



Báo cáo Toán rời rạc Trần Công Định

Trường Đại học Bách Khoa - Đại học Đà Nẵng (Trường Đại học Bách Khoa - Đại học Đà Nẵng)



Scan to open on Studocu

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN

Tel. (+84.0236) 3736949, Fax. (84-511) 3842771
Website: <http://dut.udn.vn/khoacntt>, E-mail: cntt@dut.udn.vn



BÁO CÁO MÔN HỌC
TOÁN RỜI RẠC
BÀI TOÁN LIỆT KÊ

HỌ TÊN SINH VIÊN	LỚP
Trần Công Định	BSKT Khóa 48

CBHD : TS. Nguyễn Văn Hiệu

Đà Nẵng, 06/2024

MỤC LỤC

BÀI TOÁN LIỆT KÊ

1. GIỚI THIỆU BÀI TOÁN LIỆT KÊ	1
2. PHƯƠNG PHÁP THỰC HIỆN	1
2.1. Phương pháp sinh	1
2.1.1. Điều kiện	1
2.1.2. Ứng dụng phương pháp sinh.....	2
2.2. Phương pháp quay lùi (Backtracking Algorithm)	14
2.2.1. Khái niệm	14
2.2.2. Ứng dụng phương pháp quay lùi	16

BÀI TOÁN LIỆT KÊ

1. GIỚI THIỆU BÀI TOÁN LIỆT KÊ

- Bài toán đưa ra danh sách tất cả cấu hình tổ hợp thoả mãn một số tính chất cho trước được gọi là *bài toán liệt kê tổ hợp*.
- Do số lượng cấu hình tổ hợp cần liệt kê thường là rất lớn ngay cả khi kích thước cấu hình chưa lớn:
 - ✓ Số hoán vị của n phần tử là $n!$
 - ✓ Số tập con m phần tử của n phần tử là $n!/(m!(n-m)!)$
 - ✓ Do đó cần có quan niệm thế nào là giải bài toán liệt kê tổ hợp
- Bài toán liệt kê tổ hợp là giải được nếu như ta có thể xác định một thuật toán để theo đó có thể lần lượt xây dựng được tất cả các cấu hình cần quan tâm.
- Một thuật toán liệt kê phải đảm bảo 2 yêu cầu cơ bản:
 - ✓ Không được lặp lại một cấu hình.
 - ✓ Không được bỏ sót một cấu hình.
- Phương pháp sử dụng giải bài toán liệt kê:
 - ✓ Phương pháp sinh
 - ✓ Phương pháp quay lui

2. PHƯƠNG PHÁP THỰC HIỆN

2.1. Phương pháp sinh

2.1.1. Điều kiện

- 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 một 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 được thuật toán từ cấu hình chưa phải là cuối cùng đang có, đưa ra cấu hình kế tiếp nó. Thuật toán nói đến trong điều kiện 2) được gọi là Thuật toán Sinh kế tiếp

2.1.2. Xây dựng được thuật toán:

Bản chất	Chú thích
<pre>Generating_method(...) {<Cấu hình ban đầu>;stop = islastconfigure(...);while (stop==0) { <Cấu hình hiện thời ><Sinh_kế_tiếp>; } }</pre>	<ul style="list-style-type: none"> - 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

- Giải thích:

✓ *Sinh_kế_tiếp* là thủ tục thực hiện thuật toán sinh kế tiếp đã xây dựng trong điều kiện 2). Thủ tục này sẽ xây dựng cấu hình kế tiếp của cấu hình đang có trong thứ tự đã xác định.

✓ *Chú ý:* Do tập các cấu hình tổ hợp cần liệt kê là hữu hạn nên luôn có thể xác định được thứ tự trên nó. Tuy nhiên, thứ tự cần xác định sao cho có thể xây dựng được thuật toán Sinh kế tiếp.

