# Thu t toán quay lùi

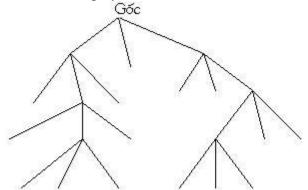
DUY Sơn (Sưu Tầm) Triệu Sơn – Thanh Hoá

Một bài toán liệt kê tổ hợp luôn cần phải đảm bảo hai nguyêntắc, đó là: không được bỏ sót một cấu hình và không được trùng lặp một cấu hình. Có thể nói rằng phương pháp liệt kê là cách cuốicùng để có thể giải được một số bài toán tổ hợp hiện nay. Mộttrong những phương pháp liệt kê có tính phổ dụng cao đó là phương pháp quay lui.

Nội dung chính của phương pháp này là việc xây dựng dần cácthành phần của cấu hình bằng cách thử tất cả các khả năng. Giả thiếtcấu hình cần được tìm được mô tả bằng một bộ giá trị  $(x_1,x_2,...,x_N)$ . Giả sử đã xác định được i - 1 thành phần  $(x_1,x_2,...,x_N)$ , bây giờ ta xác định thành phần  $x_i$  bằng cách duyệt tất cảcác khả năng có thể đề cử cho nó (đánh số từ 1 đến  $n_i$ ). Với mỗi khả năng j, kiểm tra xem j có được chấp nhận hay không. Cóthể có hai trường hợp có thể xảy ra:

- Nếu j được chấp nhận thì xác định  $x_i$  theo j, sau đó nếu j=N thì ta được một cấu hình,trái lại ta tiếptục tiến hành việc xác định  $x_{i+1}$ .
- Nếu thử tất cảcác khả năng mà mà không có khả năng nào được chấp nhận thì ta sẽlùi lại bước trướccđể xác định x  $_{i-1}$ .

Thông thường ta phân tích quá trình tìm kiếm thành cây tìm kiếm. Không gian tìm kiếm càng lớn hay càng nhiều khả năng tìm kiếm thì câytìm kiếm càng lớn, càng nhiều nhánh. Vì vậy hạn chế và bỏ bớt cácnhánh vô nghiệm của cây tìm kiếm thì sẽ tiết kiệm được thời gianvà bộ nhớ, tránh bị tràn dữ liệu. Quá trình tìm kiếm lời giải theothuật toán quay lui có thể được mô tả bởi mô hình cây tìm dướiđây:



Khả năng chọn x<sub>1</sub>

Khả năng chọn x2, với x1 đã chọn

Khả năng chọn x3, với x1, x2 đã chọn

Cần phải lưu ý là ta phải ghi nhớ tại mỗi bước đã đi qua,những khả năng nào đã thử để tránh trùng lặp. Những thông tin nàyđược lưu trữ theo kiểu dữ liệu ngăn xếp - Stack ( $vao\ sau\ ra\ tr U\acute{O}c$ ) - vì thế nên thuật toán này phù hợp thể hiện bởi thủ tục đệ quy. Ta có thể mô tả bước xác định x  $_i$  bởi thủ tục đệ quy sau:

Procedure Try (i: integer);

Var j : integer;

Begin

For j := 1to  $n_i$  do

If (chấp nhận j) then

Begin

(xác định x<sub>i</sub> theo j)

if i = N then (ghi nhận một cấu hình)

else try(i+1); End; End;

Trong thủ tục mô tả trên, điều quan trọng nhất là đưa rađược một danh sách các khả năngđề cử và xác định được giá trịcủa biểu thức logic [ chấp nhận j ]. Ngoài việc phụ thuộc j, giá trị này còn phụ thuộc vào việc đã chọn các khả năng tại i - 1bước trước đó. Trong những trường hợp như vậy, cần ghi nhớ *trạng thái mới* của quá trìnhsau khi [xác định x<sub>i</sub> theo j ] vàtrả lại *trạng thái cũ* sau lời gọi Try(i+1).Các trạng thái này được ghi nhận nhờ một số biến tổng thể(global), gọi là các biến trạng thái.

