**Bài tập 4: Phương pháp thử sai (Quay lui, Nhánh cận)**

**2.1**

Biểu diễn tập n phần tử như là tập n số tự nhiên và tìm các hoán vị của tập này.

**Phân tích:**

* Ta sinh lần lượt các hoán vị có dạng .Sử dụng chương trình đệ quy Hoanvi(i) để sinh các phần tử thứ của hoán vị.Ta sẽ sử dụng mảng chọn để đánh dấu các phần tử được chọn.
* nếu k chưa có mặt trong dãy hoán đang xây dựng.
* nếu k đã có mặt trong dãy hoán vị đang xây dựng.
* Khởi tạo:
* Gán các giá trị của các mảng đánh dấu bằng 0.
* Biến đếm count (biến đếm số hoán vị ) =0
* Gọi chương trình đệ quy Hoanvi(1)
* Hàm đệ quy
* Nếu thì
* Nếu .Ta gán cho j các giá trị từ 1 đến n nếu thì gán và gọi đệ quy để xác định .Khi ra khỏi lời gọi đệ quy , trả lại các giá trị cho

**Mô tả dữ liệu**

* số phần tử
* mảng đánh dấu các phần tử đã chọn
* Mảng đánh dấu các phần tử đã được chọn
* : biến đếm số hoán vị

***Mã giải (Pseudo Code)***

*HoanVi(i){*

*if(i=n+1){*

*In dãy hoán vị;*

*Count=count+1;*

*}*

*else{*

*for j=1 to n do*

*if chonj=0{*

*si=j;*

*chonj=1;*

*Hoanvi(i+1);*

*chon­­j=0;*

*}*

*}*

*}*

*Main(){*

*Nhập số n;*

*for i:=1 to n do choni=0;*

*count:=0;*

*HoanVi(1);*

*Xuất số lượng các hoán vị.count;*

*}*

**2.2**

Liệt kê các tổ hợp k phần tử của các số từ 1 đến n.

**Phân tích**

* Ta cần sinh các tổ hợp k phần tử của tập {1, 2, … n}
* Ta sinh các tổ hợp có dạng
* Khởi tạo
* Phần tử ­, biến dếm count = 0, lời gọi ban đầu ToHop(1)
* Giả sử sau bước thứ i – 1, ta đã có 1 dãy bây giờ ta thực hiện bước i (lời gọi đên quy ToHop(i))

+ Nếu i = k + 1, in ra giá trị tổ hợp:

+ Nếu i < k + 1, cho nhận lần lượt các giá trị

Với mỗi giá trị của ta lại thực hiện lời gọi đệ quy ToHop(i + 1)

**Mô tả dữ liệu**

* n: số tự nhiên n
* k: số phần tử của tổ hợp
* Màng s[1..100] lưu trữ giá trị của tổ hợp
* count: biến đếm số tổ hợp

***Mã giải (Pseudo Code)***

*ToHop(i){*

*If(i = k + 1) In dãy tổ hợp ra; count++;*

*else*

*for j = to do*

*ToHop(i + 1);*

*}*

*main(){*

*Nhập số n, k;*

*count = 0;*

*ToHop(1);*

*Xuất giá trị của count;*

*}*

**2.3**

**Thuật toán sau nhằm giải bài toán tổng quát:**

“Đặt n quân hậu lên bàn cờ nxn, sao cho không có quân hậu nào khống chế quân hậu nào”

**Phân tích:**

Ta đánh số các hàng (r), và các cột (c) lần lượt theo thứ tự từ trên xuống dưới và từ trái sang phải là 1,2,…, n

Mỗi ô trên bàn cờ được xác định bởi cặp toạ độ hàng và toạ độ cột (r, c)

r : toạ độ hàng

c : toạ độ cột

Bàn cờ nxn ngoài các hàng, các cột còn bao gồm các đường chéo:

* Đường chéo trên (up) : ví dụ ở bàn cờ 4x4 trên là các đường tô màu cam, xanh, tím. Có 2n – 1 đường chéo trên.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