2.1.2. Ứng dụng phương pháp sinh

a. Liệt kê dãy nhị phân có độ dài n

✓ Bước 1: Xác định thứ tự

- Dãy nhị phân được biểu diễn: $b = (b_1 b_2 \dots b_n)$ thỏa mãn $b_i \in \{0,1\}$
- Định nghĩa thứ tự từ điển: $b = (b_1 b_2 \dots b_n)$ và $*b = (*b_1 *b_2 \dots *b_n)$ **thứ tự** $b < *b$, nếu $q(b) < q(*b)$

✓ Bước 2: Cấu hình đầu và cuối

Khi $n = 4$ phần tử, có 16 dãy nhị phân được liệt kê như sau:

b	q(b)	b	q(b)
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10

b	q(b)	b	q(b)
0011	3	1011	11
0011	4	1100	12
0100	5	1101	13
0110	6	1110	14
0111	7	1111	15

✓ Bước 3: Xác định thuật toán

0000 => 0001 0001 => 0010 0011 => 0100 0111 => 1000	✓ Tìm i từ bên phải thỏa mãn: $b_i = 0$. ✓ Gán lại $b_i = 1$ và $b_j = 0$ với mọi $j > i$
--	---

✓ Chương trình thực hiện bằng ngôn ngữ C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication1
{
    class Program
    {
        static void init(int[] s, int k)
        {
            for (int i=0; i<k; i++)
            {
                s[i]=0;
            }
        }
    }
}

```

```

static bool isLast(int[] s, int n)
{

    for (int i = n-1; i >= 0; i--)
    {
        if (s[i] != 1 ) return false;
    }
    return true;
}

static void gen_nextString(int [] s, int n)
{
    int i = n-1;
    //Tìm vị trí số 0 từ phải sang trái
    while (s[i] == 1 && i>=0) { i--; }
    //gán số vị trí số 0 thành 1
    s[i] = 1;
    //Thay thế số các số còn lại sau số 1 thành 0
    for (int j=i+1;j<n;j++)
    {
        s[j] = 0;
    }
}

static void method_String(int[] s, int n)
{
    int count = 0;
    init(s, n);
    outString(s, n, count);

    bool stop = isLast(s,n);//stop=true/false

    while(stop==false)
    {
        count++;
    }
}

```

```

        gen_nextString(s, n);
        outString(s, n, count);
        stop = isLast(s, n);
    }
}

static void outString(int[] s, int n, int count)
{
    Console.WriteLine();

    for (int i = 0; i < n; i++)
    {
        Console.Write(s[i]);
    }

    Console.Write("      :"+ count );
}

static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.UTF8;
    Console.InputEncoding = Encoding.Unicode;

    Console.Write("Nhập giá trị của chiều dài của dãy nhị phân: ");
    int n = int.Parse(Console.ReadLine());
    Console.Write("Dãy nhị phân có chiều dài " + n + " là: ");
    int[] s = new int[n];

    method_String(s, n);

    Console.ReadKey();
}
}
}

```

✓ **Kết quả thực hiện**

file:///E:/Program C sharp/Learning/lietke1/ConsoleApplication1/bin/Debug/ConsoleApplication1.EXE

Nhập giá trị của chiều dài của dãy nhị phân: 4

Dãy nhị phân có chiều dài 4 là:

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

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

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

$$a = (a_1, a_2, \dots, a_k) \text{ thỏa mãn } 1 \leq a_1 < a_2 < \dots < a_k \leq n.$$

✓ Bước 1: Xác định thứ tự

Định nghĩa thứ tự từ điển:

$a = (a_1, a_2, \dots, a_k)$ và $b = (b_1, b_2, \dots, b_k)$ trong thứ tự từ điển $a < b$, nếu tồn tại j ($1 \leq j \leq k$) sao cho:

$$a_1 = b_1, a_2 = b_2, \dots, a_{j-1} = b_{j-1}, a_j < b_j.$$

✓ Bước 2:

- Các tập con 3 phần tử của tập 5 phần tử **{1,2,3,4,5}**
- $C_5^3 = 10$
- Chuỗi liệt kê như sau:


{ 1,2,3 } Cấu hình đầu

{ 1,2,4 }

{ 1,2,5 }

{ 1,3,4 }
 { 1,3,5 }
 { 1,4,5 } **Cách sinh**
 { 2,3,4 }
 { 2,3,5 }
 { 2,4,5 }
 { 3,4,5 } **Cấu hình cuối**

✓ **Bước 3: Xác định thuật toán**

n = 5, k=3	Giả sử $a = (a_1 \dots a_k)$ không là cuối cùng
i = 1 2 3	
a = {1, 4, 5}	
n-k+i= 3 4 5	B1: Tìm vị trí i đầu tiên từ bên phải của dãy: $a_i \neq n-k+i$
	
{2, , }	B2: Thay a_i bởi $a_i + 1$
{2, 3, 4}	B3: Thay a_j bởi $a_i - i + j$, với $j = i+1, \dots, k$

✓ **Chương trình thực hiện bằng ngôn ngữ C#**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace lietke2
{
    class Program
    {
        static void init(int[] a, int k)
        {
            for (int i = 0; i < k; i++)
            {

