\documentclass[12pt,a4paper]{article}

\usepackage[utf8]{inputenc}

\usepackage{vntex}

\usepackage{ucs}

\usepackage{amsmath}

\usepackage{amsfonts}

\usepackage{amssymb}

\usepackage{graphicx}

\usepackage{indentfirst}

\begin{document}

\author{Đặng Hữu Tú}

\title{\textbf{Bài toán liệt kê}}

\maketitle

**\section{Giới thiệu bài toán}**

Nếu như trong bài toán đếm, ta chỉ đòi hỏi đếm số cấu hình tổ hợp là bao nhiêu thì trong nhiều tình huống, ta còn phải cần chỉ rõ những cấu hình tổ hợp đó là những cấu hình nào.Bài toán đưa ra danh sách tất cả các cấu hình tổ có thể có được gọi là bài toán liệt kê. Khác với bài toán đếm tìm kiếm một công thức cho lời giải, bài toán liệt kê lại cần xác định một thuật toán để theo đó có thể lần lượt xây dựng tất cả các cấu hình đang quan tâm. Có nhiều cách liệt kê , tuy nhiên phải đảm bảo hai nguyên tắc:

\begin{itemize}

\item Không được lặp lại một cấu hình.

\item Không được bỏ sót một cấu hình.

\end{itemize}

Có thể nói rằng phương pháp liệt kê là cách cuối cùng để có thể giải được một số bài toán tổ hợp hiện nay. Khó khắn chính của phương pháp này là sự "bùng nổ tổ hợp".Trong chương này, chúng ta sẽ trình bày hai phương pháp liệt kê thường sử dụng nhất, đó là thuật toán sinh và thuật toán quay lui.

**\section{Thuật toán và độ phức tạp tính toán}**

**\subsection{Khái niệm thuật toán}**

Định nghĩa:Thuật toán giải bài toán giải bài toán đặt ra là một thủ tục xác định bao gồm một dãy hữu hạn các bước cần thực hiện để thu được lời giải của bài toán.

Thuật toán có các đặc trưng sau đây:

\begin{itemize}

\item Đầu vào(input): thuật toán nhận dữ liệu từ một tập nào đó.

\item Đầu ra(output): với mỗi tập dữ liệu đầu vào, thuật toán đưa ra các dữ liệu tương ứng với lời giải của bài toán.

\item Chính xác(precision):các bước của thuật toán được mô tả chính xác.

\item Hữu hạn(finiteness):thuật toán cần phải đưa được đầu ra sau một số hữu hạn bước với mọi đầu vào.

\item Đơn trị(uniqueness):các kết quả trung gian của từng bước thực hiên thuật toán được xác định một cách đơn trị và chỉ phụ thuộc vào đầu vào và các kết quả của các bước trước.

\item Tổng quát(generality): thuật toán có thể áp dụng để giải mọi bài toán có dạng đã cho.

\end{itemize}

**\subsection{Độ phức tạp của thuật toán}**

**\section{Phương pháp sinh}**

Phương pháp sinh có thể áp dụng để giải bài toán liệt kê tổ hợp đặt ra nếu như hai điều kiện sau được thực hiện:\\

1) Có thể xác định được motjoj thứ tự trên tập các cấu hình tổ hợp cần liệt kê.Từ đó có thể xác định được cấu hình đầu tiên và cấu hình cuối cùng trong thứ tự đã xác định.\\

2) Xây dựng thuật toán từ cấu hình chưa phải là cuối cùng đang có , đưa ra cấu hình kế tiếp nó.

**\subsection\*{3.1.Thuật toán liệt kê các dãy nhị phân độ dài n}**

Viết dãy nhị phân dưới dạng $b\_1b\_2\ldots b\_n$ trong đó $b\_i \in (0,1)$.Xem mỗi dãu nhị phân $b=b\_1b\_2 \ldots b\_n$ là biểu diễn nhị phân của một số nguyên $p(b)$.Khi đó thứ tự hiển nhiên nhất có thể xác địn trên tập các dãy nhị phân là thứ tự tự nhiên( còn gọi là thứ tự từ điển) được xác định như sau: ta có dãy nhị phân $b=b\_1b\_2 \ldots b\_n$ đi trước dãy nhị phân $b^{'}=b\_1^{'}b\_2^{'} \ldots b\_n^{'}$ trong thứ tự tự nhiên là $b<b^{'}$ nếu $p(b)<p(b^{'})$.\\

