BÀI 3: BÀI TOÁN LIỆT KÊ TỔ HỢP

Giới thiệu

Bài học này trình bày nội dung bài toán liệt kê tổ hợp, bài toán này quan tâm đến tất cả các cấu hình có thể có, vì thế lời giải của nó cần được biểu diễn dưới dạng thuật toán "vét cạn" tất cả các cấu hình. Lời giải trong từng trường hợp cụ thể sẽ được máy tính giải quyết nhờ chạy một chương trình cài đặt theo thuật toán đã tìm. Bài toán liệt kê thường được "làm nền" cho nhiều bài toán khác. Hiện nay, một số bài toán tổ hợp vẫn chưa có cách nào giải ngoài cách giải liệt kê. Khó khăn chính của cách giải này là có quá nhiều cấu hình, tuy nhiên tính khả thi của phương pháp liệt kê ngày càng được nâng cao nhờ sự tiến bộ nhanh chóng về chất lượng của máy tính điện tử.

Nội dung

- Giới thiệu bài toán liệt kê tổ hợp
- Trình bày thuật toán quay lui
- Liệt kê một số cấu hình cơ bản

Thời lượng học

- 6 tiết

Mục tiêu

Sau khi học bài này, các bạn có thể:

- Nắm được yêu cầu của bài toán liệt kê tổ hợp.
- Sử dụng thuật toán quay lui trong việc thực hiện bài toán liệt kê tổ hợp.
- Liệt kê được một số câu hình cơ bản như: liệt kê dãy nhị phân, liệt kê hoán vị, liệt kê tổ hợp.

Sử dụng các kiến thức của bài toán liệt kê trong việc giải quyết một số tình huống thực tế.

TÌNH HUỐNG DẪN NHẬP

Tình huống

"Tìm cách xếp 8 quân Hậu trên bàn cờ Vua sao cho không có quân nào ăn được quân nào".

Câu hỏi

Có bao nhiêu cách xếp hậu thỏa mãn yêu cầu của bài toán và đó là những cách nào?

3.1. Giới thiệu bài toán

Bài toán liệt kê tổ hợp nhằm lần lượt đưa ra từng cấu hình sao cho *không bỏ sót* và *không trùng lặp*. Như vậy, khác với những cách giải thông thường, trong đó trình bày các lập luận, chứng minh, hay các tính toán qua các công thức, lời giải của bài toán này phải được trình bày dưới dạng thuật toán, trong đó chỉ ra các bước xây dựng từng cấu hình thỏa mãn điều kiện đã nêu.

Vào thời chưa có máy tính, hoặc máy tính còn dưới dạng sơ khai, việc liệt kê chủ yếu nhờ vào sức thủ công, vì thế kết quả rất hạn chế. Khi đó, việc liệt kê chỉ được thực hiện trên những bài toán kích thước không đáng kể, nhằm minh họa một số khái niệm hay kiểm chứng một vài kết quả đơn giản. Hiện nay, với sự phát triển mạnh mẽ của máy tính, tốc độ lên tới hàng triệu phép tính trong một giây, việc liệt kê nhờ máy tính ngày càng khả thi và giải pháp liệt kê ngày càng được chú ý, nhất là nhờ nó mà một số bài toán tồn đọng hàng thế kỷ đã được giải quyết.

Với sự hỗ trợ của máy tính, bài toán liệt kê thường được làm nền để giải quyết những bài toán tổ hợp khác (các bài toán đếm, tồn tại, tối ưu) trong những tình huống không còn lựa chọn tốt hơn. Khó khăn chính của bài toán này là số cấu hình thường quá lớn mà việc chờ đợi kết quả vượt quá khả năng ngay cả khi thực thi bằng máy tính. Để khắc phục khó khăn này, một mặt con người cố gắng xây dựng những thuật toán hữu hiệu, một mặt nâng cao khả năng xử lý của máy tính. Việc nghiên cứu chế tạo các máy tính có nhiều bộ xử lý đồng thời với việc phát triển các giải thuật song song chắc chắn sẽ nâng cao tính khả thi của những bài toán liệt kê lên rất nhiều.

Bài này giới thiệu một thuật toán cơ bản mang tính phổ dụng của toán hữu hạn cho phép liệt kê các cấu hình và cách cài đặt nó bằng một chương trình trên máy tính.

3.2. Thuật toán quay lui

Thuật toán *quay lui* thực chất là thuật toán duyệt tất cả các khả năng xây dựng cấu hình sao cho không bỏ sót và không trùng lặp. Thông thường một cấu hình được biểu diễn dưới dạng một bộ có thứ tự $(x_1, x_2, ..., x_n)$, trong đó các thành phần được xác định từ những tập giá trị (hữu hạn) nào đấy, thỏa mãn những điều kiện đề ra.