```

```

        a[i] = i+1;
    }
}
static bool isLast(int[] a, int n, int k)
{
    for (int i = k-1; i >=0; i--)
    {
        if (a[i] != n-k+i+1) return false;
    }
    return true;
}
static void gen_nextString(int[] a, int n, int k)
{
    int i = k - 1;
    //Tìm vị trí i đầu tiên a[i] đầu từ phải sang trái khác n-k+i
    while (a[i] == n-k+i+1 && i >= 0) { i--; }
    //gán số a[i]=a[i]+1
    a[i] = a[i]+1;
    //Thay thế Thay a[j] bởi a[i] - i +j; j =i+1
    for (int j = i + 1; j < k; j++)
    {
        a[j] = a[i]-i+j;
    }
}
static void method_String(int[] s, int n, int k)
{
    int count = 1;

    init(s, k);
    outString(s, k,count);

    bool stop = isLast(s, n, k); //stop=true/false

    while (stop == false)

```

```

        {
            count++;
            gen_nextString(s, n,k);
            outString(s, k, count);
            stop = isLast(s, n, k);
        }
    }
    static void outString(int[] a, int k, int count)
    {
        Console.WriteLine();
        for (int i = 0; i < k; i++)
        {
            Console.Write(a[i]);
        }
        Console.Write(" :"+count);
    }
    static void Main(string[] args)
    {
        Console.OutputEncoding = Encoding.UTF8;
        Console.InputEncoding = Encoding.Unicode;

        Console.Write("Nhập vào phần tử mẹ n = ");
        int n = int.Parse(Console.ReadLine());
        Console.Write("Nhập vào phần tử con k = ");
        int k = int.Parse(Console.ReadLine());
        Console.Write("Các tập con " + k + " phần tử của tập "+n+" phần tử
là:");

        int[] a = new int[k];
        method_String(a,n, k);
        Console.ReadKey();
    }
}
}

```

✓ **Kết quả thực hiện**

file:///E:/Program C sharp/Learning/lietke2/lietke2/bin/Debug/lietke2.EXE

```
Nhập vào phần tử mẹ n = 5
Nhập vào phần tử con k = 3
Các tập con 3 phần tử của tập 5 phần tử là:
123 :1
124 :2
125 :3
134 :4
135 :5
145 :6
234 :7
235 :8
245 :9
345 :10
```

c. Liệt kê các hoán vị của tập n phần tử

Cho $X = \{1, 2, \dots, n\}$, 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:

$$a = (a_1, a_2, \dots, a_n)$$

Thỏa mãn $a_i \in X, i = 1, 2, \dots, n, a_p \neq a_q, p \neq q$

✓ Bước 1: Xác định thứ tự từ điển

$$a = (a_1, a_2, \dots, a_n)$$

$b = (b_1, b_2, \dots, b_n)$ thứ tự $a < b$, nếu tồn tại j ($1 \leq j \leq n$) sao cho:

$$a_1 = b_1, a_2 = b_2, \dots, a_{j-1} = b_{j-1}, a_j < b_j$$

✓ Bước 2:

Liệt kê các hoán vị của 4 phần tử như sau:

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

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

✓ **Bước 3: Xây dựng thuật toán sinh**

n = 4	Giả sử $a = (a_1, a_2, \dots, a_n)$ không là cuối cùng
i = 1 2 3 4	B1: Tìm từ Phải sang trái, j đầu tiên thỏa mãn: $a_j \leq a_{j+1}$
a = {1, 3, 4, 2}	B1: Tìm từ Phải sang trái, j đầu tiên thỏa mãn: $a_j \leq a_{j+1}$
k = 1 2 3 4	B2: Tìm từ Phải sang trái, k đầu tiên thỏa mãn: $a_k > a_j$
a = {1, 3, 4, 2}	B3: Hoán vị a_k và a_j
a = {, 4, 3, }	
a = {, , 2, 3 }	B4: Lật ngược đoạn a_{j+1} và a_n

✓ **Chương trình thực hiện bằng ngôn ngữ C#**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace hoanvi
{
    class Program
    {
        static void init(int[] a, int n)
        {
            for (int i = 0; i < n; i++)
            {
                a[i] = i + 1;
            }
        }
    }
}
```

```

}
static bool isLast(int[] a, int n)
{
    for (int i = 0; i < n-1; i++)
    {
        if (a[i] < a[i+1]) return false;
    }
    return true;
}
static void gen_nextString(int[] a, int n)
{
    //Tìm vị trí đầu tiên từ bên phải mà a[j]<=a[j+1]
    int j = n - 2;
    while (a[j] >a[j+1] && j >= 0) { j--; }
    //Tìm vị trí đầu từ từ bên phải mà a[k]>a[j]
    int k = n-1;
    while (a[k] < a[j] && k >= 0) k--;
    //hoán vị a[j] và a[k]
    int temp = a[j];
    a[j] = a[k];
    a[k] = temp;
    //Lật ngược đoạn a[j+1] đến a[n]
    int l = j + 1; int r = n - 1;
    while (l<r)
    {
        int temp1 = a[l];
        a[l] = a[r];
        a[r] = temp1;
        l++; r--;
    }
}
static void method_String(int[] a, int n)
{
    int count = 1;

```

```

init(a, n);
outString(a, n, count);
bool stop = isLast(a, n); //stop=true/false
while (stop == false)
{
    count++;
    gen_nextString(a, n);
    outString(a, n, count);
    stop = isLast(a, n);
}
}

static void outString(int[] a, int n, int count)
{
    Console.WriteLine();
    for (int i = 0; i < n; i++)
    {
        Console.Write(a[i]);
    }
    Console.Write(" :" + count);
}

static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.UTF8;
    Console.InputEncoding = Encoding.Unicode;
    Console.Write("Phần tử hoán vị n = ");
    int n = int.Parse(Console.ReadLine());

    Console.Write("Các hoán vị của tập " + n + " phần tử là:");
    int[] a = new int[n];
    method_String(a, n);
    Console.ReadKey();
}
}

```


}

✓ Kết quả thực hiện

file:///E:/Program C sharp/Learning/lietke3/hoanvi/bin/Debug/hoanvi.EXE

