TOÁN RỜI RẠC

CHƯƠNG 5
BÀI TOÁN LIỆT KÊ

Lecturer: PhD. Ngo Huu Phuc
Tel: 0438 326 077
Mob: 098 5696 580
Email: ngohuuphuc76@gmail.com

NỘI DUNG CHƯƠNG 5

5.1. Giới thiệu bài toán.

5.2. Nhắc lại kiến thức đệ quy.

5.3. Sinh hoán vị - Sinh tổ hợp.

5.4. Thuật toán quay lui. Bài toán xếp hậu.

5.5. Bài tập chương 5.

5.1. Giới thiệu bài toán (1/3)

- Cần có giải thuật để lần lượt xây dựng được tất cả các cấu hình đang quan tâm → **BÀI TOÁN LIỆT KÊ**.
- Đối với bài toán liệt kê, cần đảm bảo 2 nguyên tắc:
  - ***Không được lặp lại một cấu hình***.
  - ***Không được bỏ sót một cấu hình***.
- Khó khăn chính của phương pháp này là sự

"***bùng nổ tổ hợp***"!

5.1. Giới thiệu bài toán (2/3)

Ví dụ 5.1:

- Cho tập hợp các số $a_1, a_2,.., a_n$ và số **M**. Hãy tìm tất cả các tập con k phần tử của dãy số $\{a_n\}$ sao cho tổng số các phần tử trong tập con đó đúng bằng M.

Giải ví dụ 5.1.

- Số các tập con k phần tử của tập gồm n phần tử là C(n,k).
- Cần duyệt trong số C(n,k) tập k phần tử để lấy ra những tập có tổng các phần tử đúng bằng M.
- Để thực hiện được bài toán → cần liệt kê các cấu hình.

5.1. Giới thiệu bài toán (3/3)

Ví dụ 5.2:

- Một người bán hàng tại 8 thành phố. Người này có thể bắt đầu hành trình của mình tại một thành phố nào đó nhưng phải qua 7 thành phố kia theo bất kỳ thứ tự nào mà người đó muốn. Hãy chỉ ra lộ trình ngắn nhất mà người đó có thể đi.

Giải ví dụ 5.2.

- Có tất cả 7! = 5040 cách đi của người bán hàng.
- Tuy nhiên trong 5040 cách chúng ta phải duyệt toàn bộ để chỉ ra một hành trình là ngắn nhất.

5.2. Nhắc lại kiến thức đệ quy (1/9)

5.2.1. Định nghĩa bằng đệ quy (1/4)

- Trong thực tế, nhiều đối tượng mà khó có thể định nghĩa nó một cách tường minh, nhưng lại dễ dàng định nghĩa đối tượng qua chính nó.
- Kỹ thuật định nghĩa đối tượng qua chính nó được gọi là kỹ thuật đệ qui (**recursion**).
- Các giải thuật đệ qui thường được xây dựng qua hai bước:
  - **bước phân tích**
  - **bước thay thế ngược lại**

5.2. Nhắc lại kiến thức đệ quy (2/9)

5.2.1. Định nghĩa bằng đệ quy (2/4)

Ví dụ 5.2.3:

- Để tính tổng S(n) = 1 + 2 +...+ n.

Giải quyết bài toán:

- **Bước phân tích:**
  - Để tính toán được S(n), cần tính S(n-1), sau đó tính S(n) = S(n-1) +n.
  - .....
  - Và cuối cùng S(1) chúng ta có ngay kết quả là 1.
- **Bước thay thế ngược lại:**
  - Xuất phát từ S(1) thay thế ngược lại chúng ta xác định S(n):
  - S(1) = 1
  - S(2) = S(1) + 2
  - .....
  - S(n) = S(n - 1) + n

5.2. Nhắc lại kiến thức đệ quy (3/9)

5.2.1. Định nghĩa bằng đệ quy (3/4)

Ví dụ 5.2.4:

- Định nghĩa hàm bằng đệ quy:

$$f(n) = n!$$

Phân tích và thực hiện:

- Ta có f(0) = 1.
- Vì (n+1) ! = 1. 2.3... n(n+1) = n! (n+1), nên ta có:

f(n+1) = (n+1). f(n) với ∀ n nguyên dương.

5.2. Nhắc lại kiến thức đệ quy (4/9)

5.2.1. Định nghĩa bằng đệ quy (4/4)

Ví dụ 5.2.5: Tập hợp định nghĩa bằng đệ quy:

- Định nghĩa đệ quy tập các xâu: Giả sử Σ^* là tập các xâu trên bộ chữ cái Σ. Khi đó Σ^* được định nghĩa bằng đệ quy như sau:
  - $\lambda \in \Sigma^*$, trong đó λ là xâu rỗng
  - $wx \in \Sigma^*$ nếu $w \in \Sigma^*$ và $x \in \Sigma^*$.

