BÀI 3

BÀI TOÁN LIỆT KÊ

Giáo viên: TS. Nguyễn Văn Hiệu
Email: nvhieuqt@dut.udn.vn

1.1. Giới thiệu

Nội dung

- ✓ 1.1. Giới thiệu
- ✓ 1.2. Phương pháp sinh
- ✓ 1.3. Phương pháp quay lui

1.1. Giới thiệu

Mục đích

- Liệt kê danh sách tất cả các cấu hình

Bản chất

- Xây dựng thuật toán:
  - Dữ liệu vào: cấu hình hiện tại.
  - Dữ liệu ra: cấu hình kế tiếp.

1.1. Giới thiệu

Nguyên tắc

- Không được lặp.
- Không được bỏ sót.

Lưu ý

- Chưa có phương pháp giải:
  - ✓ **Phương pháp Sinh**
  - ✓ **Phương pháp quay lui**

1.2. Phương pháp sinh

Sử dụng

- Giải bài toán liệt kê

Điều kiện

- Xác định được một thứ tự.
  - Xác định cấu hình đầu tiên
  - Xác định cấu hình cuối cùng
- Xây dựng được thuật toán

1.2. Phương pháp sinh

Bản chất

```
Generating_method(…)  {

  <Cấu hình ban đầu>;
  stop = islastconfigure(…);
  while (stop==0) {
    <Cấu hình hiện thời >
     <Sinh_kế_tiếp>;
      }
}
```

Chú thích

- Stop = =1, là cấu hình cuối cùng
- Stop == 0, chưa phải là cấu hình cuối cùng
- <Sinh_kế_tiếp> là thuật toán sinh cấu hình tiếp theo trên thứ tự đã xác định trước

1.2. Phương pháp sinh

Ứng dụng

- Liệt kê dãy nhị phân có độ dài n.
- Liệt kê các tập con k phần tử của tập n phần tử.
- Liệt kê các hoán vị của tập n phần tử

1.2. Phương pháp sinh

Ví dụ 1:

Liệt kê các dãy nhị phân có độ dài n

Bước 1: Xác định thứ tự

- Dãy nhị phân được biểu diễn:

$$b = (b_1\ b_2\ \ldots\ b_n)$$

thỏa mản $b_i \in \{0,1\}$

- Định nghĩa thứ tự từ điển:

$b = (b_1\ b_2\ \ldots\ b_n)$ và $*b = (*b_1\ *b_2\ \ldots\ *b_n)$

thứ tự b < *b,

nếu $q(b) < q(*b)$

1.2. Phương pháp sinh

Ví dụ 1:

Liệt kê các dãy nhị phân có độ dài n

Bước 2: Cấu hình đầu và cuối

- ✓ Khi n = 4 phần tử, có 2^4 dãy nhị phân được liệt kê

1.2. Phương pháp sinh

Ví dụ 1:

Liệt kê các dãy nhị phân có độ dài n

Bước 2:

b	q(b)	b	q(b)
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

1.2. Phương pháp sinh

Bước 3: xác định thuật toán

0000	0001
0001	0010
0011	0100
0111	1000

Thuật toán

- ✓ Tìm i từ bên phải: $b_i = 0$.
- ✓ Gán lại $b_i = 1$ và $b_j = 0$ với mọi $j > i$.

```
i= n;
while (i>=1 && b[i]=='1')
        b[i--] = '0';
b[i] = '1';
```

1.2. Phương pháp sinh

```
#include <stdio.h>
int n,stop,count;
void Init(char b[]){
   count = 1;
   int i;
   for(i=1;i<=n;i++) b[i]='0';
}
void Out(char b[]){
   printf("\n%d:=  ",count);
   int i;
   for(i=1;i<=n;i++) printf("%c",b[i]);
   }
void Next_bit_string(char b[]){
   int i=n;
   while(i>=1&&b[i]=='1') b[i--]='0';
   b[i]='1';
   }
```

1.2. Phương pháp sinh

```
int Lastconfigure(char b[]){
   int i;
   for(i=1;i<=n;i++){ if(b[i]=='0') return 0;}
   return 1;
   }
void Genarate(char b[]){
   Init(b); Out(b);
   stop =Lastconfigure(b);
   while(stop==0){
   count++;
   Next_bit_string(b); Out(b);
   stop = Lastconfigure(b);
   }}
main(){
   printf("n = ");scanf("%d",&n);
   char b[n];
   Genarate(b);
   getch();
   }
```

1.2. Phương pháp sinh

Ví dụ 1:

Liệt kê các dãy nhị phân có độ dài n

Kết quả	**Chương trình**
Result	**Source Code**

1.2. Phương pháp sinh

Ví dụ 2

Liệt kê các tập con k phần tử của tập n phần tử

Chuẩn bị

- ✓ Ánh xạ tập n phần tử vào tập

  $$X = \{1,2,\ldots,n\}$$

- ✓ Mỗi tập con k phần tử của X được biểu diễn bởi bộ có thứ tự gồm k thành phần:

  $$a = (a_1\ a_2\ a_3\ \ldots..\ a_k\ )$$

  thỏa mãn $1 \leq a_1 < a_2 < a_3 < \ldots.. < a_k \leq n$

1.2. Phương pháp sinh

Ví dụ 2

Liệt kê các tập con k phần tử của tập n phần tử

Bước 1: Xác định thứ tự

- ✓ Định nghĩa thứ tự từ điển:

$$a = (a_1\ a_2\ \ a_3\ \ldots a_k) \text{ và } b = (b_1\ b_2\ b_3\ \ldots\ b_k)$$

thứ tự a < b, nếu tồn tại **j** $(1 \leq j \leq k)$:

$$a_1 = b_1, \ldots, a_{j-1} = b_{j-1},\ a_j < b_j$$

1.2. Phương pháp sinh

Ví dụ 2:

Liệt kê các tập con k phần tử của tập n phần tử

Bước 2:

- ✓ Các tập con 3 phần tử của tập 5 phần tử **{1,2,3,4,5}**
- ✓ $C^3_5 = 10$

{ 1,2,3 }
{ 1,2,4 }
{ 1,2,5 }
{ 1,3,4 }
{ 1,3,5 }
{ 1,4,5 }
{ 2,3,4 }
{ 2,3,5 }
{ 2,4,5 }
{ 3,4,5 }

Cấu hình đầu

Cách sinh

Cấu hình cuối

1.2. Phương pháp sinh

Bước 3: xác định thuật toán

n=5, k=3

i =	1	2	3
a =	**{1,**	**4,**	**5 }**
n-k+i =	$\boxed{3}$	4	5

{ $\boxed{2}$, , }

{ 2, $\boxed{3, 4}$ }

Thuật toán

- Giả sử $a = (a_1 \ldots a_k)$ không là cuối cùng.
  - ✓ B1: Tìm vị trí i đầu tiên từ bên phải của dãy: $\mathbf{a_i \neq n\text{-}k\text{+}i}$.
  - ✓ B2: Thay $\mathbf{a_i}$ bởi $\mathbf{a_i + 1}$.
  - ✓ B3: Thay $\mathbf{a_j}$ bởi $\mathbf{a_i - i + j}$, với $j = i+1, \ldots, k$

1.2. Phương pháp sinh

Thuật toán

- Giả sử $a = (a_1 \ldots a_k)$ không là cuối cùng.
- ✓ B1: Tìm vị trí i đầu tiên từ bên phải của dãy: $\mathbf{a_i \neq n\text{-}k\text{+}i.}$
- ✓ B2: Thay $\mathbf{a_i}$ bởi $\mathbf{a_i + 1.}$
- ✓ B3: Thay $\mathbf{a_j}$ bởi $\mathbf{a_i - i + j}$, với $j = i+1,\ldots,k$

Code

B1:

```
i=k;
while (a[i]==n-k+i) i-- ;
```

B2:

```
a[i]= a[i] + 1;
```

B3:

```
for (j=i+1;j<=k;j++)
    a[j]= a[i]+ j –i;
```

1.2. Phương pháp sinh

```
#include <stdio.h>
int n,stop;
void Init(int a[],int k){
   int i;
   for(i=1;i<=k;i++)a[i]=i;
   }
void Out(int a[],int k){
   int i; printf("\n");
   for(i=1;i<=k;i++) printf("%d ",a[i]);
   }
int lastconfigure(int a[],int k)
{
   int i;
   for(i=k;i>=1;i--) if(a[i]!=n-k+i) return 0;
   return 1;
   }
```

1.2. Phương pháp sinh

```
void next_combination(int a[],int k){
   int i=k;
   while(i>=1&&a[i]==n-k+i) i--;
   a[i]++;
   int j;
   for(j=i+1;j<=k;j++)a[j]=a[i]+j-i;
   }
void genarate(int a[],int k){
   Init(a,k); Out(a,k);
   stop = lastconfigure(a,k);
   while(stop==0){
   next_combination(a,k);Out(a,k);
   stop = lastconfigure(a,k);
   } }
main(){
   int k; printf("pt me n:=");scanf("%d",&n);
   printf(" pt con k:=");scanf("%d",&k);
   int a[k]; genarate(a,k); getch();
   }
```

1.2. Phương pháp sinh

Ví dụ 2

Liệt kê các tập con k phần tử từ tập n phần tử

Kết quả

Result

Chương trình

Source Code

1.2. Phương pháp sinh

Ví dụ 3

Liệt kê hoán vị của tập n phần tử

1.2. Phương pháp sinh

Minh họa

{1,2,3,4}
{1,2,4,3}
{1,3,2,4}
{1,3,4,2}
{1,4,2,3}
{1,4,3,2}
{2,1,3,4}
{2,1,4,3}
{2,3,1,4}
{2,3,4,1}
{2,4,1,3}
{2,4,3,1}
{3,1,2,4}
{3,1,4,2}
{3,2,1,4}
{3,2,4,1}
{3,4,1,2}
{3,4,2,1}
{4,1,2,3}
{4,1,3,2}
{4,2,1,3}
{4,2,3,1}
{4,3,1,2}
{4,3,2,1}

- ✓ **{1,2,3,4} ?**
- ✓ **{4,3,2,1}?**
- ✓ {3,4,2,1} ➔ {4,1,2,3}?
- ✓ **Xây dựng thuật toán sinh**

1.2. Phương pháp sinh

Định nghĩa

- Ánh xa tập n phần tử bất kỳ vào tập

  $X = \{1,2,\ldots,n\}$

- Mỗi hoán vị của X được biểu diễn bởi bộ có thứ tự gồm n thành phần:

  $a = (a_1\ a_2\ \ldots\ a_n)$

  thỏa mản $a_i \in X$, $i=1,\ldots,n$, $a_p \neq a_q$, $p \neq q$.