```
Phần tử hoán vị n = 4
Các hoán vị của tập 4 phần tử là:
1234 :1
1243 :2
1324 :3
1342 :4
1423 :5
1432 :6
2134 :7
2143 :8
2314 :9
2341 :10
2413 :11
2431 :12
3124 :13
3142 :14
3214 :15
3241 :16
3412 :17
3421 :18
4123 :19
4132 :20
4213 :21
4231 :22
4312 :23
4321 :24
```

2.2. Phương pháp quay lùi (Backtracking Algorithm)

2.2.1. Khái niệm

- Để giải bài toán liệt kê hoặc tối ưu tổ hợp
- 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
- Một cấu hình hay một nghiệm: $s_1 s_2 \dots s_n$ $s_i \in D$

Bài toán: Liệt kê tất cả các cấu hình của S

$S = s_1 s_2 \dots s_n$

Bài toán con: Giả sử đã tìm được $s_1 s_2 \dots s_i$, $i < n$

Hãy tìm cấu hình S

Giả sử có: $s_1 s_2 \dots s_i$

- Tìm giá trị cho s_i :

+ Duyệt $\forall j \in D$ đề cử cho s_i

+ Mỗi $j \in D$ kiểm tra:

* Nếu j chấp nhận ($\in P$), $s_i = j$

- Nếu $i = n$ ($\in Q$), được 1 cấu hình,

- Nếu $i < n$, tìm s_{i+1}

* Nếu $\forall j$ không được chấp nhận ($\notin P$) (ngõ cụt) thì quay lại bước xác định s_{i-1}

Để liệt kê tất cả cấu hình $S = s_1 s_2 \dots s_n$

- Giả sử có: $s_1 s_2 \dots s_{i-1}$

+ Ở bước thứ 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 $s_i = j$,

» nếu $i = n$ được 1 cấu hình

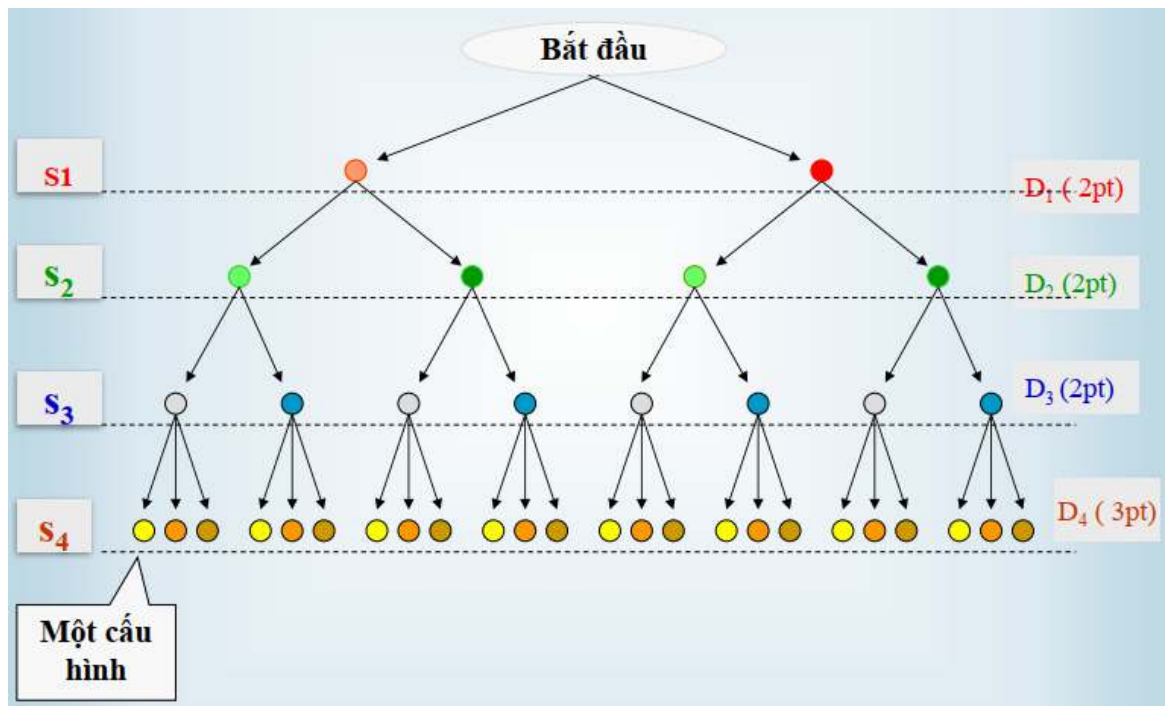
» ngược lại, thì tiến hành xác định thành phần s_{i+1} .