5.2. Nhắc lại kiến thức đệ quy (5/9)

5.2.2. Giải thuật đệ quy (1/5)

Khái niệm:

- Một thuật toán được gọi là đệ quy nếu nó giải bài toán bằng cách rút gọn bài toán ban đầu thành bài toán tương tự như vậy sau một số hữu hạn lần thực hiện.
- Trong mỗi lần thực hiện, dữ liệu đầu vào tiệm cận tới tập dữ liệu dừng.

5.2. Nhắc lại kiến thức đệ quy (6/9)

5.2.2. Giải thuật đệ quy (2/5)

Ví dụ 5.2.6: Tính a^n bằng giải thuật đệ quy, với mọi số thực a và số tự nhiên n.

- **Đoạn mã của giải thuật:**

```
double power( float a, int n ){
    if ( n==0 )
        return 1;
    else
        return a *power(a,n-1);
}
```

5.2. Nhắc lại kiến thức đệ quy (7/9)

5.2.2. Giải thuật đệ quy (3/5)

Ví dụ 5.2.7: Thuật toán đệ quy tính ước số chung lớn nhất của hai số nguyên dương a và b.

- **Đoạn mã của giải thuật:**

```
int USCLN( int a, int b ){
    if (a == 0)
        return b;
    else
        return USCLN( b % a, a);
}
```

5.2. Nhắc lại kiến thức đệ quy (8/9)

5.2.2. Giải thuật đệ quy (4/5)

Ví dụ 5.2.8: Thuật toán đệ quy tính n!

- **Đoạn mã của giải thuật:**

```
long factorial( int n){
    if (n ==1)
        return 1;
    else
        return n * factorial(n-1);
}
```

5.2. Nhắc lại kiến thức đệ quy (9/9)

5.2.2. Giải thuật đệ quy (5/5)

Ví dụ 5.2.9: Thuật toán đệ quy tính số fibonacci thứ n.

- **Đoạn mã của giải thuật:**

```
int fibonacci( int n) {
    if (n==0) return 0;
    else if (n==1) return 1;
      return fibonacci(n-1)+fibonacci(n-2);
}
```

5.3. Sinh hoán vị - Sinh tổ hợp (1/20)

5.3.1. Giới thiệu thuật toán sinh (1/4)

Ý tưởng của phương pháp sinh:

a. **Xây dựng một cấu hình tổ hợp ban đầu thoả mãn các điều kiện đã cho.**

b. **Đưa ra cấu hình đã có.**

c. **Từ các thông tin của cấu hình đã có, xây dựng cấu hình mới cũng thoả mãn các điều kiện đã cho:**

- nếu sinh được cấu hình mới ta tiếp tục lặp lại bước b,
- nếu không sinh được thì dừng lại.

5.3. Sinh hoán vị - Sinh tổ hợp (2/20)

5.3.1. Giới thiệu thuật toán sinh (2/4)

Giả mã của phương pháp sinh:

```
void Generate(void) {
    <Xây dựng cấu hình ban đầu>;
    stop = false;
    while (not stop) {
        <Đưa ra cấu hình đang có>;
        Sinh_Kế_Tiếp;
    }
}
```

Trong đó, **Sinh_Kế_Tiếp** thủ tục sinh cấu hình kế tiếp từ cấu hình ban đầu.

5.3. Sinh hoán vị - Sinh tổ hợp (3/20)

5.3.1. Giới thiệu thuật toán sinh (3/4)

Ví dụ 5.3.1: Liệt kê các dãy nhị phân có độ dài 3.

- Cấu hình ban đầu **000**.
- Đưa ra cấu hình đã có.
- Từ cấu hình đang có, duyệt từ **phải** ⟶ **trái**:
  a. Nếu gặp phần tử "**0**" đầu tiên thì đổi thành "**1**" và đổi các phần tử bên phải của nó thành "**0**" quay lại bước b.
  b. Nếu không gặp phần tử "**0**" thì dừng lại.
- Dãy kết quả của bài toán:

'000' → '001' → '010' → '011' → '100' → '101' → '110' → '111'

5.3. Sinh hoán vị - Sinh tổ hợp (4/20)

5.3.1. Giới thiệu thuật toán sinh (4/4)

Trong ví dụ 5.3.1, giả mã thuật toán sinh kế tiếp được mô tả trong thủ tục sau:

```
void Next_Bit_String( int *B, int n ){
    i = n;
    while (b_i == 1 ) {
        b_i = 0
        i = i-1;
    }
    b_i = 1;
}
```

5.3. Sinh hoán vị - Sinh tổ hợp (5/20)

5.3.2. Sinh các hoán vị (1/9)

- Mọi phần tử của tập **n** phần tử Ω bất kỳ đều có thể tương ứng một-một với tập các số nguyên

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

- Để liệt kê các hoán vị của tập Ω ta có thể **sinh ra các hoán vị của tập A**, sau đó thay thể các số nguyên bằng các phần tử của Ω có chỉ số tương ứng.
- Phương pháp liệt kê các hoán vị của tập A được sử dụng: **phương pháp sinh hoán vị theo thứ tự từ điển**.