Như vậy dãy đầu tiên sẽ là $00 \ldots 0$ , còn dãy cuối cùng là $11 \ldots 1$. Giả sử $b\_1b\_2 \ldots b\_n $ là dãy đang có, ta thấy rằng nếu dãy này gồm toàn chữ số 1 thì quá trình liệt kê kết thúc , còn không, dãy kế tiếp sẽ nhận được bằng cách cộng thêm 1 (theo module 2, có nhớ) vào dãy hiện tại . Từ đó ta nhận được quy tắc sinh ngẫu nhiên như sau:

\begin{itemize}

\item Tìm i đầu tiên (theo thứ tự $i=n,n-1, \ldots ,1$) thỏa mãn $b\_i=0$.

\item Gán lại $b\_i=1$ và $b\_j=0$ với tất cả $i>j$. Dãy mới thu được sẽ là dãy cần tìm.

\end{itemize}

Thuât toán sinh kế tiếp được mô tả như sau:

Dưới đây là chương trình C thực hiện việc liệt kê các dãy nhị phân có độ dài n bằng phương pháp sinh:

\begin{verbatim}

# include<stdio.h>

# include<conio.h>

# include<stdlib.h>

# define MAX 100

# define TRUE

# define FALSE

int stop, count;

void Init(int \*B, int n){

int i;

for(i=1;i<=n;i++){

B[i]=0;

count=0;

}

}

void result(int \*B,int n){

int i;

count++;

printf("\nXau nhi phan thu %d:",count);

for(i=1;i<=n;i++){

printf("%3d",B[i]);

}

}

void nextBit(int \*B, int n){

int i=n;

while(i>0&&B[i]==1){

B[i]=0;

i--;

}

if (i==0) stop =TRUE;

else B[i]=1;

}

void genarate(int \*B, int n){

int i;

stop=FALSE;

while(!stop){

result(B,n);

nextBit(B,n);

}

}

void main(void){

int i,\*B,n;

printf("Nhap n="); scanf("%d",&n);

B=(int \*)malloc(n\*sizeof(int));

Init(B,n);

genarate(B,n);

free(B);

getch();

}

\end{verbatim}

**\subsection{Liệt kê các hoán vị của tập n phần tử}**

Bài toán đặt ra là: Cho $ X= \{1,2,\ldots,n\}.$ Hãy liệt kê các hoán vị từ n phần tử của X. Mỗi hoán vị từ n phần tử của X có thể biểu diễn bởi bộ có thứ tự gồm n thành phần $ a=(a\_1,a\_2, \ldots,a\_n)$ thỏa mãn:

\begin{center}

$a\_i \in X, i=1,2, \ldots,n, a\_p \neq a\_q, p\neq.$

\end{center}

Trên tập các hoán vị từ n phần tử của X có thể xác định nhiều thứ tự khác nhau. Thứ tự đơn giản nhất là thứ tự từ điển và kí hiệu là $a<a^{'}$ nếu tìm được chỉ số k $(1<=k<=n)$ sao cho:

\begin{center}

$a\_1=a\_1^{'}, a\_2=a\_2^{'}, \ldots,a\_{k-1}=a\_{k-1}^{'}, a\_k<a\_k{'}$

\end{center}

Giả sử $a=a\_1,a\_2, \ldots,a\_n$ là hoán vị đang có chưa phải cuối cùng ,khi đó có thể chứng minh được rằng , hoán vị kế tiếp trong thứ tự từ điển có thể xây dựng bằng cách thực hiện các quy tắc biến đổi sau với các hoán vị đang có:

\begin{itemize}

\item Tìm từ phải qua trái hoán vị đang có chỉ số j đầu tiên thỏa mãn $a\_j < a\_{j+1}$ (j là chỉ số lớn nhất thoản mãn $a\_j<a\_{j+1};$

\item Tìm $a\_k$ là số nhỏ nhất còn lớn hơn $a\_j$ trong các số từ bên phải $a\_j;$

\item Đổi chỗ $a\_j$ và $a\_k;$

\item Lật ngược đoạn từ $a\_{j+1}$ đến $a\_n;$

\end{itemize}

Chương trình liệt kê hoán vị:

\begin{verbatim}

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

#define TRUE 1

#define FALSE 0

#define MAX 20

int P[MAX],n,k,count,stop;

void Init(){

int i;

count=0;

printf("Nhap n="); scanf("%d",&n);

for(i=1;i<=n;i++){

P[i]=i;