Nội dung của thuật toán quay lui là lần lượt xác định các thành phần của cấu hình bắt đầu từ thành phần đầu tiên. Để xác định một thành phần, ta thử tất cả các giá trị khả dĩ cho nó trong trạng thái các thành phần trước đấy đã được xác định. Vì thế, thích hợp hơn cả là phát biểu thuật toán này bằng quy nạp.

Giả sử đã xác định được các thành phần $x_1, x_2, ..., x_{i-1}$. Dưới đây là bước xác định thành phần x_i (bước thử thứ i). Gọi S_i là tập các giá trị thử cho x_i (gọi là *tập đề cử*, nó được xác định từ những điều kiện của cấu hình). Duyệt tất cả các giá trị j thuộc S_i và thử nó cho x_i. Xảy ra hai tình huống:

- Có một j mà việc thử cho x_i là chấp nhận được (dựa vào các điều kiện của cấu hình). Khi đó gán j cho x_i. Nếu i = n (x_i là thành phần cuối) thì liệt kê được một cấu hình (sau đó duyệt j tiếp, nếu hết, lùi lại bước trước để thử giá trị khác cho x_{n-1}), trái lại sang bước i + 1 để xác định thành phần kế tiếp.
- Mọi j thuộc S_i đều không được chấp nhận. Khi đó lùi về bước trước để thử giá trị khác cho x_{i-1}.

Để không bỏ sót, tập giá trị đề cử S_i cho x_i cần phải xem xét một cách cẩn thận, mặc dù không phải giá trị đề cử nào cũng chấp nhận được, nhưng nếu bỏ sót giá trị đề cử thì sẽ dẫn đến bỏ sót cấu hình. Để không trùng lặp, trong mỗi bước tìm kiếm, ta phải lưu lại những thông tin cần thiết để khi lùi lại, không thử những giá trị đã thử rồi. Những thông tin này cần được cất giữ theo cơ chế vào sau, ra trước (ngăn xếp).

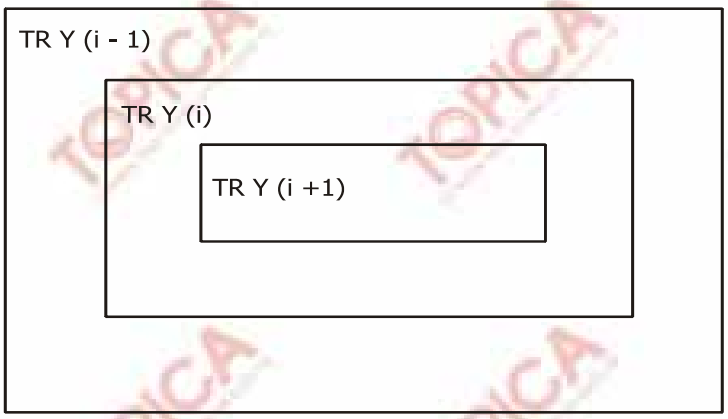
Để cài đặt, tốt hơn cả là dùng một ngôn ngữ lập trình cho phép gọi đệ quy. Với những ngôn ngữ này, ta tận dụng được cơ chế ngăn xếp của việc đệ quy mà không phải tự tổ chức lấy ngăn xếp. Điều này làm việc viết chương trình trở nên đơn giản rất nhiều. Hiện nay các ngôn ngữ thuật toán được cài đặt trên máy tính như C, Pascal đều có khả năng này.

Nội dung của thuật toán quay lui có thể mô tả qua thủ tục đệ quy dưới đây (viết mô phỏng theo ngôn ngữ Pascal):

Thuật toán quay lui

```
PROCEDURE TRY (i: INTEGER);
VAR j: INTEGER;
BEGIN
   FOR (j thuộc Si) DO
   IF (chấp nhận j) THEN
   BEGIN
         xi := j;
         IF (i = n) THEN (ghi nhận một cấu hình) ELSE
TRY(i+1);
   END;
END;
```

Thủ tục TRY(i) xác định x_i bằng cách duyệt tất cả các giá trị đề cử cho nó (vòng lặp FOR). Trong thủ tục có khai báo biến địa phương j dùng để duyệt các giá trị đề cử (không mất tính tổng quát, ta có thể giả thiết các giá trị này là nguyên). Khi xác định xong x_i, việc tiến hành bước tiếp được thực hiện bằng lời gọi đệ quy TRY(i+1). Khi xong vòng lặp duyệt, thủ tục TRY(i) kết thúc, và trở về vòng lặp duyệt của TRY(i−1) để tiếp tục thử các giá trị đề cử khác cho x_{i-1}.