5.3. Sinh hoán vị - Sinh tổ hợp (6/20)

5.3.2. Sinh các hoán vị (2/9)

Khái niệm về phương pháp sinh hoán vị theo thứ tự từ điển:

- Hoán vị $a_1a_2\ldots a_n$ được gọi là **đi trước** (nhỏ hơn) hoán vị $b_1b_2\ldots b_n$ nếu với **k** nào đó ($1 \le k \le n$) ta có:

$$a_1 = b_1,\ a_2 = b_2,\ \ldots,\ a_{k-1} = b_{k-1},\ \text{và}\ a_k < b_k.$$

- **Giải thích**:
  - Một hoán vị của tập **n** số nguyên dương đầu tiên được gọi là đi trước (theo thứ tự từ điển) một hoán vị khác nếu tại vị trí đầu tiên mà hai hoán vị khác nhau, **con số của hoán vị đầu nhỏ hơn con số thuộc hoán vị thứ hai**.

5.3. Sinh hoán vị - Sinh tổ hợp (7/20)

5.3.2. Sinh các hoán vị (3/9)

Ví dụ 5.3.2:

- Xét tập A = {1, 2, 3, 4, 5}, khi đó:
  - Hoán vị 23145 của tập là đi trước hoán vị 23514,
    - vì những hoán vị này trùng nhau ở hai vị trí đầu tiên, nhưng số 1 ở vị trí thứ ba của hoán vị đầu nhỏ hơn số 5 cũng ở vị trí thứ ba của hoán vị sau.
  - Hoán vị 41532 là đi trước 52143.

5.3. Sinh hoán vị - Sinh tổ hợp (8/20)

5.3.2. Sinh các hoán vị (4/9)

Thuật toán sinh ra các hoán vị:

- **Phương pháp dựa trên việc xây dựng hoán vị kế tiếp theo thứ tự từ điển của hoán vị cho trước $a_1a_2…a_n$.**
- **Phân tích bài toán:**
  1. Giả sử $a_{n-1} < a_n$, nếu đổi chỗ a_{n-1} và a_n sẽ nhận được hoán vị mới đi ngay sau hoán vị đã cho.
  2. Nếu $a_{n-1} > a_n$ thì không thể nhận được hoán vị lớn hơn bằng cách đổi chỗ hai số hạng này trong hoán vị đã cho.
  3. Với trường hợp của ý 2: xem xét **ba số hạng cuối cùng trong hoán vị** $a_1a_2…a_n$.

5.3. Sinh hoán vị - Sinh tổ hợp (9/20)

5.3.2. Sinh các hoán vị (5/9)

- **Phân tích bài toán (tiếp):**

4. Nếu $a_{n-2} < a_{n-1}$ thì có thể sắp xếp lại ba số cuối cùng để có thể nhận được một hoán vị mới liền sau hoán vị xuất phát. Trong hai số a_{n-1} và a_n ta chọn số nhỏ hơn đặt vào vào vị trí **n - 2**, sau đó đặt số nguyên còn lại và số a_{n-2} vào hai vị trí cuối cùng theo thứ tự tăng dần.

5. Nếu $a_{n-2} > a_{n-1}$ (và $a_{n-1} > a_n$), thì không thể nhận được hoán vị lớn hơn bằng cách đổi chỗ 3 số hạng cuối cùng của hoán vị. Khi đó, xét 4 số hạng.

5.3. Sinh hoán vị - Sinh tổ hợp (10/20)

5.3.2. Sinh các hoán vị (6/9)

Phương pháp tổng quát tạo hoán vị liền sau:

1. Tìm cặp số nguyên liền kề đầu tiên tính từ **phải** sang bên **trái** của hoán vị mà số đầu nhỏ hơn số sau, tức là tìm các số nguyên a_j và a_{j+1} sao cho $a_j < a_{j+1}$ và $a_{j+1} > a_{j+2} > \ldots > a_n$.
2. Để nhận được hoán vị liền sau, ta đặt vào vị trí thứ **j** số nguyên **nhỏ nhất trong các số** $> a_j$ của tập $a_{j+1}, a_{j+2}, \ldots, a_n$, rồi liệt kê theo thứ tự tăng dần của các số còn lại của $a_{j+1}, a_{j+2}, \ldots, a_n$ vào các vị trí **j + 1, …, n**.