Đường chéo thứ nhất là 1 ô (1, 1)

Đường chéo thứ hai là 2 ô (1, 2) và (2, 1)

Đường chéo thứ ba là 3 ô (1, 3); (2, 2); (3, 1)

…

Đường chéo thứ (2n – 1) là 1 ô (n, n)

Nhận xét với mỗi đường chéo, tổng các toạ độ của các ô là bằng nhau.Ví dụ với đường chéo thứ ba thì tổng các toạ độ của các ô là bằng nhau. Ví dụ với đường chéo thứ 3 thì tổng

r + c = 1 + 3 = 2 + 2 = 3 + 1 = 4

Như vậy ta sẽ kí hiệu mỗi đường chéo trên theo tổng (r + c) của các ô nằm trên đường chéo đó.

* Đường chéo dưới (down):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

Tương tự mỗi đường chéo dưới có thể đặc trưng bởi (r – c + n) với (r, c) là toạ độ hàng, cột của các ô nằm trên đường chéo đó.

Nhận xét :

1 quân hậu đặt ở ô (r, c) sẽ khống chế:

Hàng r

Cột c

Đường chéo trên (r + c)

Đường chéo dưới (r – c + n)

Dựa trên những nhận xét trên, ta xây dựng thuật toán đệ quy, sử dụng phương pháp “quay lui” để giải bài toán :

Hàm đệ quy PutQueen(r) nhằm tìm cách đặt quân hậu trên hàng r

* Lời gọi đầu tiên PutQueen(1);
* Giả sử ta đã đặt r – 1 quân hậu lên các hàng 1, 2, …, r – 1; bây giờ ta xét cách đặt quân hậu vào hàng thứ r (trong lời gọi đệ quy PutQueen(r))
  + Nếu r = n + 1 thì việc đặt n quân hậu đã hoàn thành, ta in kết quả ra, cho biến đếm số cách đặt count tăng lên 1.
  + Nếu r < n + 1 cho biến c chạy từ 1 -> n. Kiểm tra xem ô (r, c) có đặt được hậu hay không bằng hàm Test(r, c); nếu Test(r, c) = 0 ta đặt quân hậu vào ô này, sử dụng hàm Insert(r , c) đánh dấu ô này đã có quân hậu, Sau đó ta xét cách đặt hậu vào hàng r + 1 bằng lời gọi PutQueen(r + 1). Sauk hi ra khỏi PutQueen(r + 1), gỡ bỏ quân hậu khỏi ô (r, c) bằng hàm Remove(r, c).

Các biến:

* Test\_col[c] : đánh dấu xem cột c đã bị quân hậu nào khống chế chưa
  + Test\_col[c] = 0 nếu chưa bị khống chế
  + Test\_col[c] = 1 nếu đã bị khống chế
* Test\_up[r + c] đánh dấu đường chéo trên có ô có tổng toạ độ (r + c) đã bị khống chế chưa. Nó bằng 0 nếu chưa bị và bằng 1 nếu có 1 quân hậu nào đó khống chế.
* Tương tự Test\_down[r – c + n]: đánh dấu đường chéo dưới r – c + n

Các hàm kiểm tra:

* Test(r, c) = 0: ô (r, c) có thể đặt được hậu nếu:

Ngược lại ta gán Test(r, c) = 1

* Đánh dấu ô (r, c) đã đặt hậu: ta gán lại các giá trị
* Gỡ quân hậu ra khỏi ô (r, c) được thực hiện bằng hàm Remove(r, c) ta gán lại các giá trị