Định nghĩa

- **Thứ tự từ điển:**

  $a = (a_1\ a_2\ \ a_3\ \ldots a_n)$

  $b = (b_1\ b_2\ b_3\ \ldots\ b_n)$

  thứ tự a < b,

  nếu tồn tại **j** $(1 \leq j \leq n)$:

  $a_1 = b_1, \ldots, a_{j-1} = b_{j-1},\ a_j < b_j$

1.2. Phương pháp sinh

Bước 3:

n= 4

i = 1 2 3 4

a = {1 **3** 4 2}

k = 1 2 3 4

a = { 1 **3** **4** 2}

a = { , **4** , **3**, }

a = { , , **2**, **3** }

Thuật toán

- a = (a_1 a_2...a_n) chưa phải là cuối cùng.
- ✓ B1: Tìm Right -> Left, j đầu tiên:

$$\mathbf{a_j \le a_{j+1}}$$

- ✓ B2: Tìm Right -> Left, k đầu tiên:

$$\mathbf{a_k > a_j.}$$

- ✓ B3: hoán vị $\mathbf{a_j}$ và $\mathbf{a_k}$
- ✓ B4: Lật ngược đoạn a_{j+1} đến a_n

1.2. Phương pháp sinh

Bước 3:

- B1: Tìm Right -> Left, j đầu tiên:

  $$\mathbf{a_j \le a_{j+1}}$$

- B2: Tìm Right -> Left, k đầu tiên:

  $$\mathbf{a_k > a_j.}$$

- B3: hoán vị $\mathbf{a_j}$ và $\mathbf{a_k}$
- B4: Lật ngược đoạn a_{j+1} đến a_n

```
int j = n-1;
while(a[j]>a[j+1])j--;

int k = n;
while(a[j]>=a[k])k--;

int temp = a[j];
a[j] = a[k];  a[k] = temp;

int l = j+1;int r = n;
while(l<r) {
  int temp = a[l];
   a[l]= a[r]; [r]= temp;
    l++;       r--;
}
```

1.2. Phương pháp sinh

```c
#include <stdio.h>
int n,stop,count=1;
void Init(int a[]){
   int i;
   for(i=1;i<=n;i++) a[i]=i;
   }
void Out(int a[]){
   printf("\n%d: ",count);
   count++;
   int i;
   for(i=1;i<=n;i++) printf("%d",a[i]);
   }
int isLastconfigure(int a[]){
   int i;
   for(i=1;i<=n-1;i++) if(a[i]<a[i+1]) return 0;
   return 1;
   }
```

1.2. Phương pháp sinh

```
void nextconfigure(int a[]){
   int j = n-1;
   while(j>=1 && a[j]>a[j+1])j--;
   int k = n;
   while(a[j]>a[k])k--;
   int temp = a[j];
   a[j] = a[k];
   a[k] = temp;
   int l = j+1; int r = n;
   while(l<r) {
      int temp = a[l];
      a[l]= a[r];
      a[r]= temp;
      l++; r--;
      }
   }
```

1.2. Phương pháp sinh

```
void genarate(int a[]){
  Init(a); Out(a);
  stop = isLastconfigure(a);
  while(stop==0){
    nextconfigure(a); Out(a);
    stop = isLastconfigure(a);
    }
  }
main(){
  printf("pt hoan vi n:= ");scanf("%d",&n);
  int a[n];
  genarate(a);
  getch();
  }
```

1.2. Phương pháp sinh

Ví dụ 2

Liệt kê các tập hoán vị của tập n phần tử

Kết quả	**Chương trình**
Result	**Source Code**

Bài toán liệt kê

Nội dung

- ✓ 1.1. Giới thiệu
- ✓ 1.2. Phương pháp sinh
- ✓ 1.3. Phương pháp quay lui

Phương pháp quay lui

Nội dung

- ✓ Mục đích
- ✓ Khái niệm
- ✓ Phương pháp
- ✓ Mã giả
- ✓ Minh họa

1.3. Phương pháp quay lui

Mục đích

- ✓ Để giải bài toán liệt kê hoặc tối ưu tổ hợp
- ✓ Đã giải:
  - ✓ Bài toán mã đi tuần
  - ✓ Bài toán xếp hậu
  - ✓ Bài toán mê cung
  - ✓ Bài toán người giao hàng

Khái niệm

- ✓ Backtracking
- ✓ Поиск с возвратом
- ✓ Quay lui
  - ✓ **Backtracking**
  - ✓ 1950,
  - ✓ **D.H.Lehmer**

1.3. Phương pháp quay lui

Khái niệm

- ✓ Backtracking– đi tìm lời giải cho bài toán mà nghiệm của nó là một cấu hình hoặc một tập cấu hình:
  - ✓ Tính chất P: xác định cấu hình
  - ✓ Tính chất Q: xác định tính dừng của bài toán

Khái niệm

- ✓ Một cấu hình hay một nghiệm:

$$s_1 s_2 \ldots s_n$$

$$s_i \in D$$

- ✓ **Bài toán:**

Liệt kê tất cả các cấu hình của S

$$S = s_1 s_2 \ldots s_n$$

- ✓ **Bài toán con:**

Giả sử đã tìm được $s_1 \ldots s_i$, $i < n$

Hãy tìm cấu hình S

1.3. Phương pháp quay lui

Phương pháp

- Giả sử có: $s_1 \ldots s_{\mathbf{i-1}}$

✓ Tìm giá trị cho $\mathbf{s}_{\mathbf{i}}$:

- ✓ Duyệt $\forall$ j $\in$ *D* đề cử cho $\mathbf{s_i}$
- ✓ Mỗi j $\in$ *D* kiểm tra:
  - ✓ Nếu j chấp nhận (∈ P), s_i = j
    - ✓ Nếu i = n (∈ Q) , được 1 cấu hình,
    - ✓ Nếu i< *n*, tìm s_{i+1}.
  - ✓ Nếu $\forall$ j không được cấp nhận (∉ P) (ngõ cụt) thì quay lại bước xác định s_{i-1}

Bản chất

- ✓ Một quy trình tìm kiếm theo chiều sâu
- ✓ Ví dụ tìm số có 3 chữ số đầu tiên
- ✓ D = {1,2, 3}
- ✓ P:
  - ✓ (2,1,3) – chấp nhận
  - ✓ (2,2,3) – không chấp nhận

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

1.3. Phương pháp quay lui

Phương pháp

- Để liệt kê tất cả cấu hình $S = s_1\ s_2 \ldots\ s_n$
- ✓ Giả sử có: $s_1\ s_2 \ldots\ s_{i-1}$
  - ✓ Ở bước thử tìm giá trị cho thành phần s_i:
    - Duyệt tất cả các khả năng j đề cử cho s_i .
    - Ứng với mỗi khả năng j kiểm tra:
      - Nếu chấp nhận **j** thì $\mathbf{s_i = j}$,
        - » nếu $\mathbf{i = n}$ được 1 cấu hình
        - » ngược lại, thì tiến hành xác định thành phần $\mathbf{s_{i+1}}$.
      - Nếu thử tất cả khả năng mà không được chấp nhận thì quay lại bước xác định thành phần $\mathbf{s_{i-1}}$

1.3. Phương pháp quay lui

Phương pháp

- Giả sử có: $s_1 \ldots s_{\mathbf{i-1}}$

✓Tìm giá trị cho $\mathbf{s}_{\mathbf{i}}$:

✓ Duyệt $\forall$ j $\in$ *D* đề cử cho $\mathbf{s_i}$

✓ Mỗi *i* j $\in$ *D* kiểm tra:

  ✓ Nếu j chấp nhận (∈ P), s_i = j

    ✓ Nếu i = n (∈ Q) , được 1 cấu hình,

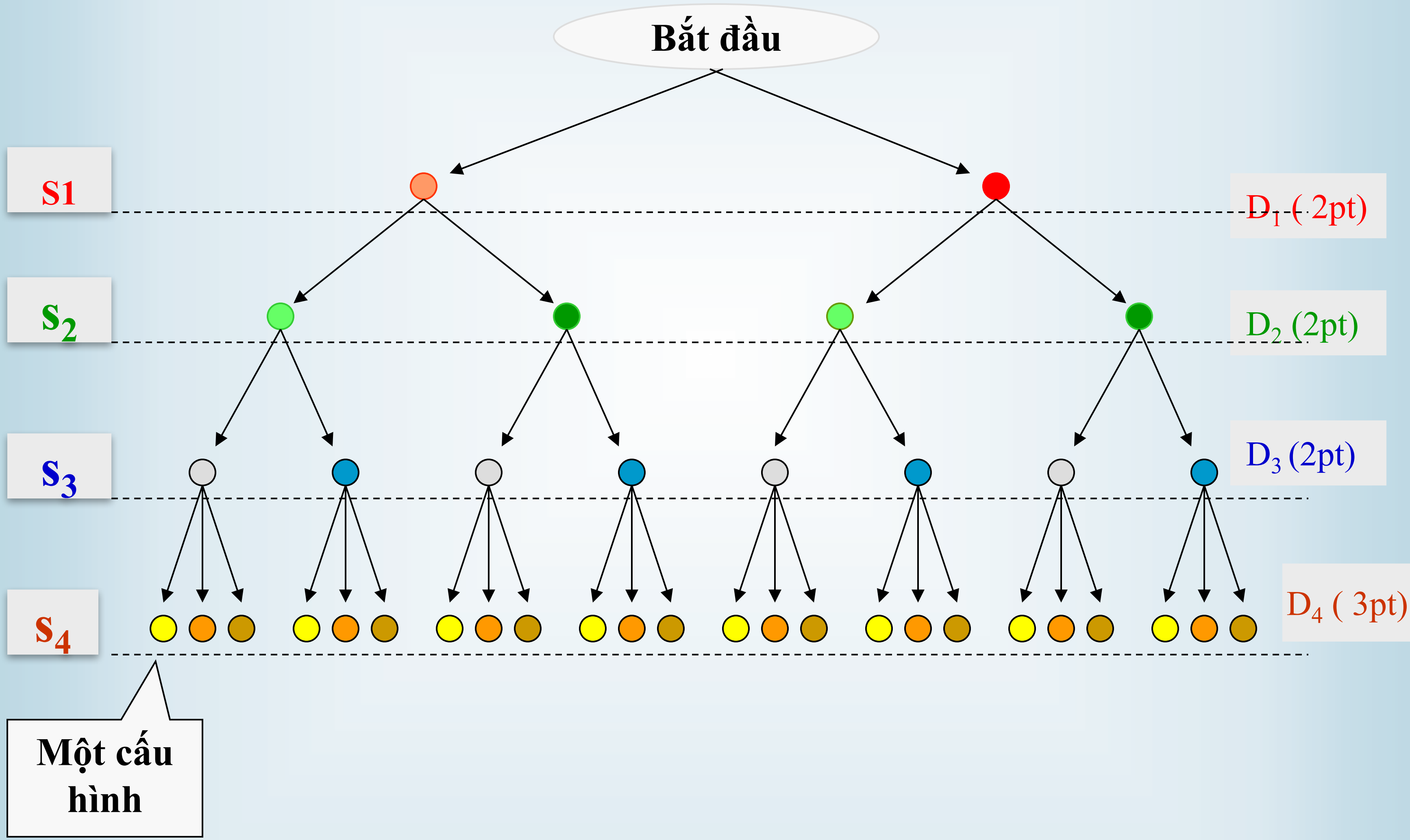
    ✓ Nếu i< *n*, tìm s_i +1.