Vòng lặp đệ quy lồng nhau của TRY(i)

Trong TRY(i), mệnh đề (*chấp nhận j*) là một biểu thức lôgic, không những phụ thuộc j mà trong nhiều tình huống còn phụ thuộc vào các giá trị đã thử ở những bước trước, vì thế để tính biểu thức này, ta cần tổ chức thêm những biến phụ (được khai báo toàn cục) ghi nhận sự thay đổi trạng thái của bài toán sau mỗi bước tìm kiếm (vì thế các biến này được gọi là các *biến trạng thái*). Độ phức tạp của những biến này phụ thuộc vào độ phức tạp của cấu hình cần liệt kê. Nếu có mặt những biến như vậy, trong TRY(i) cần thêm vào các khối lệnh (*ghi nhận trạng thái mới*), (*trả về trạng thái cũ*), nhằm cập nhật lại giá trị của các biến này tại những nơi thích hợp, như đề nghị dưới đây:

Vòng lặp đệ quy lồng nhau của Try(i)

```
PROCEDURE TRY (i: INTEGER);
VAR j: INTEGER;
BEGIN
FOR (j thuộc Si) DO
IF (chấp nhận j) THEN
BEGIN
     xi := j;
     (ghi nhận trạng thái mới);
     IF (i = n) THEN (ghi nhận một cấu hình) ELSE TRY(i+1);
      (trả về trạng thái cũ);
  END;
END;
```

TRY(i) được khởi động bằng lời gọi TRY(1) trong chương trình chính. Khi TRY(1) kết thúc, quá trình liệt kê được hoàn tất. Dĩ nhiên, trước khi gọi TRY(1), trong chương trình chính cần phải gọi các thủ tục nhập dữ liệu và khởi gán các giá trị ban đầu. Cũng nên thiết kế một thủ tục làm nhiệm vụ (*ghi nhận một cấu hình*) được gọi trong TRY(i), nhằm xử lý cấu hình nhận được cho phù hợp với yêu cầu của bài toán (có thể đưa ra màn hình, ghi ra file, hoặc áp dụng một thao tác nào đấy trên cấu hình này). Chẳng hạn, nếu dùng liệt kê để giải bài toán đếm, thì thủ tục này đơn giản là tăng biến đếm lên một đơn vị (biến đếm cần được khởi gán 0), nếu chỉ cần chứng minh có cấu hình (bài toán tồn tại), thì khi nhận được cấu hình đầu tiên, ta có thể kết thúc ngay.

Thứ tự liệt kê các cấu hình phụ thuộc vào thứ tự duyệt các giá trị đề cử cho mỗi thành phần. Thông thường các giá trị đề cử được sắp xếp tăng dần, khi đó các biểu diễn cấu hình sẽ được xếp theo thứ tự từ điển.

Tên gọi thuật toán quay lui, xuất phát từ chính nội dung của nó. Thuật toán này cũng được biết đến với tên gọi thuật toán *thử-sai*.

Cũng chú ý rằng, trên đây chỉ là mô hình có tính chất định hướng cho việc xây dựng chương trình thực hiện thuật toán quay lui. Nội dung cụ thể của nó phụ thuộc vào kết quả phân tích cấu hình, trong đó việc tổ chức dữ liệu để mô tả cấu hình, việc xác định các tập đề cử, việc xây dựng các biến trạng thái và biểu thức kiểm tra giá trị thử, ... đóng một vai trò quan trọng trong việc quyết định chất lượng của chương trình. Ngoài việc nắm vững ngôn ngữ được dùng, người lập trình cần phải có những kiến thức toán học nhất định, liên quan đến vấn đề đang xét.

Mục dưới đây đưa ra một số thí dụ minh họa việc dùng thuật toán quay lui để liệt kê một số cấu hình đơn giản, trong đó các chương trình đều được cài đặt theo cùng khuôn dạng như sau:

Ví dụ: Thuật toán quay lui

```
PROCEDURE INIT;
PROCEDURE OUT;
PROCEDURE TRY(i: INTEGER);
BEGIN (* chương trình chính*)
  INIT;
  TRY(1);
END.
```

Thủ tục INIT nhập dữ liệu và khởi gán các giá trị ban đầu, thủ tục OUT đếm và đưa ra cấu hình $x_1, x_2, ..., x_n$ mỗi khi xây dựng xong, thủ tục TRY(i) thực hiện việc xác định x_i bằng đệ quy.

Trong các thủ tục trên, thủ tục TRY(i) là quan trọng nhất. Vì thế trong các thí dụ minh họa, chủ yếu chúng tôi trình bày việc phân tích và thiết kế thủ tục này.

3.3. Liệt kê một số cấu hình đơn giản

3.3.1. Liệt kê dãy nhị phân

Một dãy nhị phân độ dài n (còn được gọi là chuỗi n bit) là một bộ có thứ tự gồm n thành phần $(x_1, x_2, ..., x_n)$ trong đó mỗi thành phần x_i nhận một trong hai giá trị 0, 1.

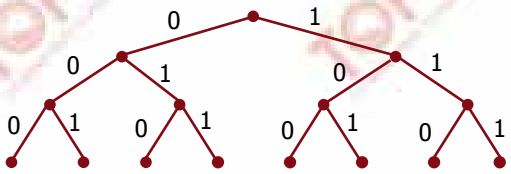
Trong bài toán đếm ta đã biết số các dãy nhị phân độ dài n là 2^n, bây giờ ta giải bài toán liệt kê tất cả các dãy nhị phân độ dài n bằng cách viết một chương trình theo mô hình đã cài đặt.