}

}

void Result(){

int i;

count++;

printf("\nHoan vi thu %d:",count);

for(i=1;i<=n;i++){

printf("%3d",P[i]);

}

}

void NextPermutation(){

int j,k,r,s,temp;

j=n-1;

while(j>0&&P[j]>P[j+1]){

j--;

}

if(j==0){

stop=TRUE;

}

else{

k=n;

while(P[j]>P[k]) k--;

temp=P[j];

P[j]=P[k];

P[k]=temp;

r=j+1;

s=n;

while(r<s){

temp=P[r];

P[r]=P[s];

P[s]=temp;

r++; s--;

}

}

}

void Permutation(){

stop=FALSE;

while(!stop){

Result();

NextPermutation();

}

}

void main(){

Init();

Permutation();

getch();

}

\end{verbatim}

**\subsection{Liệt kê các tập con m phần tử của tập n phần tử}**

Bài toán đặt ra là: Cho $X=\{1,2, \ldots,n\}.$ Hãy liệt kê các tập con m phần tử của X. Mỗi tập con m phần tử của X có thể biểu diễn bởi bộ thứ tự gồm m thành phần :

\begin{center}

$ a=(a\_1,a\_2, \ldots a\_m)$

\end{center}

thỏa mãn

\begin{center}

$1 \le a\_1<a\_2< \ldots < a\_m \le n;$

\end{center}

Trên tập các tập các tập con m phần tử của X có thể xác định nhiều thứ tự khác nhau. Thứ tự đơn giản nhất là thứ tự từ điển và kí hiệu là $a<a^{'}$ nếu tìm được chỉ số k $(1<=k<=m)$ sao cho:

\begin{center}

$a\_1=a\_1^{'}, a\_2=a\_2^{'}, \ldots,a\_{k-1}=a\_{k-1}^{'}, a\_k<a\_k{'}$

\end{center}

Như vậy, tập con đầu tiên trong thứ tự từ điển là (1,2,\ldots ,m) và tập con cuối cùng là (n-m+1,n-m+2,\ldots,n).\\

Giả sử $a=(a\_1,a\_2, \ldots,a\_m)$ là tập con đang chưa phải cuối cùng , khi đó có thể chứng minh được rằng, tập con kế tiếp trong thứ tự từ điển có thể được xây dựng bằng cách thực hiện các quy tắc biến đổi sau đối với tập đang có :

\begin{itemize}

\item Tìm từ bên phải dãy $a\_1,a\_2, \ldots ,a\_n$ phần tử $a\_i \neq n-m+i$;

\item Thay $a\_i$ bởi $a\_i+1;$

\item Thay $a\_j $ bởi $a\_i+j-i,$ với $j=i+1,i+2, \ldots,m.$

\end{itemize}

\begin{verbatim}

#include<conio.h>

#include<stdio.h>

#define TRUE 1

#define FALSE 0

#define MAX 100

int n,count,k,stop,C[MAX];

void Init(){

int i;

printf("Nhap n="); scanf("%d",&n);

printf("Nhap k="); scanf("%d",&k);

for(i=1;i<=k;i++){

C[i]=i;

}

}

void Result(){

int i;

count++;

printf("\nTap con thu %d :",count);

for(i=1;i<=k;i++){

printf("%3d",C[i]);

}

}

void NextCombination(){

int i,j;

i=k;

while(i>0&&C[i]==n-k+i){

i--;

}

if(i>0){

C[i]=C[i]+1;

for(j=i+1;j<=k;j++){

C[j]=C[i]+j-i;

}

}

else stop=TRUE;

}

void Combination(){

stop=FALSE;

while(!stop){

Result();

NextCombination();

}

}

void main(void){

Init();

Combination();

getch();

}

\end{verbatim}

Ta nhận thấy không phải cấu hình kế tiếp nào cũng sinh được một cách đơn giản từ cấu hình hiện tại , mặt khác cấu hình ban đầu không phải dễ tìm vì ngay cả sự tồn tại của cấu hình nhiều khi vẫn còn là nghi vấn. Vì vậy,thông thướng thuật toán sinh chỉ có thể xây dựng được đối với những bài toán liệt kê tổ hợp đơn giản .Để giải những bài toán liệt kê phức tạp , người ta thường dùng thuật toán được trình bày trong mục tiếp theo, có tính phổ dụng cao hơn.Đó là thuật toán quay lui.