*Test\_col[c] = 0;*

*Test\_up[r + c] = 0;*

*Test\_down[r – c + n] = 0;*

*Pseudo code:*

*Test(r, c){*

*If(Test\_col[c] = 0 & Test\_up[r + c] = 0 &Test\_down[r – c + n] = 0)*

*Return 0;*

*Else return 1;*

*}*

*Insert(r, c){*

*Test\_col[c] = 1;*

*Test\_up[r + c] = 1;*

*Test\_down[r – c + n] = 1;*

*}*

*Remove(r, c){*

*Test\_col[c] = 0;*

*Test\_up[r + c] = 0;*

*Test\_down[r – c + n] = 0;*

*}*

*PutQueen(r){*

*If (r = n + 1){*

*In kết quả;*

*count = count + 1;*

*}*

*Else {*

*For c = 1 to n do*

*If (Test(r, c) = 0){*

*Insert(r, c);*

*PutQueen(r + 1);*

*Remote(r, c);*

*}*

*}*

*}*

*main(){*

*Nhập số hàng số cột : n;*

*//Khởi tạo*

*For c = 1 to n do Test\_col[c] = 0;*

*For r + c = 2 to 2n do Test\_up[r + c] = 0;*

*For r – c + n = 1 to 2n – 1 do Test\_down[r – c + n] = 0;*

*PutQueen(1);*

*Xuât số cách đếm count;*

*}*

**3.**

**Phương pháp quay lui:**

**Bài toán số 49:** (bài toán do nhóm 1 phát minh ra) bài toán yêu cầu tìm số tự nhiên nhỏ nhất, có ít nhất k chữ số chỉ gồm toàn các chữ số 1,2,3 mà chia hết cho 49.

**Phân tích:**

* Gọi s là số tự nhiên cần tìm, n là số chữ số của s;
* Ta lần lượt xét n=k,k+1,k+2,… cho đến khi nào tìm được s thì thôi;
* Với mỗi n, ta xây dựng tất cả các số s có dạng bằng kĩ thuật quay lui. Nếu số s vừa xây dựng mà chia hết cho 49 thì dừng chương trình và kết luận đó là số tự nhiên nhỏ nhất cần phải tìm; ngược lại nếu ta đã liệt kê tất cả các dạng trên mà không tìm được số nào chia hết cho 49 thì ta tăng n lên 1 đơn vị và lặp lại tìm kiếm trên.

**Mã chương trình :** (viết bằng C++)

*/\**

*Name: So 49*

*Copyright: Nhom 1*

*Author: Nhom 1*

*Date: 26/03/12 23:45*

*Description: tìm so tu nhien nho nhat chi gom toan chu so 1,2,3 ma chia het cho 49*

*\*/*

*#include<stdio.h>*

*#include<stdlib.h>*

*#include<conio.h>*

*#include<iostream.h>*

*#include<math.h>*

*int s=0;//So gom toan cac chu so 1,2,3 va chia het cho 49 ma ta phai tim*

*int k=5;//Gioi han duoi cho so chu so cua s*

*int n;//So chu so cua s*

*int kt;//bien kiem tra: kt=0 neu chua tim duoc so nho nhat thoa man, bang 1 neu nguoc lai*

*int status;*

*void SinhChuSo(int i)//xet chu so thu i*

*{*

*if (i==(n+1)) {}//So s da du n chu so*

*else*

*for (int a=1; a<=3; a++){//Xet cac kha nang cua chu so thu i*

*s=s\*10+a;*

*if (s%49 ==0) {//Da tim duoc so s thoa man yeu cau de bai*

*cout<<"So nho nhat thoa man la "<<s;*

*kt=1;*

*status=getch();//thoat khoi chuong trinh*

*exit(status-'0');*

*}*

*else SinhChuSo(i+1);//xet chu so thu i+1*

*s=(int)((s-a)/10);*

*}*

*}*

*int main()*

*{*

*n=k;//bat dau voi so co k chu so*

*kt=0;*

*while (kt==0){*

*s=0;*

*SinhChuSo(1);*

*n++;*

*}*

*getch();*

*return 0;*

*}*

**Đánh giá:**

Trong trường hợp không tốt, số s thỏa mãn có số chữ số lớn hơn k, ta phải xét ít nhất là số s có dạng . Như vậy có thể coi độ phức tạp của bài toán này là .