  ✓ Nếu ∀ j không được cấp nhận (∉ P) (ngõ cụt) thì quay lại bước xác định s_{i-1}

Mã giả

```
void try (i, n){
    ∀j ∈ Di{
      if (< chấp nhận j>){
        si = j;
       if (i==n)
         <Ghi cấu hình S>
       else try (i+1, n);
       }
    }
}
```

1.3. phương pháp quay lui



1.2. Phương pháp quay lui

Ví dụ

- ✓ Liệt kê dãy nhị phân có độ dài n.
- ✓ Liệt kê hoán vị tập n phần tử.
- ✓ Bài toán Xếp Hậu.

1.3. Phương pháp quay lui

Ví dụ 1

- ✓ Liệt kê xâu nhị phân độ dài n
- ✓ Biểu diễn dãy nhị phân:

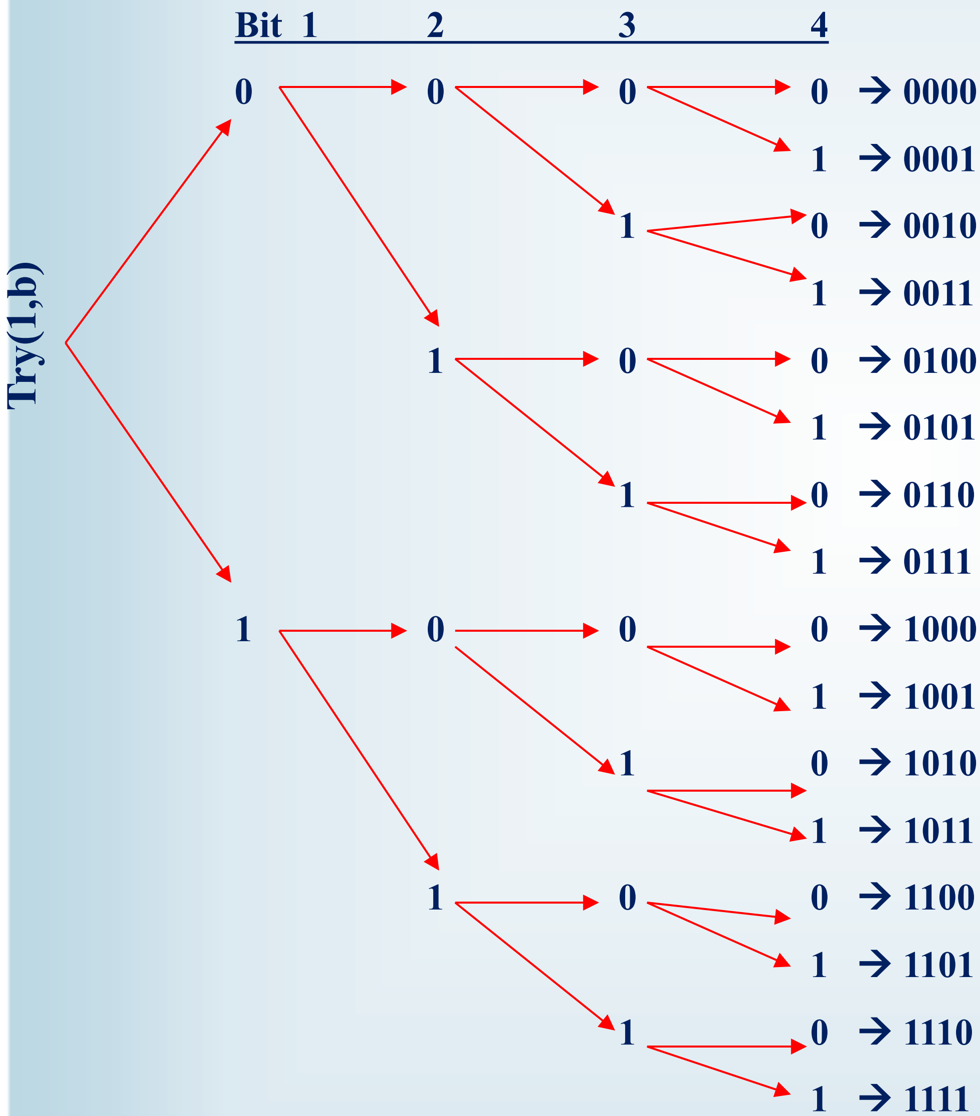
  $b = (b_1\ b_2 \dots b_n)$

  $b_i \in \{0,1\}$
- ✓ Try(i,…) - xác định b_i
- ✓ $D = \{0,1\}$
- ✓ P
- ✓ $i = n$ - được 1 cấu hình $\in Q$

Mã giả

```
void Try(int i, char b[]){
  char j;
   for(j='0'; j<='1';j++){
   //P – bỏ qua kiêm tra
    b[i]=j;
    if(i==n) Out(b);
    else Try(i+1,b);
  }
}
```