**\section{Thuật toán quay lui}**

Nội dung của thuật toán này là việc xây dựng dần các thành phần của cấu hình bằng cách thử tất cả các khả năng. Giả thiết cấu hình cần tìm được mô tả gồm một bộ gồm n thành phần $x\_1,x\_2 \ldots x\_n$. Giả sử đã xác định được i-1 thành phần $x\_1,x\_2 \ldots x\_{i-1}$, bây giờ ta sẽ xác định thành phần $x\_i$ bằng cách duyệt tất cả các khả năng có thể đề cử cho nó( đánh số các khả năng từ 1 đến $n\_i$). Với mỗi khả năng j, kiểm tra xem j có chấp nhận được hay không. Xảy ra hai trường hợp:

\begin{itemize}

\item Nếu chấp nhận j thì xác định $x\_i$ theo j, sau đó nếu i=n thì ta được một cấu hình, còn trái lại ta tiến hành việc xác định $x\_{i+1}$.

\item Nếu thử tất cả các khả năng mà không có khả năng nào được chấp nhận thì quay lại bước trước để xác định $x\_i-1$.

\end{itemize}

Điểm quan trọng của thuật toán là phải ghi nhớ tại mỗi bước đã đi qua , những khả năng nào đã thử để tránh trùng lặp . Rõ ràng những thông tin này cần được lưu trừ theo cơ cấu ngăn xếp(stack- vào sau ra trước). Vì thế thuật toán này rất phù hợp với việc lập trình trên một ngôn ngữ cho phép gọi đệ quy. Bây giờ chúng ta sẽ đi vào các bài toán điển hình bằng thuật toán quay lui.

**\subsection{Liệt kê các dãy nhị phân độ dài n}**

Biểu diễn dãy nhị phân dưới dạng $b\_1,b\_2 \ldots b\_n$ trong đó $b\_i \in \{0,1\}$. Thủ tục đệ quy Try(i) xác định $b\_i$, trong đó các giá trị đề cử là 0 và 1. Các giá trị này mặc nhiên được chấp nhận và không phải thỏa mãn điều kiên gì.

Chương trình C liệt kê các dãy nhị phân độ dài n bằng thuật toán quay lui:

\begin{verbatim}

#include<conio.h>

#include<stdio.h>

int n;

int B[100];

int count;

void Init(){

printf("Nhap n=");

scanf("%d",&n);

}

void result(){

int i;

count=count+1;

printf("\nDay thu %d:",count);

for(i=1;i<=n;i++){

printf("%3d",B[i]);

}

}

void try(int i){

int j;

for(j=0;j<=1;j++){

B[i]=j;

if(i==n) result();

else try(i+1);

}

}

void main(){

Init();

try(1);

getch();

}

\end{verbatim}

**\subsection{Liệt kê các hoán vị của $ \{1,2,\ldots n\}$}**

Biểu diễn hoán vị dưới dạng $p\_1,p\_2, \ldots p\_n$ trong đó $p\_i$ nhận giá trị từ 1 đến n và $p\_i \neq p\_j$ với $i \neq j$. Các giá trị từ 1 đến n lần lượt đề cử cho $p\_i$, trong đó giá trị j được chấp nhận nếu nó chưa được dùng. Vì thế, cần phải đối với mỗi giá trị j xem nó đã được dùng hay chưa. Điều này được thực hiện nhờ một dãy biến logic $b\_j$, trong đó $b\_j$ bằng true nếu j chưa được dùng. Các biến cần phải được khởi gán giá trị true trong hàm Init. Sau khi gán j cho $p\_i$ cần ghi nhận false cho $b\_j$ và phải gán lại true khi thực hiện xong

Result hay Try(i+1). Các phần còn lại giống như bài toán trước.