Dễ thấy rằng bài toán vô nghiệm khi ta đã duyệt hết mọi khảnăng mà không có khả năng nào thoả mãn yêu cầu. Ta nói rằng là đã vét cạn mọi trường hợp.Chú ý rằng là đến một lúc nào đó ta phải lùi liên tiếp nhiều lần.Từ đó suy ra rằng, thông thường bài toán vô nghiệm khi không thể lùiđược nữa. Thuật toán này sẽ không có tính khả thi cao bởi dùng thủtực đệ quy dễ bị lỗi tràn Stack.

Bài 1: Hành trình ký tự Cho tệp văn bản HT\_KITU.INP chứa các dòng ký tự chiều dài khôngquá 32. Hãy lập trình thực hiện các công việc sau: Lần lượt đọc cácdòng vào một xâu, sau đó từ xâu xây dựng lưới ô vuông dạng tam giácnhư sau: ví dụ xâu ='Vinh', lưới ô vuông có dạng như hình 1. Xuấtphát từ ô góc trên trái (chữ V), đi theo các hướng có thể để xâydựng lại xâu đã cho. Với mỗi hành trình thành công hãy in ra số thứtự của hành trình và chỉ ra hành trình trên lưới, mỗi ký tự của hànhtrình thay bằng một dấu '\*'.

#### Ví du:

Lưới kí tư ban đầu

	1	_
1	n	18
1		
	1	i Iình

Hành trình thứ 1

	100	- 1000
n	h	
h		
	n h	n h

Hình 2

Sau mỗi lời giải phải ấn ENTER để in lời giải tiếp.

# H**ướ**ngdẫn giải

Tổ chức hai mảng hai chiều F, Kt[1..32,1..32] of Char. Mảng Kt dùng để tạo ra ma trận kí tự dạng tam giác như trên gồm các kí tự từ xâuS đọc từ file dữ liệu. Mảng F dùng để ghi nhận các hành trình thànhcông, nếu ô (i,j) thuộc hành trình thì F[i,j] = '\*'. Sau khi xây dựng xong ma trận kí tự, ta dùng thủ tục đệ quy Try(i,j,h: byte) để tìm tất cả các hành trình. Giả sử ta đangở ô (i,j) nào đó trên hành trình và đã được một xâu kí tự độ dài h  $\leq$  length(S ). Nếu h = length(S )thì ta đã được một hành trình và ta sẽ ghi nhận nó, in ra màn hìnhhành trình đó. Còn nếu h < length(S )thì từ ô (i,j) tasẽ có thể đi theo hai hướng đó là đến ô (i,j+1) hoặc là ô (i+1,j). Từmỗi ô đó ta lại tiếp tục đến các ô khác để tìm hành trình. Quátrình đó được tiếp tục thực hiện các ô đó cho đến khi duyệt được hết nghiệm của bài toán.

Ban đầu ta xuất phát tại  $\hat{o}$  (1,1) và độ dài của xâu ta đang có là 1 nên ở chương trình chính ta gọi thủ tục Try(1,1,1). Để ý rằng nếu độ dài xâu S là L thì ta sẽ có tất cả  $2^{L-1}$  hành trình.

### Vănbản chương trình

Program Hanh\_trinh\_ki\_tu;

Uses Crt:

```
Const D : Array[1..2] of shortint=(0,1);
C: Array[1..2] of shortint= (1,0);
Fi = 'HT_KITU.INP';
Var Kt,F: Array[1..32,1..32] of Char;
S: string;
t: word;
dem: longint;
Procedure Init;
Var k,i,j: byte;
G: Text;
Begin
Assign(G,Fi); Reset(G);
Read(G,S); t := length(S);
Fillchar(F, size of (F), '');
F[1,1]:='*'; k:=0;
For i := 1 to t do
begin
For j:=1 to t do
begin
Kt[i,j]:=S[j+k];
If Kt[i,j] = \#0 then Kt[i,j] := ' ';
end;
inc(k);
end;
Close(G);
End;
Procedure Write_Out;
Var i,j: Byte;
Begin
Inc(dem);
TextColor(Red); Writeln('Hanh trinh thu:',dem);
For i:=1 to t do
begin
For i:=1 to t do
If F[i,j]='*' then
begin
TextColor(White); Write('* ')
end
Else
begin
TextColor(Green);Write(Kt[i,j],' ');
end;
Writeln;
end;
Readln;
End;
```

```
Procedure Try(i,j,h: byte);
Var k,x,y: byte;
Begin
If h = t then Write_Out else
begin
For k:=1 to 2 do
begin
x := i + D[k]; y := j + C[k];
F[x,y]:='*';
Try(x,y,h+1);
F[x,y]:=';
end;
end;
End:
BEGIN
Clrscr:
Init:
Try(1,1,1);
END.
```