1.3. Phương pháp quay lui



Mã giả

```
void Try(int i, char b[]){
  char j;
   for(j='0'; j<='1';j++){
   //P – bỏ qua kiêm tra
    b[i]=j;
    if(i==n) Out(b);
    else Try(i+1,b);
  }
}
```

1.3. Phương pháp quay lui

```c
#include <stdio.h>
int n,count=1;
void Out(char b[]){
  printf("\n%d:\t",count);
  int i; count++;
  for(i=1;i<=n;i++)putchar(b[i]);
 }
void Try(int i, char b[]){
  char j;
  for(j='0';j<='1';j++){
    b[i]=j;
    if(i==n) Out(b);
    else Try(i+1,b);
  }}
main(){
  printf("nhap n = ");scanf("%d",&n);
  char b[n]; printf("xau nhi phan %d bit",n);
  Try(1,b);   getch();
  }
```

1.2. Phương pháp quay lui

Ví dụ 1

Liệt kê xâu nhị phân độ dài n

Kết quả	**Chương trình**
Result	**Source Code**

1.3. Phương pháp quay lui

Ví dụ 2

- Liệt kê các hoán vị của tập
  $X = \{1,2,\ldots,n\}$.

✓ Biểu diễn hoán vị dạng

$s_1\ s_2\ \ldots\ s_n$

$s_i \in X,\ s_p \neq s_q,\ p \neq q.$

- Try(i,…) - xác định s_i

✓ $D = \{1,\ldots,n\}$

✓ Chấp nhận $j \in D$ nếu j chưa được chọn.

Ví dụ 2

✓ Sử dụng mảng b

$b = \{ b[1], b[2],\ldots, b[n]\}$

$$b[j] = \begin{cases} 1, & j \text{ chưa chọn} \\ 0, & j \text{ đã được chọn} \end{cases}$$

✓ Khởi tạo $b[j] = 1, \forall j \in D$

✓ $i = n$ - được 1 cấu hình $\in Q$

1.3. Phương pháp quay lui

Ví dụ 2

- Liệt kê các hoán vị của tập

  $X = \{1,2,\ldots,n\}$.

✓ Biểu diễn hoán vị dạng

$$s_1\ s_2\ \ldots\ s_n$$

$$s_i \in X\ ,\ s_p \neq s_q\ ,\ p \neq q.$$

✓ Try(i,…) - xác định s_i

Mã giả

```
void Try(int i,int b[],int s[]){
  int j;
  for(j=1;j<=n;j++){
    if(b[j]==1){
      s[i]=j;
      b[j]=0;
      if(i==n) Out(s);
      else Try(i+1,b,s);
      b[j]=1;
    } }
}
```

1.3. Phương pháp quay lui

```c
#include <stdio.h>
int n,count=1;
void Init(int b[]){
   int j;
   for(j=1;j<=n;j++)b[j]=1;
}
void Out(int s[]){
   printf("\n %d:\t",count);
   int i;
   for(i=1;i<=n;i++) printf("%d",s[i]);
   count++;
}
```

1.3. Phương pháp quay lui

```
void Try(int i,int b[],int s[]){
   int j;
   for(j=1;j<=n;j++){
      if(b[j]==1){
         s[i]=j;
         b[j]=0;
         if(i==n) Out(s);
         else Try(i+1,b,s);
         b[j]=1;
      } }
}
main(){
   printf("pt hoan vi n:= ");scanf("%d",&n);
   int b[n],s[n];
   Init(b); Try(1,b,s);
   getch();
   }
```

1.2. Phương pháp sinh

Ví dụ 2

Liệt kê hoán vị của tập n phần tử

Kết quả

Result

Chương trình

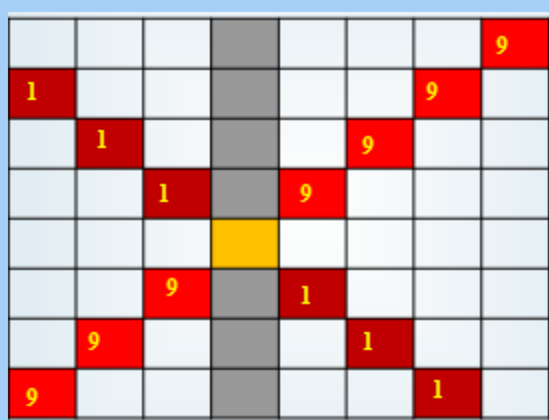
Source Code

1.3. Phương pháp quay lui

Ví dụ 3

- Liệt kê tất cả các cách xếp 8 quân Hậu trên bàn cờ 8x8 sao cho chúng không ăn được nhau.

✓



Ví dụ 3

✓ Đánh số cột từ 1.. 8

✓ Đánh số hàng từ 1.. 8

✓ Biểu diễn cách xếp:

$$x_1\ x_2\ \dots\ x_8$$

✓ x_i được xếp ở hàng i

✓ D = {1,…,8}

 ✓ **x_i= j :** Hậu thứ i được xếp vào cột j

✓ **j** được chấp nhận nếu ô (i,j) được tự do

1.3. Phương pháp quay lui

Ví dụ 3

- ô (i,j) được tự do
  - **Quản lý cột**
  - Quản lý đường chéo thuận
  - Quản lý đường chéo nghịch

Ví dụ 3

- Mảng a – quản lý cột
  a = { a[1], …, a[8]}

$$a[j] = \begin{cases} 1, & \text{cột } j \text{ tự do} \\ 0, & \text{cột } j \text{ đã chiếu} \end{cases}$$