Theo định nghĩa dãy nhị phân, giá trị đề cử cho x_i là {0, 1}, việc chọn giá trị 0 hay 1 cho thành phần này mặc nhiên được chấp nhận, không phụ thuộc vào các giá trị của các thành phần trước đấy. Đây là trường hợp mà TRY(i) có dạng đơn giản nhất, trong đó không có các khối (*chấp nhận j*), (*ghi nhận trạng thái mới*), (*trả về trạng thái cũ*).

Thuật toán tìm kiếm quay lui các dãy nhị phân

```
PROCEDURE TRY (i: INTEGER);
VAR j: INTEGER;
BEGIN
   FOR j := 0 TO 1 DO
   BEGIN
         x[i] := j;
         IF (i = n) THEN OUT ELSE TRY(i+1);
   END;
END;
```

Các bước tìm kiếm quay lui các dãy nhị phân độ dài 3 có thể mô tả bởi cây liệt kê dưới đây:



Kết quả chạy chương trình với n = 3, ta được $2^3 = 8$ dãy nhị phân theo thứ tự như sau:

1) 0 0 0
2) 0 0 1
3) 0 1 0
4) 0 1 1
5) 1 0 0
7) 1 0 1
8) 1 1 0
9) 1 1 1

3.3.2. Liệt kê hoán vị

Một hoán vị của các phần tử 1, 2, ..., n là một cách xếp thứ tự của các phần tử đó. Như thế ta có thể biểu diễn hoán vị đang xét như một bộ có thứ tự gồm n thành phần (x_1, x_2, ..., x_n) trong đó các thành phần x_i lấy những giá trị khác nhau trên tập {1, 2, ..., n}.

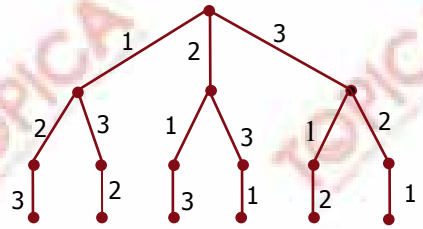
Từ đó nhận được tập đề cử cho x_i là {1, 2, ..., n} và điều kiện chấp nhận j cho x_i là j không được trùng với các giá trị đã gán cho các thành phần trước đấy (j chưa được dùng).

Để kiểm tra điều kiện này, ta xây dựng các biến lôgic b_j (j = 1, 2, ..., n), đóng vai trò các biến trạng thái, trong đó mỗi biến b_j kiểm soát trạng thái của j với quy ước b_j bằng TRUE nếu j chưa được dùng và b_j bằng FALSE nếu trái lại. Khi đó, mệnh đề (chấp nhận j) sẽ là (b_j) và các câu lệnh (ghi nhận trạng thái mới), (trả về trạng thái cũ) sẽ tương ứng với các lệnh gán (b_j := FALSE) và (b_j := TRUE). Các biến trạng thái b_j cần phải được khởi gán tất cả bằng TRUE trước khi gọi TRY(1).

Thuật toán tìm kiếm quay lui các hoán vị

```
PROCEDURE TRY (i: INTEGER);
VAR j: INTEGER;
BEGIN
   FOR j := 1 TO n DO
   IF (bj) THEN
   BEGIN
         xi := j;
         bj := FALSE;
         IF (i = n) THEN OUT ELSE TRY(i+1);
         bj := TRUE;
   END;
END;
```

Các bước tìm kiếm quay lui các hoán vị độ dài 3 có thể mô tả bởi cây liệt kê dưới đây:



Kết quả chạy chương trình với n = 3, ta được 3! = 6 hoán vị theo thứ tự như sau:

1) 1 2 3
2) 1 3 2
3) 2 1 3
4) 2 3 1
5) 3 1 2
7) 3 2 1

3.3.3. Liệt kê tổ hợp

Một tổ hợp chập m của n phần tử {1, 2, ..., n} (m ≤ n) là một tập con m phần tử của tập đã cho. Mỗi tập con như vậy được biểu diễn dưới dạng một bộ không kể thứ tự (x_1, x_2, ..., x_m) gồm m thành phần nhận những giá trị khác nhau từ tập {1, 2, ..., n}. Như thế các thành phần của nó phải được ràng buộc thêm điều kiện:

$$1 \le x_1 < x_2 < ... < x_m \le n.$$

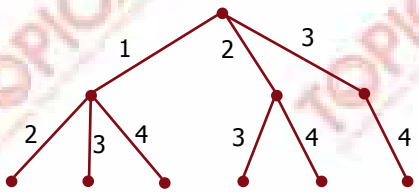
Nhận xét rằng để điều kiện trên thỏa mãn, cần và đủ là tại mỗi bước thử thứ i (i = 1, 2, ..., m) giá trị của x_i phải thỏa mãn:

$$x_{i-1}+1 \le x_i \le n-m+i \qquad (\text{bổ sung } x_0 = 0)$$

từ đó nhận được tập đề cử cho x_i là $\{x_{i-1}+1, ..., n-m+i\}$ và mọi giá trị thuộc tập này đều mặc nhiên được chấp nhận. Với tập đề cử đã chọn, TRY(i) trở thành đơn giản giống như trường hợp liệt kê dãy nhị phân.

```
PROCEDURE TRY (i: INTEGER);
VAR j: INTEGER;
BEGIN
   FOR j := x[i-1]+1 TO n-m+i DO
   BEGIN
         x[i] := j;
         IF (i = m) THEN OUT ELSE TRY(i+1);
   END;
END;
```

Các bước tìm kiếm quay lui các tổ hợp chập 2 của 4 có thể mô tả bởi cây liệt kê dưới đây:



Kết quả chạy chương trình với m = 2, n = 4 ta được $C_4^2 = 6$ tổ hợp theo thứ tự như sau:

1) 1 2
2) 1 3
3) 1 4
4) 2 3
5) 2 4
6) 3 4

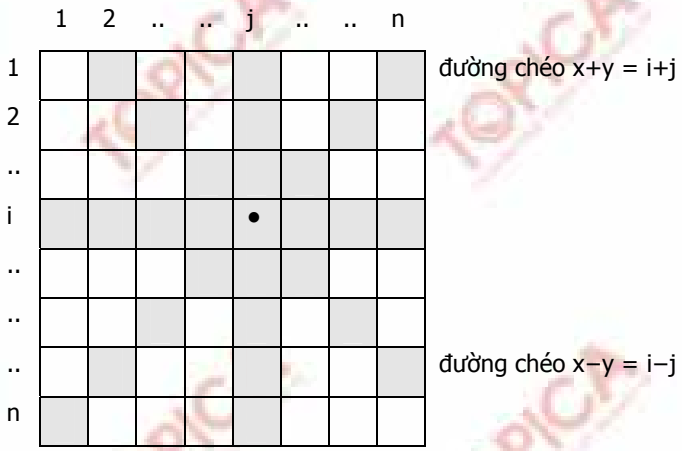
3.4. Một số liệt kê khác

Trong mục này, ta sẽ xét một số bài toán liệt kê có liên quan đến các quân cờ mà việc xây dựng các cấu hình của chúng là những thí dụ điển hình của thuật toán quay lui.

3.4.1. Bài toán xếp Hậu

Nội dung bài toán như sau: "Tìm cách xếp 8 quân Hậu trên bàn cờ Vua sao cho không có quân nào ăn được quân nào". Lời giải được tìm kiếm bằng cách xếp dần các quân Hậu, sao cho quân Hậu mới xếp không nằm ở vị trí bị các quân Hậu đã xếp khống chế. Nếu không tìm được một vị trí như vậy, cần xếp lại quân Hậu trước. Thực chất cách giải này là thử lần lượt tất cả vị trí để xếp Hậu và khi cần thiết thì quay lui. Rõ ràng là cách giải này có thể áp dụng cho mọi kích thước n (bàn cờ n×n và n quân Hậu) và đưa ra được tất cả các cách xếp có thể. Ta sẽ lập một chương trình, theo mô hình quay lui đã trình bày, liệt kê tất cả các cách xếp Hậu, thỏa mãn điều kiện đã nêu, với kích thước n cho trước.

Đầu tiên là việc biểu diễn cách xếp Hậu. Đánh số cột và dòng của bàn cờ từ 1 đến n. Vì quân Hậu ăn theo dòng nên mỗi dòng xếp đúng một quân Hậu (quân Hậu i bầy vào dòng i, i = 1, 2, ..., n). Như thế vị trí của các quân Hậu được xác định nếu biết tọa độ cột của chúng. Điều này gợi ý biểu diễn mỗi cách xếp Hậu như là một bộ có thứ tự (x_1, x_2, ..., x_n) trong đó x_i là tọa độ cột của quân Hậu i. Khi đó, tập đề cử cho x_i sẽ là {1, 2, ..., n}. Giá trị j thử cho x_i được chấp nhận khi và chỉ khi ô (i, j) không bị các quân Hậu trước chiếu đến (còn tự do). Trạng thái của ô (i, j) cần được xác định trước khi thử giá trị j cho x_i. Điều này được thực hiện bằng cách tổ chức các biến trạng thái thích hợp, suy từ luật ăn quân của quân Hậu (ngang, dọc và hai đường chéo):