5.3. Sinh hoán vị - Sinh tổ hợp (11/20)

5.3.2. Sinh các hoán vị (7/9)

Thuật toán sinh hoán vị kế tiếp:

```
void Next_Permutation( int
*A, int n){
  int j, k, r, s, temp;
  j = n-1;
  while (a_j > a_j+1 )
    j = j -1;
  k = n;
  while (a_j > a_k )
    k= k - 1;
  temp =a_j;
  a_j = a_k;
  a_k = temp;
  r = j + 1;
  s = n;
  while ( r < s) {
    temp = a_r;
    a_r = a_s;
    a_s = temp;
    r = r +1;
    s = s - 1;
  }
}
```

5.3. Sinh hoán vị - Sinh tổ hợp (12/20)

5.3.2. Sinh các hoán vị (8/9)

- **Ví dụ 5.3.3:**

  Tìm hoán vị liền sau theo thứ tự từ điển của hoán vị

  3 6 2 5 4 1.

- **Lời giải:**
  - Cặp số nguyên đầu tiên phải tính từ phải qua trái có số trước nhỏ hơn số sau là $a_3 = 2$ và $a_4 = 5$.
  - Số nhỏ nhất trong các số bên phải của số 2 mà lại lớn hơn 2 là số 4.
  - Đặt số 4 vào vị trí thứ 3.
  - Sau đó đặt các số 2, 5, 1 theo thứ tự tăng dần vào ba vị trí còn lại.
  - Hoán vị liền sau theo thứ tự từ điển của hoán vị đã cho 362541 là

    3 6 4 1 2 5.

5.3. Sinh hoán vị - Sinh tổ hợp (13/20)

5.3.2. Sinh các hoán vị (9/9)

- **Ví dụ 5.3.4:**
  - Sử dụng thuật toán sinh hoán vị để tạo các hoán vị của các số nguyên 1, 2, 3 theo thứ tự từ điển.
- **Lời giải:**
  - Chúng ta bắt đầu bằng hoán vị 123.
  - Đổi chỗ 3 và 2 ta nhận được hoán vị tiếp theo là 132.
  - Tiếp theo, vì 3 > 2 và 1 < 3 nên phải xét cả 3 số trong nhóm hoán vị này.
  - Đặt số nhỏ nhất trong hai số 3 và 2 (số 2) vào vị trí thứ nhất, sau đó là hai số còn lại (1 và 3) vào vị trí thứ 2 và thứ 3 theo thứ tự tăng dần, tức là ta có 213.
  - Đổi chỗ 1 và 3 cho nhau vì 1 < 3 được 231. Vì 2 < 3 nên được đặt 3 vào vị trí đầu tiên và sau đó là 1 và 2, tức là 312.
  - Cuối cùng, đổi chỗ 1 với 2 ta nhận được hoán vị lớn nhất 321.
  - Kết quả trình tự các hoán vị xây dựng được là :

    123; 132; 213; 231; 312; 321

5.3. Sinh hoán vị - Sinh tổ hợp (14/20)

5.3.3. Sinh các tổ hợp (1/9)

Giới thiệu chung về sinh các tổ hợp:

- Tổ hợp của một tập hữu hạn chính là một tập con của tập đó.
- Có ánh xạ một- một giữa $\Omega = \{a_1, a_2, \ldots, a_n\}$ và các **xâu nhị phân độ dài n**.
- Một xâu nhị phân ứng với một tập con, tại ví trí **k**, sẽ có số 1 hoặc số 0 tuỳ theo phần tử a_k thuộc vào tập con đó hay không.
- Nếu có thể liệt kê tất cả các xâu nhị phân độ dài n thì ta sẽ nhận được tất cả các tập con của tập n phần tử (và cũng chính là các tổ hợp của tập đó).

5.3. Sinh hoán vị - Sinh tổ hợp (15/20)

5.3.3. Sinh các tổ hợp (2/9)

Giới thiệu chung về sinh các tổ hợp (tiếp):

- Mặt khác, một xâu nhị phân độ dài **n** cũng là khai triển nhị phân của một số nguyên nằm giữa **0** và $\mathbf{2^n - 1}$.
- Khi đó, $\mathbf{2^n}$ xâu nhị phân có thể liệt kê theo thứ tự tăng dần của số nguyên trong biểu diễn nhị phân của chúng.
- Sẽ bắt đầu từ xâu nhị phân bé nhất **00…0** (n số 0).
- Mỗi bước để tìm xâu liền sau một xâu nhị phân cho trước bao gồm việc tìm vị trí đầu tiên tính từ phải qua trái mà ở đó là số 0, sau đó thay tất cả số 1 ở bên phải số này bằng 0 và đặt số 1 vào chính vị trí này.