Chương trình C liệt kê các hoán vị của $ \{1,2,\ldots n\}$ bằng thuật toán quay lui:

\begin{verbatim}

#include<conio.h>

#include<stdio.h>

#define TRUE 1

#define FALSE 0

int B[100],var[100],n;

int count=0;

void Process(int var[],int n){

printf("\nHoan vi thu %d:",count);

int i;

for(i=1;i<=n;i++){

printf("%3d",var[i]);

}

printf("\n");

}

void Try(int i, int n){

int j;

if(i==n+1){

count++;

Process(var,n);

}

for(j=1;j<=n;j++){

if(B[j]){

var[i]=j;

B[j]=FALSE;

Try(i+1,n);

B[j]=TRUE;

}

}

}

void main(){

printf("Chuong trinh liet ke cac hoan vi\n");

printf("Nhap so phan tu cua tap:");

scanf("%d",&n);

int dem;

for(dem=1;dem<=n;dem++) B[dem]=TRUE;

Try(1,n);

getch();

}

\end{verbatim}

**\subsection{Liệt kê các tổ hợp chập m của n $\{1,2,\ldots,n\}$}**

Biểu diễn tổ hợp dưới dạng $c\_1,c\_2,\ldots ,c\_m$ trong đó:

\begin{center}

$1 \le c\_1 \le c\_2 \le \ldots c\_m \le n$

\end{center}

Từ đó suy ra các giá trị đề cử cho $c\_i$ là $c\_{i-1}$ đến n-m+i

Để điều này đúng cho cả trường hợp i=1, cần thêm vào $c\_0$ với $c\_0$=0. Chương trình C liệt kê các tổ hợp chập m của $\{1,2, \ldots, n\}: $

\begin{center}

\begin{verbatim}

#include<conio.h>

#include<stdio.h>

int m,n,count;

int C[20];

void Init(){

printf("Nhap n="); scanf("%d",&n);

printf("Nhap m="); scanf("%d",&m);

C[0]=0;

count=0;

}

void result(){

int i;

count=count+1;

printf("\nTo hop chap %d thu %d la:",m,count);

for(i=1;i<=m;i++){s

printf("%3d",C[i]);

}

}

void try(int i){

int j;

for(j=C[i-1]+1;j<=n-m+i;j++){

C[i]=j;

if(i==m) result();

else try(i+1);

}

}

void main(){

Init();

try(1);

getch();

}

\end{verbatim}

\end{center}

**\subsection{Liệt kê các chỉnh hợp không lặp chập k của tập $S=\{1,2, \ldots, n\}$}**

Thủ tục try(i) xét tất cả các khả năng chọn $x\_i$, thử hết các giá trị từ 1 đến n mà các giá trị này chưa bị các phần tử đứng trước chọn. Muốn xem những giá trị nào chưa được chọn, ta dùng mảng đánh dấu:\\

\begin{itemize}

\item Khởi tạo một mảng $c\_1,c\_2,\ldots,c\_n$ mang kiểu logic. Ở đây $c\_i$ cho biết giá trị i còn tự do hay đã bị chọn rồi. Ban đầu khởi tạo tất cả các phần tử mảng c là TRUE, nghĩa là các phần tử từ đến n đều tự do.

\item Tại bước chọn các giá trị có thể của $x\_i$ ta chỉ xét giá trị j có $c\_j$=TRUE có nghĩa là chỉ chọn những giá trị tự do.

\item Trước khi gọi đệ quy $x\_{i+1}$, ta đặt giá trị j vừa gán cho $x\_i$ đã bị chọn, nghĩa là đặt x$c\_j$=FALSE để các thủ tục try(i+1), try(i+2)... gọi sau này không phải chọn giá trị j nữa.

\item Sau khi gọi đệ quy tìm $x\_{i+1}$: $c\_j$=TRUE

\end{itemize}

Chương trình C liệt kê các chỉnh hợp không lặp chập k của của tập $S=\{1,2, \ldots n\}$ bằng phương pháp quay lui:\\

\begin{verbatim}

#include<conio.h>

#include<stdio.h>

int X[20],C[20];

int n,k,count=0;

void Init(){

int i;

printf("Nhap n=");

scanf("%d",&n);

printf("Nhap k=");

scanf("%d",&k);

for(i=1;i<=n;i++){

C[i]=1;

}

}

void result(){

int i;

count++;

printf("\nChinh hop thu %d:",count);

for(i=1;i<=k;i++){

printf("%3d",X[i]);

}

}

void try(int i){

int j;

for(j=1;j<=n;j++){

if(C[j]){

X[i]=j;

C[j]=0;

if(i==k) result();

else try(i+1);