1.3. Phương pháp quay lui

Ví dụ 3:

- ✓ **Quản lý đường chéo thuận**

Ví dụ 3

- ✓ Mảng b – quản lý đường chéo thuận

$$b = \{\, b[2], \ldots, b[16] \}$$

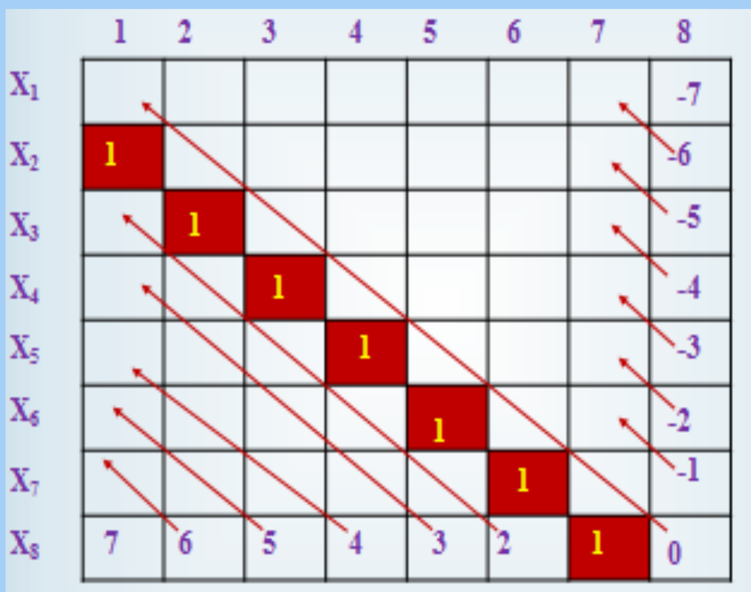
- ✓ b[k]=

$$\begin{cases} 1\,, & \text{đc } chuận\ k \quad tự\ do \\ 0\,, & \text{đc } thuận\ k \ \text{ đã } chiếu \end{cases}$$

1.3. Phương pháp quay lui

Ví dụ 3:

- ✓ **Quản lý đường chéo nghịch**



Ví dụ 3

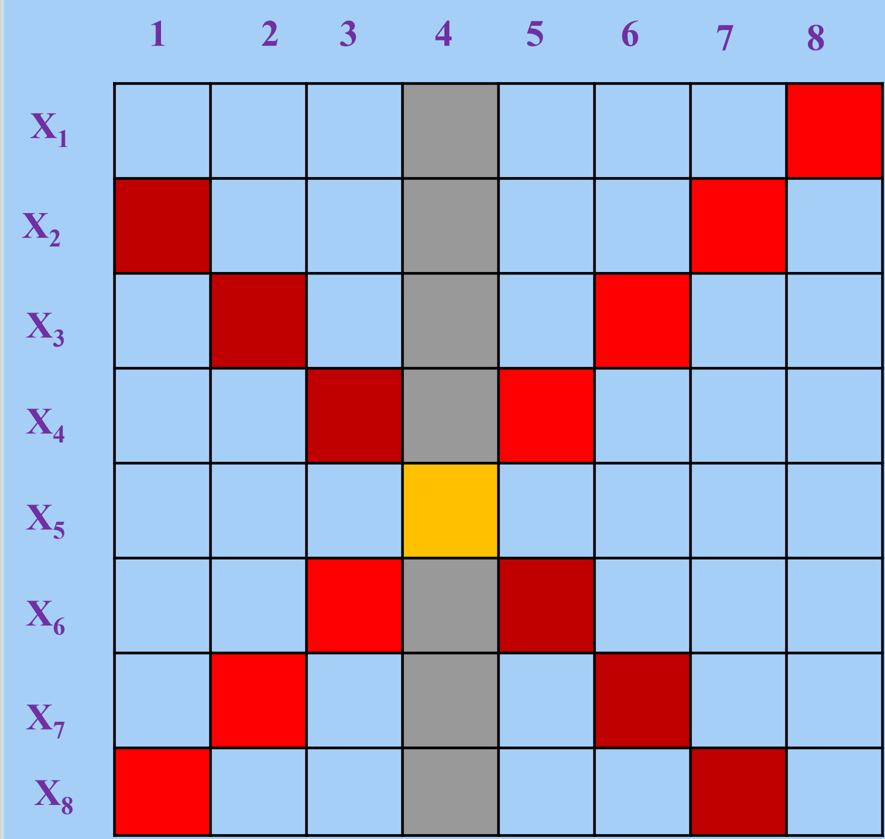
- ✓ Mảng c – quản lý đường chéo nghịch
  c = { c[-7], …,b[7]}

- ✓ c[k]=

$$\begin{cases} 1, & đc\ nghịch\ k\ \ tự\ do \\ 0, & đc\ nghịch\ k\ \ đã\ chiếu \end{cases}$$

1.3. Phương pháp quay lui

Ví dụ 3:



Ví dụ 3

- ✓ Khởi tạo:
  $\forall\, i, k$
  $a_j = 1 \,\&\&\, b_k = 1 \,\&\&\, c_k = 1$
- ✓ j được chấp nhận khi
  $a_j == 1 \,\&\&\, b_{i+j} == 1$
  $\&\&\, c_{i-j} == 1$
- ✓ Try (i,..) – tìm cột đặt con hậu ở hang i

1.3. Phương pháp quay lui

```c
#include<stdio.h>
int a[100],b[100],c[100],x[100];
int count;
int init(int n){
   int i;
   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;
}
int kq(int n){
   int i;
   for(i=1;i<=n;i++)printf("%3d", x[i]);
   printf("\n");
}
```