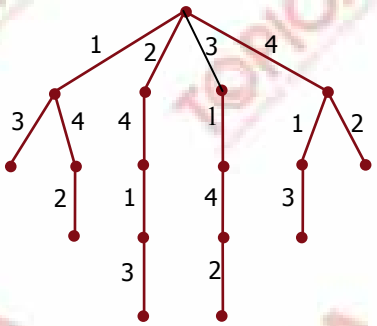
Các ô bị khống chế bởi quân Hậu ở ô (i, j)

Để ô (i, j) là tự do, cần hội các điều kiện: dòng i tự do, cột j tự do và hai đường chéo qua (i, j) là tự do. Các dòng không cần xét vì mỗi dòng chỉ có đúng một quân Hậu. Để kiểm soát các cột, ta đưa vào các biến lôgic a_1, a_2, ..., a_n trong đó a_j bằng TRUE nếu cột j còn tự do và bằng FALSE nếu trái lại. Các đường chéo có hai dạng: phương trình x + y = k và phương trình x − y = k (x là tọa độ dòng, y là tọa độ cột, k là hằng số). Với đường chéo dạng x + y = k, giá trị của k biến thiên từ 2 đến 2n, vì thế để kiểm soát chúng ta đưa vào các biến lôgic b_2, b_3, ..., b_{2n} trong đó b_k bằng TRUE nếu đường chéo này còn tự do và bằng FALSE nếu trái lại. Với đường chéo dạng x − y = k,

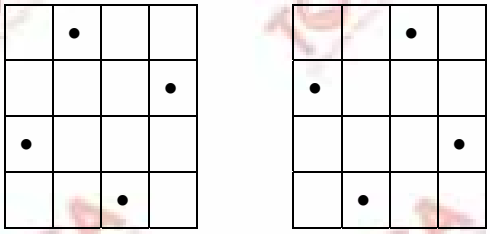
giá trị của k biến thiên từ 1 − n đến n − 1, vì thế để kiểm soát chúng ta đưa vào các biến lôgic c_{1-n}, c_{2-n}, ..., c_{n-1} (chú ý giá trị chỉ số có thể âm) trong đó c_k bằng TRUE nếu đường chéo này còn tự do và bằng FALSE nếu trái lại. Khi đó điều kiện (chấp nhận j cho x_i) sẽ là biểu thức lôgic (a_j AND b_{i+j} AND c_{i-j}), câu lệnh (ghi nhận trạng thái mới) sẽ là các lệnh gán (a_j: = FALSE; b_{i+j}: = FALSE; c_{i-j}: = FALSE) và câu lệnh (trả về trạng thái cũ) sẽ là các lệnh gán (a_j: = TRUE; b_{i+j}: = TRUE; c_{i-j}: = TRUE). Các biến trạng thái a_j (j = 1, ..., n), b_k (k = 2, ..., 2n), c_k (k = 1−n, ..., n−1) cần được khởi gán TRUE trước khi gọi TRY(1).

```
PROCEDURE TRY (i: INTEGER);
VAR j: INTEGER;
BEGIN
   FOR j := 1 TO n DO
   IF (aj AND bi+j AND ci-j) THEN
   BEGIN
         xi := j;
         aj := FALSE; bi+j := FALSE; ci-j := FALSE;
         IF (i = n) THEN OUT ELSE TRY(i+1);
         aj := TRUE; bi+j := TRUE; ci-j := TRUE;
   END;
END;
```

Các bước tìm kiếm quay lui các cách xếp 4 quân Hậu trên bàn cờ kích thước 4 có thể mô tả bởi cây liệt kê dưới đây:

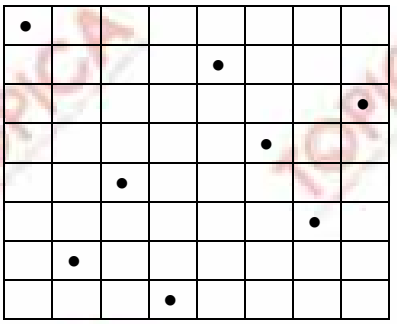


Kết quả chạy chương trình với n = 4 ta được 2 cách xếp Hậu theo thứ tự sau:



1) (2, 4, 1, 3) 2) (3, 1, 4, 2)

Với n = 8 (bàn cờ thông thường), chương trình cho 92 lời giải. Cách xếp đầu tiên tìm được là:



(1, 5, 8, 6, 3, 7, 2, 4)

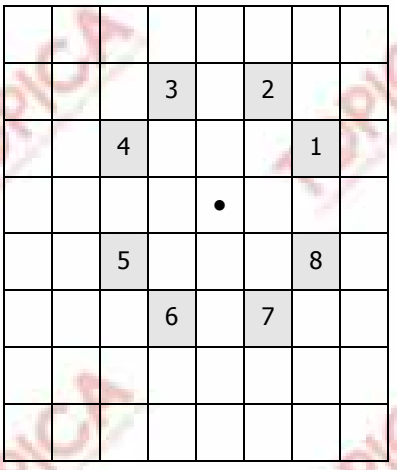
Nếu chỉ quan tâm đến số cách xếp Hậu, ta có thể chạy chương trình với một số giá trị của n và được kết quả dưới đây:

n	4	5	6	7	8	9	10	11	12	13	14
số cách	2	10	4	40	92	352	724	2680	14200	73712	365596

3.4.2. Bài toán Mã đi tuần

Một hành trình của quân Mã được gọi là *đi tuần* nếu trên hành trình đó, Mã đi qua tất cả các ô của bàn cờ, mỗi ô đúng một lần. Hãy liệt kê tất cả các hành trình đi tuần của Mã trên bàn cờ kích thước n với ô xuất phát là (u, v).