C[j]=1;

}

}

}

void main(){

Init();

try(1);

getch();

}

\end{verbatim}

**\subsection{Bài toán phân tích số}**

\textbf{Bài toán}:Cho một số nguyên dương n, hãy tìm tất cả cách phân tích số n thành tổng của các số nguyên dương,các phân tích là hoán vị của nhau chỉ tính là một cách.\\

\textbf{Giải:}\\

\begin{enumerate}

\item Ta sẽ lưu nghiệm trong mảng X, ngoài ra có một mảng T. Mảng T xây dựng như sau: $T\_i$ sẽ là tổng các phần tử trong mảng X từ $X\_1$ đến $X\_i$: $T\_i= \sum\_1^i X\_i$

\item Khi liệt kê dãy các X có tổng các phần tử đúng bằng n, để tránh sự trùng lặp ta đưa thêm điều kiện $X\_{i-1} \le X\_i$.

\item Vì số phần tử thực sự của mảng X là không cố định lên thủ tục result dùng để in ra một cách phân tích phải có thêm tham số cho biết sẽ in ra bao nhiêu phần tử.

\item Thủ tục đệ quy try(i)sẽ thử các giá trị có thể nhận của $X\_i (X\_{i-1} \le X\_i)$.

\item In ra kết quả hoặc gọi đệ quy tiếp:

\begin{itemize}

\item Khi $T\_i$=n tức là $X\_i=n-T\_i$ thì in ra kết quả

\item Khi tìm tiếp $X\_{i+1}$, thì $X\_i \le X\_{i+1}$. Mặt khác,

$T\_{i+1}=\sum\_1^{i+1} \le n$. Vậy ta có $T\_{i+1} \le n => T\_{i-1}+X\_i+X\_{i+1} \le n => X\_i+X\_{i+1} \le n-T\_{i-1}=>

X\_i \le \frac{n-T\_{i-1}}{2}$

\end{itemize}

\item Thủ tục try(i) thử các giá trị $X\_i$ mô tả như sau(để tổng quát cho i=1 ta đặt $X\_0$=1 và $T\_0=0$):

\begin{itemize}

\item Xét các giá trị của $X\_i$ từ $X\_{i-1}$ đến $\frac{n-T\_{i-1}}{2}$, cập nhật $T\_i=T\_{i-1}+X\_i$ và tìm gọi đệ quy tiếp.

\item Cuối cùng xét giá trị $X\_i=n-T{i-1}$ và in ra kết quả từ $X\_i$ đến $X\_i$

\end{itemize}

\end{enumerate}

Chương trìn C liệt kê các cách phân tích số n không tính hoán vị bằng phương pháp quay lui:\\

\begin{verbatim}

#include<conio.h>

#include<stdio.h>

int n;

int X[30],T[30],count=0;

void Init(){

printf("Nhap n=");

scanf("%d",&n);

X[0]=1;

T[0]=0;

}

void result(int k){

int i;

count++;

printf("\nCach chia thu %d:",count);

for(i=1;i<=k-1;i++){

printf("%3d",X[i]);

}

printf("%3d",X[k]);

}

void try(int i){

int j;

for(j=X[i-1];j<=(n-T[i-1])/2;j++){

X[i]=j;

T[i]=T[i-1]+j;

try(i+1);

}

X[i]=n-T[i-1];

result(i);

}

void main(){

Init();

try(1);

getch();

}

\end{verbatim}

**\subsection{Bài toán xếp hậu}**

Dưới đây là bài toán Xếp Hậu , là một bài toán kinh điển để minh họa thuật toán quay lui:

\textbf{Bài toán Xếp Hậu:} Liệt kê tất cả các cách xếp n quân Hậu trên bàn cờ n x n sao cho chúng không ăn được lẫn nhau.

\textbf{Giải:}Đánh số cột và số dòng từ 1 đến n. Mỗi dòng được xếp đúng một quân Hậu. Bây giờ, ta xem mỗi quân Hậu được xếp vào cột nào. Từ đó, dẫn đến việc biểu diễn một cách xếp bằng bộ n thành phần $x\_1, x\_2, \ldots ,x\_n$, trong đó $x\_i=j$ nghĩa là quân hậu dòng i được xếp vào cột j. Các giá trị đề cử cho $x\_i$

là từ 1 đến n. Giá trị j được chấp nhận nếu ô (i,j) chưa bị các

quân Hậu khác chiếu đến. Để kiểm tra được điều này,ta cần phải ghi nhận trạng thái của bàn cờ trước cũng như sau khi xếp được một con Hậu. Theo luật bàn cờ , quân Hậu ăn ngang, dọc và hai đường chéo.