5.3. Sinh hoán vị - Sinh tổ hợp (16/20)

5.3.3. Sinh các tổ hợp (3/9)

Thuật toán tạo tổ hợp chập k từ n phần tử {1, 2,…, n}:

- Mỗi tổ hợp chập **k** có thể biểu diễn bằng một xâu tăng, do vậy có thể liệt kê các tổ hợp theo **thứ tự từ điển**.
- Theo thứ tự từ điển:
  - Tổ hợp đầu tiên là (**1, 2, . . ., k**) và
  - Tổ hợp cuối cùng là (**n-k+1, n-k+2, . . .,n**).
- Giả sử ($a_1a_2…a_k$) là tổ hợp đang có chưa phải là tổ hợp cuối cùng theo thứ tự từ điển, có thể xây dựng bằng cách thực hiện các quy tắc biến đổi sau đối với tổ hợp đang có:
  - **Tìm từ bên phải của dãy $a_1a_2…a_k$ phần tử $a_i \neq n-k+i$.**
  - **Thay a_i bởi a_i +1**
  - **Thay a_j bởi a_i +j - i với j = i+1, i+2, .., k.**

5.3. Sinh hoán vị - Sinh tổ hợp (17/20)

5.3.3. Sinh các tổ hợp (4/9)

Thuật toán liệt kê tập con kế tiếp m phần tử của tập n phần tử:

```
void Next_Combination( int *A, int n){
    i = n;
    while ( a_i == n-k+i)
        i = i -1;
    a_i = a_i + 1;
    for ( j = i+1; j <=n; j++)
        a_j = a_i + j - i;
}
```

5.3. Sinh hoán vị - Sinh tổ hợp (18/20)

5.3.3. Sinh các tổ hợp (7/9)

Ví dụ 5.3.5:

Tìm xâu nhị phân liền sau của 1 0001 1111.

Lời giải:

- Bit đầu tiên từ bên phải sang bằng 0 là bit thứ 6,
- Thay nó bằng số 1 và 5 bit 1 bên phải nó được thay bằng số 0,
- Từ đó ta được xâu nhị phân liền sau của xâu đã cho là:

1 0010 0000

5.3. Sinh hoán vị - Sinh tổ hợp (19/20)

5.3.3. Sinh các tổ hợp (8/9)

Ví dụ 5.3.6:

Tìm tổ hợp chập 4 từ tập {1, 2, 3, 4, 5, 6} đi liền sau tổ hợp {1, 2, 5, 6}.

Lời giải:

- Lùi từ **phải** qua **trái** ta thấy $a_2 = 2$ là số hạng đầu tiên của tổ hợp đã cho thoả mãn điều kiện $a_i \neq 6 - 4 + i$ (trong trường hợp này i=2).
- Để nhận được tổ hợp tiếp sau:
  - Tăng a_i lên một đơn vị, tức $a_2 = 3$,
  - Sau đó đặt $a_3 = 3+1 = 4$ và $a_4 = 3 + 2 = 5$.
- Vậy tổ hợp liền sau đã cho:

$$\{1, 3, 4, 5\}$$

5.3. Sinh hoán vị - Sinh tổ hợp (20/20)

5.3.3. Sinh các tổ hợp (9/9)

Ví dụ 5.3.7:

Liệt kê các tổ hợp chập 3 từ tập $\{1, 2, 3, 4, 5\}$

Lời giải:

- Ta có: $C_5^3 = \frac{5!}{(5-3)!2!} = 10$
- Áp dụng thuật toán bắt đầu từ tổ hợp {1, 2, 3} ta thu được tất cả các tổ hợp cần tìm:

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

5.4. Giải thuật quay lui. Bài toán sắp hậu (1/15)

5.4.1. Giải thuật quay lui (1/9)

- **Ý tưởng của giải thuật:**
  - Giả sử cần tìm cấu hình gồm **n** thành phần $x_1, x_2,...x_n$.
  - Giả sử đã xác định được **i-1** thành phần $x_1, x_2,..,x_{i-1}$, khi đó, cần xác định thành phần x_i.
  - Với mỗi khả năng **j**, kiểm tra xem **j** có chấp nhận? với $j \in \{1, n_i\}$.
  - Xảy ra 2 trường hợp :
    - 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}.
    - Nếu thử tất cả khả năng mà không có khả năng nào được chấp nhận thì quay lại bước trước để xác định **lại** x_{i-1}.

5.4. Giải thuật quay lui. Bài toán sắp hậu (2/15)

5.4.1. Giải thuật quay lui (2/9)

- **Chú ý:**
  - 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.
  - Những thông tin này cần được lưu trữ theo cơ cấu ngăn xếp (stack).

5.4. Giải thuật quay lui. Bài toán sắp hậu (3/15)

5.4.1. Giải thuật quay lui (3/9)

Giả mã của giải thuật quay lui, xác định phần tử thứ i:

```
void Try( int i ) {
      int j;
      for ( j = 1; j < ni; j ++) {
            if ( <Chấp nhận j >) {
                  <Xác định xi theo j>
                  if (i==n)
                        <Ghi nhận cấu hình>;
                  else Try(i+1);
            }
      }
}
```

5.4. Giải thuật quay lui. Bài toán sắp hậu (4/15)

5.4.1. Giải thuật quay lui (4/9)

Ví dụ 5.4.1:

Liệt kê các dãy nhị phân độ dài n.