1.3. Phương pháp quay lui

```
int try(int i,int n){
   int j;
   for(j=1;j<=n;j++){
      if(a[j]&&b[i+j]&&c[i-j]){
         x[i]=j;
         a[j]=0;
         b[i+j]=0;
         c[i-j]=0;
         if(i==n){ kq(i);count++;}
           else try(i+1,n);
         a[j]=1;
         b[i+j]=1;
         c[i-j]=1;
   } } }
```

1.3. Phương pháp quay lui

```
int main(){
    int n;
    printf(" n=  ");
    scanf("%d",&n);
    init(n);
    try(1,n);
    printf("%d",count);
    getch();
}
```

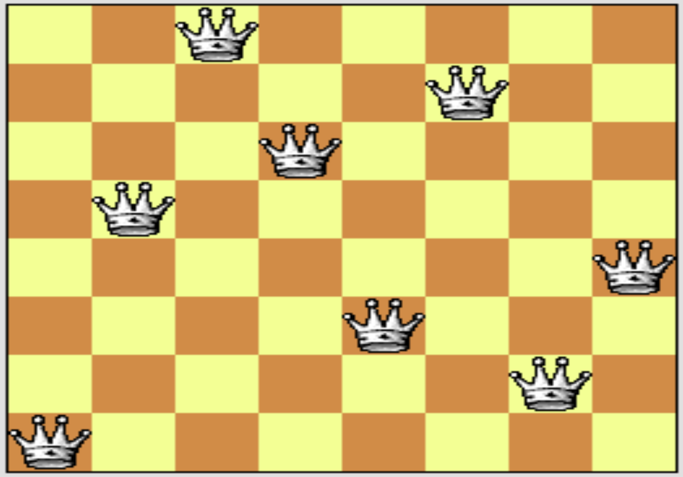
1.2. Phương pháp quay lui

Ví dụ 3

Liệt kê cách xếp hậu

Kết quả

Result



Chương trình

Source Code

x = (3 6 4 2 8 5 7 1)

Phương pháp quay lui

- Liệt kê tổ hợp chập k của n phần tử
- K = 3, n = 4
  - 1 2 3
  - 1 2 4
  - 1 3 4
  - 2 3 4
- j = s[i-1]+1…n-k+I
- s[0] = 0

- Chương trình

```
void Try(int i, int n, int k, int s[]){
    for(int j = 1;j<=n;j++){
    if(j >= s[i-1]+1 && j<=n-k+i){
        s[i] = j;
        if(i==k) Out(n,k,s);
        else Try(i+1,n,k,s);
    }
    }
}
void TH(int n, int k, int s[]){
    if(k>=0&&k<=n)s[0] = 0;
    Try(1,n,k,s);
}
```

Phương pháp quay lui

- Liệt kê chỉnh hợp chập k của n phần tử
- K = 2, n = 3
  - 1 2
  - 1 3
  - 2 1
  - 2 3
  - 3 1
  - 3 2
- c[1] = 1, c[2] = 1, c[3] = 1

- Chương trình

```
void Try(int i, int n, int k, int s[], int c[]){
    for(int j = 1;j<=n;j++){
    if(c[j] == 1){
        s[i] = j;
        c[j] = 0;
        if(i==k) Out(n,k,s);
        else Try(i+1,n,k,s,c);
        c[j] = 1;
    }
    }
}
void TH(int n, int k, int s[], int c[]){
    Test(c,n);
    if(k>=0&&k<=n)s[0] = 0;
    Try(1,n,k,s,c);
}
```

Tóm lại

- Kỹ thuật sinh:
  - (1) Xác định trạng thái đầu của bài toán.
  - (2) Xác định trạng thái kết thúc.
  - (3) Xác định một thứ tự cho các trạng thái.
  - (4) Tìm giải thuật đi từ trạng thái này sang trạng thái khác.
- Kỹ thuật quay lui:
  - (1) Tại 1 thời điểm, chỉ xét thành phần thứ i của cấu hình
  - (2) Với mọi trị j trong miền trị của thành phần này
    - 2.1- Nếu chọn được 1 trị hợp lệ thì
      - Gán $x_i = j$
      - Xử lý cấu hình ở thành phần thứ i+1

Nếu i=0 thì bài toán không có lời giải.

Bài Tập

1. Liệt kê nghiệm nguyên của pt $x_1 + x_2 + x_3 = 15$ với $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$.
2. Liệt kê số chuỗi có *độ dài 3 ký tự xyz* với

   $x \in \{ a,b,c\}$, $y \in \{ d,e\}$, $z \in \{ m,n,t\}$
3. Viết lại các bài mẫu về giải thuật quay lui nhưng ghi kết qủa lên file.

Bài Tập

Cấu hình ban đầu: trị đầu tiên của mỗi miền trị

Cách sinh: Lấy trị kế tiếp của mỗi miền trị theo cơ chế vòng tròn

Cấu hình cuối: trị cuối cùng của mỗi miền trị

x	y	z
a	d	m
a	d	n
a	d	t
a	e	m
a	e	n
a	e	t
b	d	m
b	d	n
b	d	t
b	e	m
b	e	n
b	e	t
c	d	m
c	d	n
c	d	t
c	e	m
c	e	n
c	e	t

Dùng thứ tự từ điển để so sánh:

adm < adn

Kết quả

Mã nguồn

- WHAT NEXT?