#### Bài 2: Biểu thức zero

Cho một số tự nhiên  $N \le 9$ . Giữa các số từ 1 đến N hãy thêm vào các dấu + và - sao cho kết quả thu được bằng 0. Hãy viết chương trình tìm tất cả các khả năng có thể.

Dữ liệu vào: Lấy từ file văn bản ZERO.INP với một dòng ghi số N.

Dữ liệu ra: Ghi vào file văn bản có tên ZERO.OUT có cấu trúc như sau:

- Dòng đầu ghi sốlượng kết quả tìm được.
- Các dòng sau mỗidòng ghi một kết quả tìm được.

Ví du

ZERO.INP	ZERO.OUT
7	6
	1-2-3-4-5+6+7=0
	1-2+3+4-5+6-7=0
	1-23-45+67 = 0
	1-23+4+5+6+7 = 0
	1+2-3-4+5+6-7=0
	1+2-3+4-5-6+7=0

# H**ướ**ng dẫn gi**ả**i

áp dụng thuật toán đệ quy quay lui để giải quyết bài toánnay, ta sẽ dùng thủ tục đệ quy Try(i). Giả sử ta đã điền các dấu'+' và '-' vào các số từ 1 đến i, bây giờ cần điền các dấugiữa i và i + 1. Ta có thể chọn một trong ba khả năng: hoặc là điềndấu '+', hoặc là điền dấu '-', hoặc là không điền dấu nào cả.Khi đã chọn một trong ba khả năng trên, ta tiếp tục lựa chọn dấuđể điền vào giữa i + 1 và i + 2 bằng cách gọi đệ quy Try(i+1). Ta sẽlần lượt duyệt tất cả các khả năng đó để tìm tất cả các nghiệm của bài toán, như vậy bài toán sẽ không bị thiếu nghiệm.

Nếu i = N ta sẽ kiểm tra xem cách điền đó có thoả mãn kết quảbằng 0 hay không. Để kiểm tra ta dùng thủ tục Test trong chương trình. Nếutổng đúng bằng 0 thì cách điền

đó là một nghiệm của bài toán, taghi nhận nó. Nếu i < N thì tiếp tục gọi Try(i+1). Trong chương trìnhta dùng biến dem để đếm các cách điền thoả mãn, còn mảng M kiểu string sẽ ghi nhận mọi cách điền dấu thoả mãn yêu cầu bài toán.

#### Văn b**ả**n ch**ươ**ng trình

```
Program Zero sum:
Type MangStr = array[1..15] of string;
Const Fi ='ZERO.INP';
Fo ='ZERO.OUT';
Dau : array[1..3] of string[1] = ('-', '+', '');
S: array[1..9] of char = ('1', '2', '3', '4', '5', '6', '7', '8', '9');
ChuSo = ['1'..'9'];
Var N,k,dem: byte;
D: array[2..9] of string[1];
F: Text;
St: String;
M : MangStr;
Procedure Write_out;
Var i : byte;
Begin
Assign(F,Fo); Rewrite(F);
Writeln(F,dem);
For i:= 1 to dem do writeln(F,M[i],' = 0');
Close(F); Halt;
End;
Procedure Read inp;
Begin
Assign(F,Fi); Reset(F);
Read(F,N); Close(F);
If N < 3 then write out;
End;
Function DocSo(S : String): longint;
Var M : longint;
t: byte;
Begin
M:=0; t:=0;
If S[k] in ['+','-'] then
begin
t := k; Inc(k);
end;
While (k \le length(S)) and (s[k]) in ChuSo do
m := m*10 + ord(s[k]) - ord('0');
Inc(k);
end;
If (t \lt\gt 0) and (S[t] = '-') then DocSo:= -M
```

```
else DocSo:= M;
End;
Procedure Test;
Var St : string;
i: byte;
T: longint;
Begin
St:= '1'; k:= 1; T:= 0;
For i = 2 to N do St:= St + D[i] + S[i]:
While k < length(St) + 1 do T := T + DocSo(St);
If T = 0 then
begin
Inc(dem); M[dem] := St;
end:
End;
Procedure Try(i: byte);
Var j : byte;
Begin
For j := 1 to 3 do
begin
D[i] := Dau[i];
If i = N then Test else try(i+1);
end;
End;
BEGIN
Read inp;
Try(2);
Write out;
END.
```