Gợi ý:

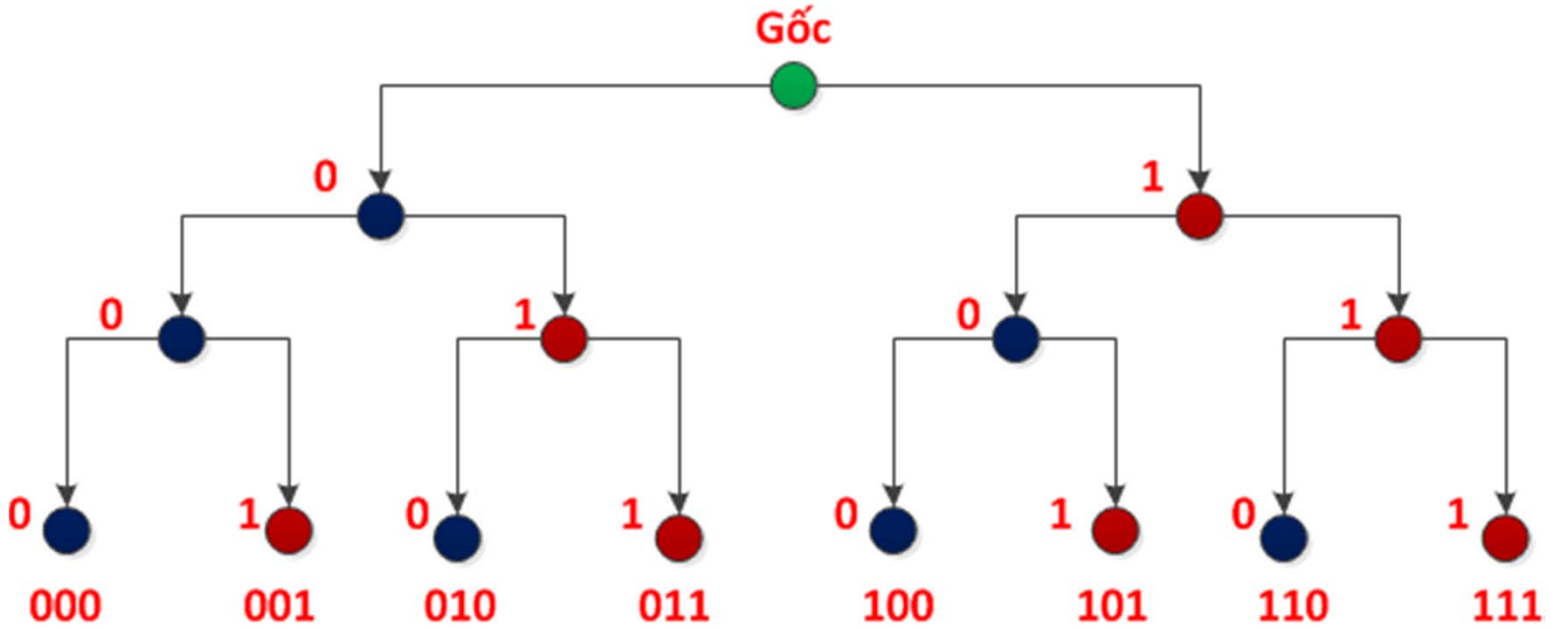
- Biểu diễn dãy nhị phân dưới dạng $b_1,b_2,...b_n$ trong đó $b_i \in \{0,1\}$.
- Thủ tục đệ quy để xác định b_i, trong đó $b_i = 0$ hoặc $b_i = 1$. Lưu ý: các giá trị này mặc nhiên được chấp nhận mà không phải thoả mãn điều kiện gì.
- Thủ tục khởi tạo **n**, khởi tạo biến đếm số kết quả có được.
- Thủ tục in kết quả.

5.4. Giải thuật quay lui. Bài toán sắp hậu (5/15)

5.4.1. Giải thuật quay lui (5/9)

Ví dụ 5.4.1. (tiếp): Liệt kê các dãy nhị phân độ dài n.

Giải thích bằng đồ thị:



Cây tìm kiếm lời giải liệt kê dãy nhị phân độ dài 3

5.4. Giải thuật quay lui. Bài toán sắp hậu (6/15)

5.4.1. Giải thuật quay lui (6/9)

Ví dụ 5.4.1. (tiếp): Liệt kê các dãy nhị phân độ dài n.

Mã chương trình:

```
#include <conio.h>
#include <iostream>
using namespace std;
void Result(int *B, int n){
  for(int i=0;i<n;i++)
    cout<<"  "<<B[i]<<"   ";
    cout<<endl;
}
void Init(int *B, int n){
  for(int i=0;i<n;i++) B[i]=0;
}
```

```
void Try(int i, int *B, int n){
  for(int j=0; j<=1;j++){
    B[i]=j;
    if(i==n-1) { Result(B,n); }
    else Try(i+1, B, n);
  }
}
void main(void){
  int *B,n;
  cout<<"\n Nhap n="; cin>>n;
  B=new int[n];
  Init(B,n);
  Try(0,B,n);
  delete B;
  getch();
}
```

5.4. Giải thuật quay lui. Bài toán sắp hậu (7/15)

5.4.1. Giải thuật quay lui (7/9)

Ví dụ 5.4.2:

Liệt kê các hoán vị của {1,2,...,n}.