Việc kiểm tra theo hàng ngang là không cần thiết vì mỗi dòng được xếp đúng một con Hậu.Việc kiểm soát theo chiều dọc được ghi nhận nhờ dãy biến \emph{lôgic} $a\_j$ với quy ước $a\_j$ bằng

\emph{true} nếu cột j còn trống. Đối với hai đường chéo ta thấy rằng một đường có phương trình \emph{i+j=const },còn đường kia \emph{i-j=const} $(2 \le i+j \le 2n, 1-n \le i-j \le n-1$. Từ đó, đường chéo thứ nhất được ghi nhận nhờ dãy biến \emph{lôgic}

$b\_j, ( 2 \le j \le 2n)$ và đường chéo thứ hai nhờ dãy biến \emph{lôgic} $c\_j (1-n \le j \le n-1)$ với quy ước các đường này còn trống nếu biến tương ứng có giá trị \emph{true}.Các biến trạng thái cần phải được khởi gán giá trị \emph{true} trong thủ tục \emph{Init}. Giá trị j được chấp nhận khi và chỉ khi cả ba biến $a\_j,b\_{i+j},c\_{i+j}$ có cùng giá trị \emph{true}. Các biến này cần gán lại \emph{false} khi xếp xong quân Hậu thứ i và trả lại \emph{true} sau khi gọi \emph{result}

hay \emph{try(i+1)}. Chương trình C giải bài toán xếp Hậu:

\begin{verbatim}

#include<conio.h>

#include<stdio.h>

int n, count=0;

int X[20],A[20],B[40],C[40];

void Init(){

int i;

printf("Nhap n="); scanf("%d",&n);

for(i=1;i<=n;i++){

A[i]=1;

}

for(i=2;i<=2\*n;i++){

B[i]=1;

}

for(i=1-n;i<=n-1;i++){

C[i]=1;

}

}

void result(){

int i;

count++;

printf("\nCach xep thu %d:",count);

for(i=1;i<=n;i++){

printf("%3d",X[i]);

}

}

void try(int i){

int j;

for(j=1;j<=n;j++){

if(A[j]==1&&B[i+j]==1&&C[i-j]==1){

X[i]=j;

A[j]=0;

B[i+j]=0;

C[i-j]=0;

if(i==n) result();

else try(i+1);

A[j]=1;

B[i+j]=1;

C[i-j]=1;

}

}

}

void main(){

Init();

try(1);

getch();

}

\end{verbatim}

**\section{Bài tập áp dụng}**

\begin{enumerate}

\item Hai đội bóng chuyền A,B thi đấu trong một giải vô địch quốc gia. Đội thắng trong trận đấu sẽ là đội giành được ba hiệp thắng trước. Hãy liệt kê tất cả các khả năng có thể có của trận đấu giữa hai đội

\item Liệt kê tất cả cách cách mất thứ tự của n số tự nhiên 1,2 \ldots , n.

\item Liệt kê các xâu nhị phân độ dài 5 không chứa hai số 0 liên tiếp

\item Liệt kê các phần tử của:

\begin{center}

$D=\{x=(x\_1,x\_2, \ldots, x\_n)| \sum\_{j=1}^{n} ,x\_j=\{0,1\},j=1,2,\ldots,n\}$

\end{center}

trong đó $a\_1,a\_2, \ldots,a\_n , b $ là các số nguyên dương .

\item Liệt kê các phần tử của:

\begin{center}

$D=\{x=(x\_1,x\_2, \ldots, x\_n)| \sum\_{j=1}^{n},x\_j\in Z\_+,j=1,2,\ldots,n\}$

\end{center}

trong đó $a\_1,a\_2,\ldots,a\_n ,b$ là các số nguyên dương ,$Z\_+ $

là tập số nguyên không âm .

\item Cho tập $X=\{1,2,\ldots,n\}$.Hãy liệt kê tất cả các phân hoạch tập X ra thành k tập con $X\_1,X\_2,\ldots,X\_k(k \le n).$

\end{enumerate}

\end{document}