Biểu diễn hành trình của Mã bằng một ma trận vuông h kích thước n, trong đó h(i, j), $1 \le i, j \le n$, bằng k nếu tại bước thứ k, Mã nhảy đến ô (i, j) (xem bước thứ nhất, Mã đứng ở ô xuất phát).



8 vị trí mà Mã có thể nhảy đến

Từ một ô, Mã có thể nhảy đến một trong 8 vị trí xung quanh như hình vẽ, vì thế ta phải thử cả 8 ô này. Ô nhảy đến được chấp nhận nếu nằm trong phạm vi bàn cờ và Mã chưa đi qua.

Ta có thể kiểm soát trạng thái này bằng việc khởi gán h(i, j) bằng 0 với mọi $1 \le i, j \le n$ ngoại trừ h(u, v) bằng 1 (ô xuất phát), ngoài ra ta có thể viền xung quanh bàn cờ 2 cột bên trái (j = 0, −1), 2 cột bên phải (j = n, n + 1), 2 dòng bên trên (i = 0, −1) và 2 dòng bên dưới (i = n, n + 1) và tại những ô "giả" này h được gán bằng một giá trị nào đấy khác 0 (chẳng hạn 1). Khi đó điều kiện chấp nhận ô (x, y) cho Mã là h(x, y) bằng 0.

Thủ tục đệ quy TRY cần tổ chức 3 tham số: (x, y) là vị trí Mã đang đứng và k là số thứ tự của ô mà Mã định nhảy đến. Thủ tục này xác định vị trí tiếp theo của Mã tương ứng với giá trị k. Khi $k = n^2$ thì Mã hoàn tất một hành trình. Lời gọi khởi động thủ tục TRY trong chương trình chính là TRY(u, v, 2). Các vị trí mà Mã có thể nhảy đến từ ô (x, y) được tính theo các độ lệch dòng $dx = (dx_1, dx_2, ..., dx_8)$, độ lệch cột $dy = (dy_1, dy_2, ..., dy_8)$ trong đó các thành phần của chúng là những hằng số được xác định trước. Chẳng hạn, theo cách đánh số trên hình vẽ, ta có dx = (−1, −2, −2, −1, 1, 2, 2, 1) và dy = (2, 1, −1, −2, −2, −1, 1, 2).

```
PROCEDURE TRY (x, y, k: INTEGER);
VAR j: INTEGER;
BEGIN
   FOR j := 1 TO 8 DO
   IF  h(x + dx[j], y + dy[j]) = 0 THEN
   BEGIN
       h(x + dx[j], y + dy[j]) = k;
       IF (k = n*n) THEN OUT ELSE TRY(x + dx[j], y + dy[j], k + 1);
         h(x + dx[j], y + dy[j]) = 0;
   END;
END;
```

Khi chạy chương trình với n = 4, ta nhận được kết quả không có một hành trình nào của Mã đi tuần với mọi vị trí xuất phát. Hành trình Mã đi tuần đầu tiên tìm được (theo thứ tự duyệt được đánh số trên hình vẽ) với n = 5 và vị trí xuất phát (1, 1) là:

1	4	9	18	21
10	17	20	3	8
5	2	13	22	19
16	11	24	7	14
25	6	15	12	23

Với những vị trí xuất phát khác nhau, ta nhận được số hành trình Mã đi tuần khác nhau. Dưới đây là số hành trình Mã đi tuần trên bàn cờ kích thước n = 5 tương ứng với các vị trí xuất phát mà chương trình tính được:

304	0	56	0	304
0	56	0	56	0
56	0	64	0	56
0	56	0	56	0
304	0	56	0	304

Chú ý: Do tính đối xứng ta chỉ cần tính tại những vị trí thuộc 1/8 bảng (nửa trên đường chéo của góc phần tư thứ nhất).

TÓM LƯỢC CUỐI BÀI

Qua bài học, các bạn đã nắm được những nét chính của bài toán liệt kê tổ hợp.

Các bạn cần ghi nhớ các vấn đề sau :

- Các yêu cầu của bài toán liệt kê tổ hợp
- Thuật toán quay lui
- Liệt kê một số cấu hình cơ bản (liệt kê dãy nhị phân; liệt kê hoán vị; liệt kê tổ hợp)
- Một số ứng dụng của bài toán liệt kê

BÀI TẬP

Các bài tập liệt kê từ 1 đến 8 dưới đây đều dựa trên thuật toán quay lui, trình tự liệt kê theo thứ tự từ điển (các giá trị đề cử được xếp tăng dần). Kết quả đếm của những bài này cần được so sánh với những kết quả đếm đã biết bằng công thức.