Gợi ý:

- Biểu diễn hoán vị dưới dạng **$p_1, p_2,..,p_n$**, trong đó:

$$p_i \in \{1, \dots, n\};\ p_i \neq p_j \text{ với } i \neq j.$$

- Các giá trị từ 1 đến n được lần lượt đề cử cho p_i, trong đó giá trị j được chấp nhận nếu nó chưa được dùng.
- Sử dụng 1 mảng biến logic **b**, để xem 1 phần tử đã dùng chưa.
- Sau khi gán j cho p_i cần được ghi nhận false cho b_j và phải gán lại true khi thực hiện xong việc in giá trị hoặc khi thử tiếp phương án mới thứ **i+1**.
- Các phần còn lại tương tự ví dụ 5.4.1.

5.4. Giải thuật quay lui. Bài toán sắp hậu (8/15)

5.4.1. Giải thuật quay lui (8/9)

Ví dụ 5.4.2:

Liệt kê các hoán vị của {1,2,...,n}.

Đoạn giả mã của giải thuật:

```
void Try(int i){
  int j;
  for(j=1; j<=n;j++){
    if(B[j]) {
      P[i]=j;
      B[j]=FALSE;
      if(i==n) In_Kết_Quả;
      else Try(i+1);
      B[j]=TRUE;
    }
  }
}
```

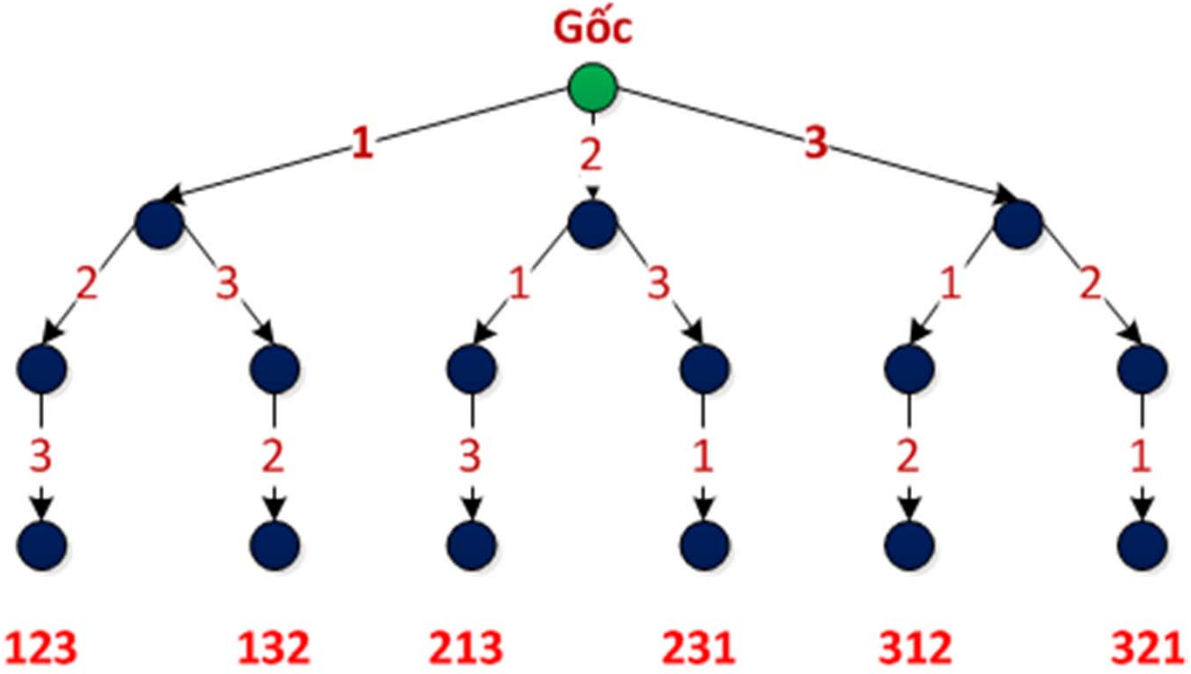
5.4. Giải thuật quay lui. Bài toán sắp hậu (9/15)

5.4.1. Giải thuật quay lui (8/9)

Ví dụ 5.4.2:

Liệt kê các hoán vị của {1,2,...,n}.

Minh họa bằng đồ thị:



5.4. Giải thuật quay lui. Bài toán sắp hậu (10/15)

5.4.1. Giải thuật quay lui (9/9)

Ví dụ 5.4.3:

Liệt kê các tổ hợp chập k của tập {1,2..., n}.