## Bài 3: Xổ số điện toán

Có N người (đánh số từ 1 đến N) tham gia một đợt xổ số điệntoán. Mỗi người nhận được một thẻ gồm M ô (đánh số từ 1 đếnM). Người chơi được chọn K ô trong số các ô đã cho bằng cách đánhdấu các ô được chọn. Sau đó các thẻ này được đưa vào máy tínhđể xử lý.

Máy tính chọn ra K ô ngẫu nhiên (gọi là các ô kết quả) và chấm điểm từng thẻ dựa vào kết quả đã sinh. Cứ mỗi ô chọn đúng vớiô kết quả thì thẻ chơi được tính 1 điểm. Giả thiết biết các ôchọn cũng như các điểm tương ứng của từng thẻ chơi, hãy xác định tất cả các kết quả có thể cómà máy sinh ra.

Dữ liệu vào đọc từ file vănbản XOSO.INP gồm:

- Dòng đầu ghi cácsố N, M, K
- Dòng thứ i trongN dòng tiếp ghi thẻ chơi của người i gồm K+1 số: K số đầu là các sốhiêu

của các ôchọn, cuối cùng là điểm tương ứng.

Ghi kết quả ra file văn bản XOSO.OUT, mỗi dòng là một kết quả gồmK số ghi số hiệu các ô mà máy đã sinh.

#### Ghi chú:

- Các số trên cùng mộtdòng trong các file vào/ ra, được ghi cách nhau ít nhất một dấu trắng.
- Giới hạn kích thước:N ≤ 100, M ≤50, K ≤10.
- Dữ liệu vào trong cáctest là hợp lệ và đảm bảo có ít nhất một đáp án.

#### Ví dụ:

XOSO.INP	XOSO.OUT
5 9 4	1 2 3 4
2 4 6 8 2	2 3 4 7
56890	
2 4 5 6 2	
1 2 3 7 3	
3 5 6 9 1	

## Hướng dẫn giải

Ta nhận thấy rằngmỗi nghiệm của bài toán chính là một cấu hình của tổ hợp chập K củaM phần tử. Ta áp dụng thuật toán quay lui để duyệt mọi cấu hình tổhợp để tìm ra cấu hình thoả mãn. Tuy nhiên để giảm bớt số lần duyệtta cần phải loại những thẻ mà chúng có tổng điểm bằng 0 và cầnđánh dấu những thẻ đã được chọn.

Dùng mảng ok[0..51] of boolean để phân biệt giữa ô có điểm và những ô không có điểm. Nếu ok[i]= false thìcho biết thẻ thứ i không có điểm. Còn logic[i,j] = true cho ta biết người thứ i đánh dấu vàothứ j của thẻ.

## Văn bản chương trình

```
Program Xoso_dien_toan;

Type MangA = array[0..100,0..11] of byte;

MangBool = array[0..51] of boolean;

MangLogic = array[0..101,0..51] of boolean;

Cauhinh = array[0..11] of byte;
```

```
Const Fi = 'XOSO.INP';
Fo = 'XOSO.OUT';

var M,N,K : byte;
A : MangA;
B : Cauhinh;
Ok : MangBool;
Diem : integer;
Logic : MangLogic;
F : Text;
```

Procedure Init;

Begin

Fillchar(A,sizeof(A),0);

Fillchar(B,sizeof(B),0);

Fillchar(ok,sizeof(ok),1);

Fillchar(logic,sizeof(logic),0);

End;

Procedure Read\_inp;

Var i,j: byte;