+ Nếu thử tất cả khả năng mà không được chấp nhận thì quay lại bước xác định thành phần s_{i-1}

PHƯƠNG PHÁP	MÃ CODE
Giả sử có: $s_1 s_2 \dots s_i$ - Tìm giá trị cho s_i : + Duyệt $\forall j \in D$ đề cử cho s_i + Mỗi $j \in D$ kiểm tra: * Nếu j chấp nhận ($\in P$), $s_i = j$ - Nếu $i = n$ ($\in Q$), được 1 cấu hình, - Nếu $i < n$, tìm s_{i+1} * Nếu $\forall j$ không được chấp nhận ($\notin P$)	<pre>void try (i, n){ $\forall j \in D\{$ if (< chấp nhận j >) { $s_i = j;$ if ($i == n$) <Ghi cấu hình S> else try ($i+1, n$); } }</pre>

P) (ngõ cụt) thì quay lại bước xác định	}
S_{i-1}	}

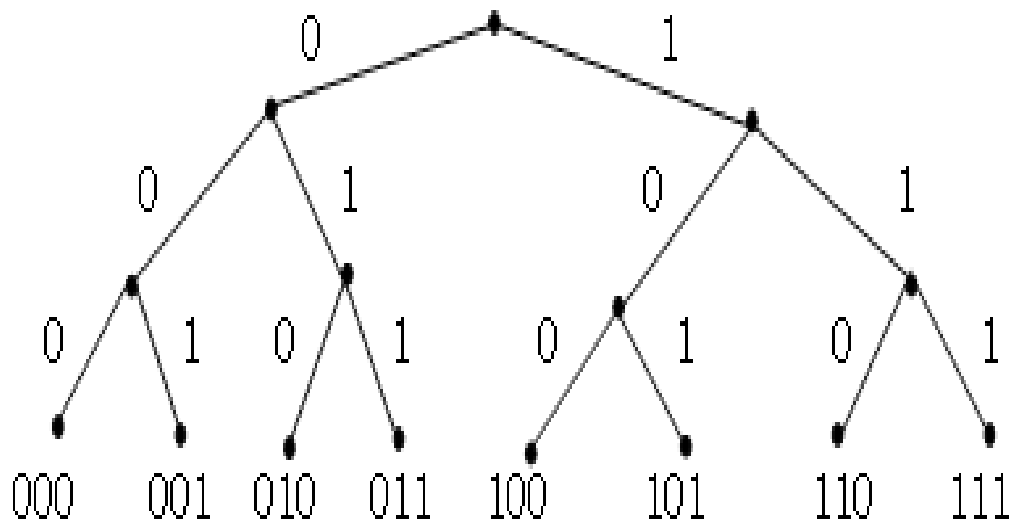
- *Cây liệt kê lời giải theo thuật toán quay lui*



2.2.2. Ứng dụng phương pháp quay lui

a. Liệt kê dãy nhị phân có độ dài n

- Dãy nhị phân được biểu diễn: $b = (b_1 b_2 \dots b_n)$ thỏa mãn $b_i \in \{0,1\}$
- Xác định b_i
- $D = \{0,1\}$
- P
- $i = n - \text{được 1 cấu hình} \in Q$, ngược lại tìm $i+1$
- Cây liệt kê dãy nhị phân độ dài 3



- Chương trình bằng ngôn ngữ C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace quayLuiNhiPhan
{
    class Program
    {
        static void outString(int[] a, int n)
        {
            Console.WriteLine();
            for (int i = 0; i < n; i++)
            {
                Console.Write(a[i]);
            }
        }
    }
}

