng thấy rằng, hiệu hoặc tổng của chỉ số hàng và chỉ số cột luôn luôn là một số không đổi đối với hai phần tử cùng đường chéo, lần lượt ứng với đường chéo chính (hướng tây bắc - đông nam) và

đường chéo phụ (hướng đông bắc - tây nam).

Vậy thì, sinh ra những bộ tọa độ đôi một thỏa mãn các điều kiện trên. Sinh các bộ tọa độ này lần lượt theo từng hàng, và đảm bảo rằng quân Hậu sau sẽ không cùng cột và cùng đường chéo với quân Hậu trước. Để khỏi phải dúng vòng lặp for lại từ đầu tập đã sinh để kiểm tra trùng lặp, duy trì một số mảng đánh dấu cột, đường chéo phụ, đường chéo chính:

$isInCol[], isInDiag1[], isInDiag2[]$

Trong đó:

- $isInCol[k]$ nhận giá trị $true$ nếu đã có một quân Hậu đã nằm ở cột k.
- $isInDiag1[k]$ nhận giá trị $true$ nếu đã có một quân Hậu đã nằm ở đường chéo phụ có tổng tọa độ x và y là k.
- $isInDiag2[k]$ nhận giá trị $true$ nếu đã có một quân Hậu đã nằm ở đường chéo chính có hiệu tọa độ x và y là k.

Một vòng đệ quy sẽ kết thúc nếu ta sinh thành công n quân Hậu. Lúc này, ta chỉ việc in kết quả, và đi tiếp tới các trường hợp khác.

Viết chương trình giải quyết bài toán Xếp hậu bằng ngôn ngữ C++ như sau:

```
int n;
// Mang danh dau cot, duong cheo phu va duong cheo chinh
bool isInCol [13], isInDiag1 [26], isInDiag2[26];
// Goi 2 tap rieng chi hang va cot
// Tap X co the bo qua do cac quan Hau duoc sinh lan luot theo tung hang vector <int> curQueensSetX, curQueensSetY;
// In ket qua dong (X, Y) void printQueensSet()
{
for (int i = 0; i < n; i++){
cout << "(" << curQueensSetX[i] << "," <<< curQueensSety[i] << ")"; if (in 1) cout << ", ";
}
cout << "\n";
}
// Ham de quy
void genQueens Set(int curRow) {
for (int curCol = 1; curCol <= n; curCol++) {
// Xac dinh duong cheo phu va chinh hien tai // intcurDiag1=curRow + curCol;
int curDiag2 = curRow - curCol + 13; // +13 de tranh chi so am
// Kiem tra toa do moi xem co thoa man khong //if(isInCol[curCol] == true) continue;
if (isInDiag1[curDiag1] == true) continue;
if (isInDiag2[curDiag2] == true) continue;
// Them no vao tap hop hien tai neu thoa man curQueensSetX.push_back(curRow); curQueensSetY.push_back(curCol); isInCol[curCol] = true;
isInDiag1[curDiag1]=true;isInDiag2[curDiag2] = true;
// Goi de quy them quan tiep theo hoac in ket qua if (curQueensSetX.size() == n) printQueensSet(); else genQueensSet (curRow + 1);
// Xoa quan vua them vao khoi tap hop curQueensSetX.pop_back(); curQueensSety.pop_back(); isInCol[curCol] = false;
isInDiag1[curDiag1]=false;isInDiag2[curDiag2] = false;
}
```

III. KẾT LUẬN

Nghiên cứu tập trung vào việc phân tích và ứng dụng thuật toán Quay lui trong lập trình, đặc biệt dành cho học sinh giỏi cấp THCS, nhằm giải quyết các bài toán phức tạp một cách hiệu quả. Đồng thời, đóng góp một cách hệ thống và rõ ràng vào việc làm sáng tỏ bản chất của thuật toán Quay lui (Backtracking) như một phương pháp giải quyết vấn đề dựa trên việc thử tất cả các lựa chọn có thể, đồng thời "quay lui" khi hướng đi hiện tại không còn phù hợp. Khác với thuật toán vét cạn tuần tự, Quay lui được trình bày như một kỹ thuật vét cạn thông minh, biết loại bỏ sớm các nhánh không thể dẫn đến lời giải, qua đó cải thiện hiệu năng xử lý bài toán. Trên cơ sở đó, bài viết cũng chỉ ra hạn chế thường gặp trong thực tiễn học tập khi học sinh quen sử dụng các thuật toán cơ bản, khiến chương trình trở nên chậm và tốn bộ nhớ với dữ liệu lớn, từ đó đề xuất việc áp dụng thuật toán Quay lui (kết hợp phương pháp nhánh cận khi cần) như một hướng tối ưu hóa độ phức tạp. Các mã giả và ví dụ minh họa cụ thể bằng C++, như bài toán Sinh dãy nhị phân, Liệt kê tập hợp hay Xếp Hậu với kỹ thuật dùng mảng đánh dấu, giúp người học dễ dàng tiếp cận và triển khai. Về ý nghĩa, bài viết không chỉ là tài liệu tham khảo thiết thực cho giáo viên, học sinh THCS trong dạy – học Tin học mà còn góp phần rèn luyện tư duy thuật toán, kỹ năng giải quyết vấn đề và định hướng phát triển năng lực cá nhân cho học sinh trong tương lai.

TÀI LIỆU THAM KHẢO

Thông tư 32/2018/TT-BGDĐT của Bộ giáo dục và đào tạo, ngày 26 tháng 12 năm 2018.

https://viblo.asia/p/thuat-toan-quay-lui-backtracking-bJzKmLbD59N.

Giáo Trình C++ Và Lập Trình Hướng Đối Tượng. Tác giả Nguyễn Hiếu Cường, Đỗ Văn Tuấn, Phạm Văn Ất, Lê Trường Thông. Nhà xuất bản Bách khoa Hà Nội.

Lê Minh Hoàng (2003), Giải thuật và lập trình.

Đệ quy và Thuật toán quay lui. Tác giả Nguyễn Đức Kiên, Trường Đại học Công nghệ, ĐHQGHN.