1. Liệt kê và đếm các chuỗi n bit thưa (là chuỗi n bit không chứa hai bit 1 liên tiếp – xem bài tập 26, bài 2).

a) thực hiện thủ công với n = 4. b) lập trình với n nhập từ bàn phím.

2. Cho k, n là các số nguyên dương ($k \le n$). Liệt kê và đếm các nghiệm nguyên dương của phương trình: $x_1 + x_2 + ... + x_k = n$

a) thực hiện thủ công với k = 3 và n = 5. b) lập trình với n, k nhập từ bàn phím.

3. Một mất thứ tự của {1, 2, ..., n} là một hoán vị $(x_1, x_2, ..., x_n)$ của {1, 2, ..., n} thỏa mãn $x_i \ne i$ với mọi i (xem bài toán bỏ thư trong bài 2). Liệt kê và đếm các mất thứ tự của {1, 2, ..., n} với n cho trước.

a) thực hiện thủ công với n = 4. b) lập trình với n nhập từ bàn phím.

4. Một phân bố của {1, 2, ..., n} là một hoán vị $(x_1, x_2, ..., x_n)$ của {1, 2, ..., n} thỏa mãn $x_i \ne i$ và $x_i \ne i + 1$ với mọi i (chú ý nếu i = n thì i + 1 = 1, xem bài toán xếp khách trong bài 2). Liệt kê và đếm các phân bố của {1, 2, ..., n} với n cho trước.

a) thực hiện thủ công với n = 4. b) lập trình với n nhập từ bàn phím.

5. Liệt kê và đếm tất cả các xâu ký tự độ dài n được lập từ k chữ cái đầu tiên của bảng chữ cái tiếng Anh {A, B, ... }, $k \le n \le 26$, trong đó mỗi chữ cái xuất hiện ít nhất một lần.

a) thực hiện thủ công với n = 3, k = 2. b) lập trình với n, k nhập từ bàn phím.

6. Một xâu ký tự được lập từ n chữ cái đầu tiên của bảng chữ cái tiếng Anh {A, B, ... }, $n \le 26$, trong đó chữ cái thứ i xuất hiện k_i lần ($1 \le k_i \le 9$, i = 1, 2, ..., n). Liệt kê và đếm các hoán vị của xâu đã cho.

a) thực hiện thủ công với n = 3, $k_1 = 2$, $k_2 = 3$, $k_3 = 1$.

b) lập trình với n và các k_i (i = 1, 2, ..., n) nhập từ bàn phím.

7. Cho một họ gồm m tập con $S_1, S_2, ..., S_m$ của tập {1, 2, ..., n} ($m \le n$). Liệt kê và đếm các hệ đại diện phân biệt của họ tập đã cho.

a) thực hiện thủ công với m = 4, n = 4 và $S_1 = \{1, 2\}$, $S_2 = \{1, 2, 3\}$, $S_3 = \{2, 3, 4\}$, $S_4 = \{3, 4\}$.

b) lập trình với m, n và các S_i nhập từ bàn phím, trong đó các S_i được xác định bởi một ma trận nhị phân T gồm m dòng, n cột như sau:

$$T(i, j) = \begin{cases} 1 & j \in S_i \\ 0 & j \notin S_i \end{cases} \qquad 1 \le i \le m,\ 1 \le j \le n$$

Các bài tập dưới đây dùng liệt kê để khẳng định sự tồn tại hay không tồn tại của một cấu hình.

8. a) Viết một chương trình chứng tỏ rằng không thể xếp 2 quân Hậu (không quân nào ăn được quân nào) khống chế được toàn bộ bàn cờ kích thước 5, trong khi tìm được cách xếp 3 quân Hậu thỏa mãn điều kiện này.

b) Từ chương trình trên, viết một chương trình tìm số quân Hậu ít nhất để có thể xếp chúng trên bàn cờ kích thước n (không quân nào ăn được quân nào), sao cho toàn bộ bàn cờ bị khống chế.

9. Một hành trình Mã đi tuần mà từ ô cuối cùng, Mã có thể nhảy về ô xuất phát được gọi là một chu trình Mã đi tuần. Viết một chương trình chứng tỏ rằng với bàn cờ kích thước 5 không tồn tại chu trình Mã đi tuần, và chỉ ra một chu trình Mã đi tuần với bàn cờ kích thước 6.

CÂU HỎI THƯỜNG GẶP

1. Nhiệm vụ chính của bài toán liệt kê tổ hợp?

2. Bản chất của thuật toán quay lui là gì?

3. Hạn chế của bài toán liệt kê là gì?