```

```

static void TryString(int[] a, int i, int n)
{
    for (int j = 0; j <= 1; j++)
    {
        a[i] = j;
        if (i == n - 1) { outString(a, n); }
        else TryString(a, i + 1, n);
    }
}

static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.UTF8;
    Console.InputEncoding = Encoding.Unicode;
    Console.Write("Nhập chiều dài của dãy nhị phân n = ");
    int n = int.Parse(Console.ReadLine());
    Console.Write("Liệt kê chuỗi nhị phân có độ dài " + n + " là:");
    int[] a = new int[n];
    TryString(a, 0, n);
    Console.ReadKey();
}
}
}

```

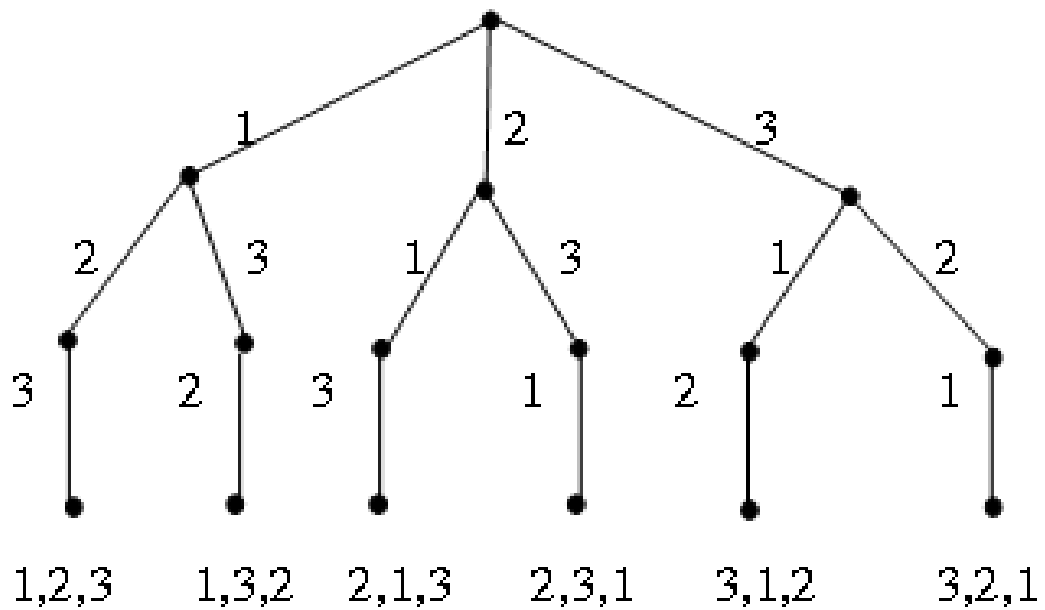
- Kết quả thực hiện:

file:///E:/Program C sharp/Learning/quayLuiNhiPhan/quayLuiNhiPhan/bin/Debug/quayLuiNhiPhan.EXE

```
Nhập chiều dài của dãy nhị phân n = 4
Liệt kê chuỗi nhị phân có độ dài 4 là:
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
```

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

- Biểu diễn hoán vị dạng $s_1 s_2 \dots s_n$ $s_i \in X$, $s_p \neq s_q$, $p \neq q$.
- Xác định S_i - Try(i,...)
- $D = \{1, \dots, n\}$
- Chấp nhận $j \in D$ nếu j chưa được chọn.
- Sử dụng mảng $b = \{b[1], b[2], \dots, b[n]\}$
 - ✓ $b[j] = 0$, j chưa được chọn
 - ✓ $b[j] = 1$, j đã được chọn
- Khởi tại $b[j]=0, \forall j \in D$
- $i = n$ - được 1 cấu hình $\in Q$, ngược lại tìm $i+1$
- Cây liệt kê hoán vị của 1,2,3



- Chương trình bằng ngôn ngữ C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace quayLuiHoanVi
{
    class Program
    {
        public static void outString(int[] a, int n)
        {
            Console.WriteLine();
            for (int i = 0; i < n; i++)
            {
                Console.Write(a[i]);
            }
        }

        //used[] để lưu đã gán giá trị hay chưa
        static void TryString(int[] a, int [] used, int i, int n)
        {

```

```

        for (int j = 0; j < n; j++)
        {
            if (used[j] == 0)
            {
                a[i] = j+1;
                used[j] = 1; // đánh dấu đã gán
                if (i == n - 1) outString(a, n);
                else TryString(a, used, i + 1, n);
                used[j] = 0;
            }
        }
    }

    static void Main(string[] args)
    {
        Console.OutputEncoding = Encoding.UTF8;
        Console.InputEncoding = Encoding.Unicode;
        Console.Write("Nhập phần số phần tử của tập là n = ");
        int n = int.Parse(Console.ReadLine());
        Console.Write("Hoán vị của tập " + n + " là:");
        int[] a = new int[n];
        int[] used = new int[n];
        TryString(a, used, 0, n);
        Console.ReadKey();
    }
}

```

- Kết quả thực hiện

file:///E:/Program C sharp/Learning/quayLuiHoanVi/quayLuiHoanVi/bin/Debug/quayLuiHoanVi.EXE

Nhập phần số phần tử của tập là n = 4

Hoán vị của tập 4 là:

1234

1243

1324

1342

1423

1432

2134

2143

2314

2341

2413

2431

3124

3142

3214

3241

3412

3421

4123

4132

4213

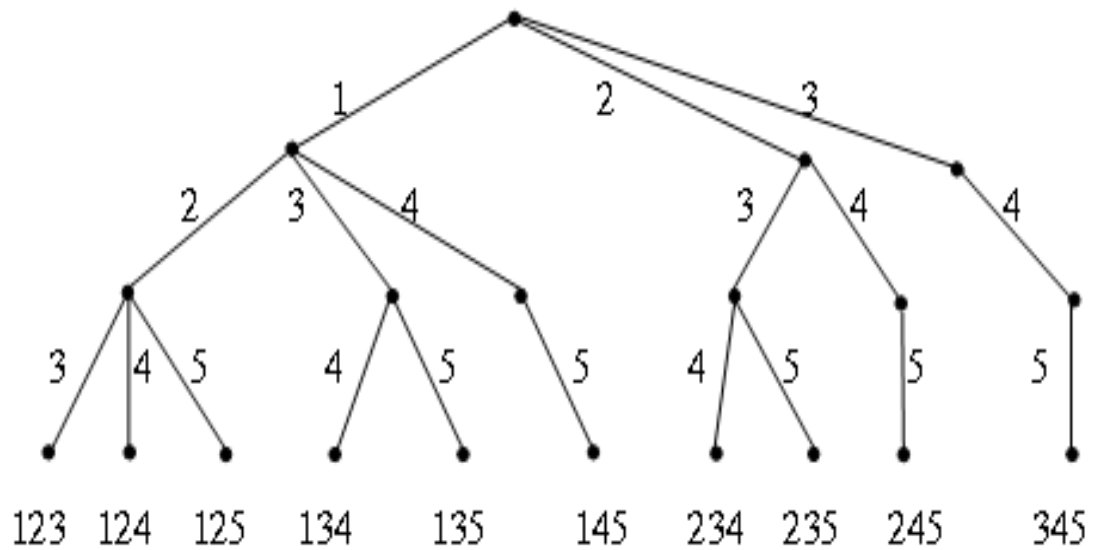
4231

4312

4321

b. Liệt kê tổ hợp chập k của n phần tử

- Biểu diễn hoán vị dạng $s_1 s_2 \dots s_n$ $s_i \in X$, $s_p \neq s_q$, $p \neq q$.
- $K = 3, n = 4$
- 1 2 3
- 1 2 4
- 1 3 4
- 2 3 4
- $j = s[i-1] + 1 \dots n - k + i$
- $s[0] = 0$
- Cây liệt kê tổ hợp chập 3 của 5 phần tử



- **Chương trình bằng ngôn ngữ C#**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace quayLuiToHop
{
    class Program
    {
        static void outString(int[] a, int k)
        {
            Console.WriteLine();
            for (int i = 1; i <= k; i++)
            {
                Console.Write(a[i]);
            }
        }
        static void TryString(int[] a, int i, int n, int k)
        {

```

```


        for (int j = a[i-1] + 1; j <= n - k + i; j++)
        {
            a[i] = j;
            if (i == k) { outString(a, k); }
            else TryString(a, i + 1, n, k);
        }
    }

    static void Main(string[] args)
    {
        Console.OutputEncoding = Encoding.UTF8;
        Console.InputEncoding = Encoding.Unicode;
        Console.Write("Nhập vào phần tử mẹ n = ");
        int n = int.Parse(Console.ReadLine());
        Console.Write("Nhập vào phần tử con k = ");
        int k = int.Parse(Console.ReadLine());
        Console.Write("Các tập con " + k + " phần tử của tập " + n + " phần tử là:");

        int[] a = new int[n];
        TryString(a, 1, n, k);
        Console.ReadKey();
    }
}

```

- **Kết quả thực hiện**

 file:///E:/Program C sharp/Learning/quayLuiToHop/quayLuiToHop/bin/Debug/quayLuiToHop.EXE

```

Nhập vào phần tử mẹ n = 4
Nhập vào phần tử con k = 3
Các tập con 3 phần tử của tập 4 phần tử là:
123
124
134
234_

```