Gợi ý:

- Biểu diễn tổ hợp dưới dạng $c_1, c_2...,c_k$, trong đó:

$$1 \le c_1 < c_2 < ...< c_k \le n$$

- Các giá trị đề cử cho c_i : $\in \{c_{i-1}+1 , n-k+i\}$.
- Để điều này đúng cho cả trường hợp i = 1, cần thêm vào c_0 với $c_0 = 0$.
- Các giá trị đề cử này mặc nhiên được chấp nhận.
- Các thủ tục khởi tạo, in kết quả được xây dựng tương tự ví dụ 5.4.2.

5.4. Giải thuật quay lui. Bài toán sắp hậu (11/15)

5.4.1. Giải thuật quay lui (9/9)

Ví dụ 5.4.3 (tiếp):

Liệt kê các tổ hợp chập k của tập {1,2..., n}.

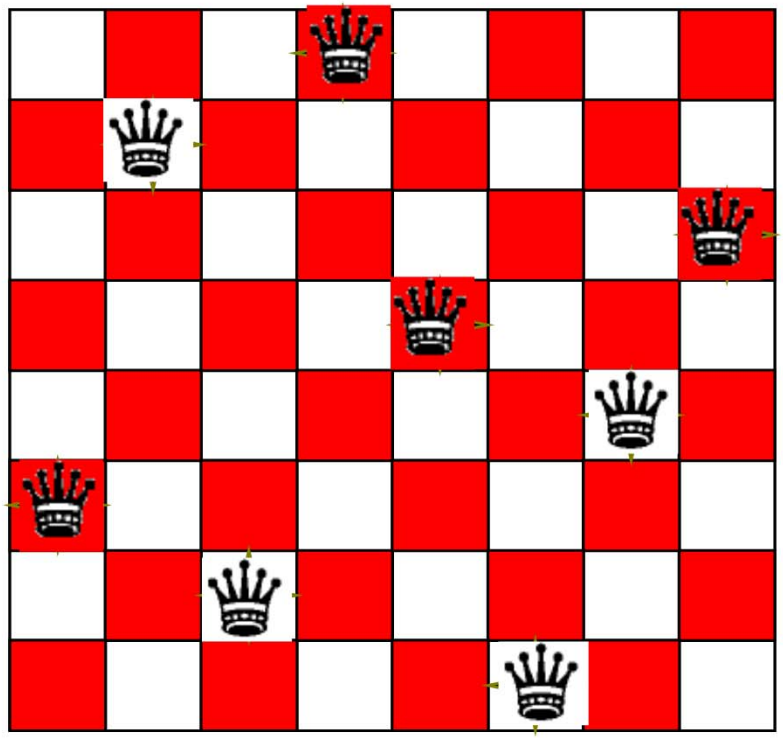
Đoạn giả mã gợi ý:

```
void Try(int i){
  int j;
  for(j=B[i-1]+1;j<=(n-k+i); j++){
    B[i]=j;
    if(i==k) In_Kết_Quả;
    else Try(i+1);
  }
}
```

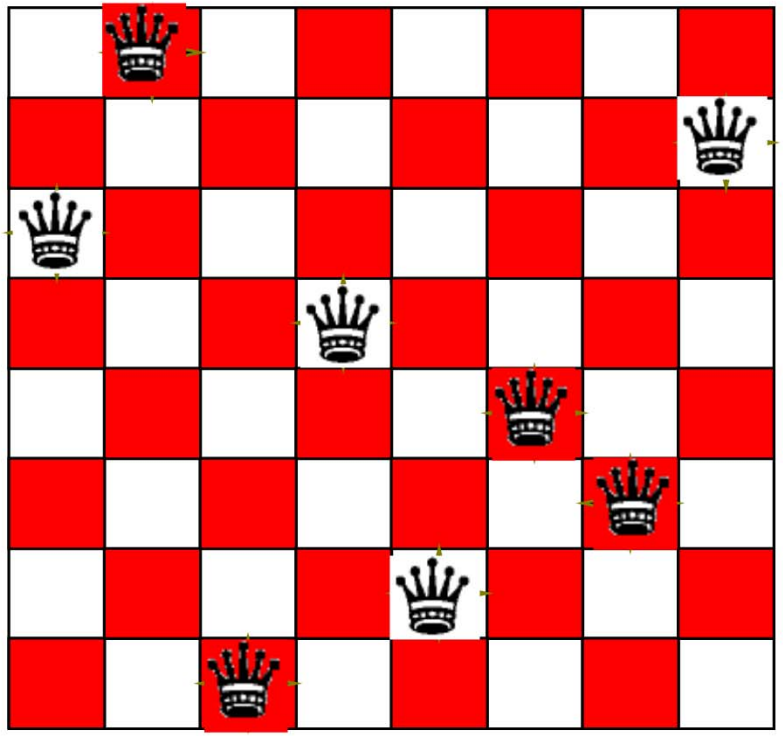
5.4. Giải thuật quay lui. Bài toán sắp hậu (12/15)

5.4.2. Bài toán sắp hậu (1/4)

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



Một lời giải

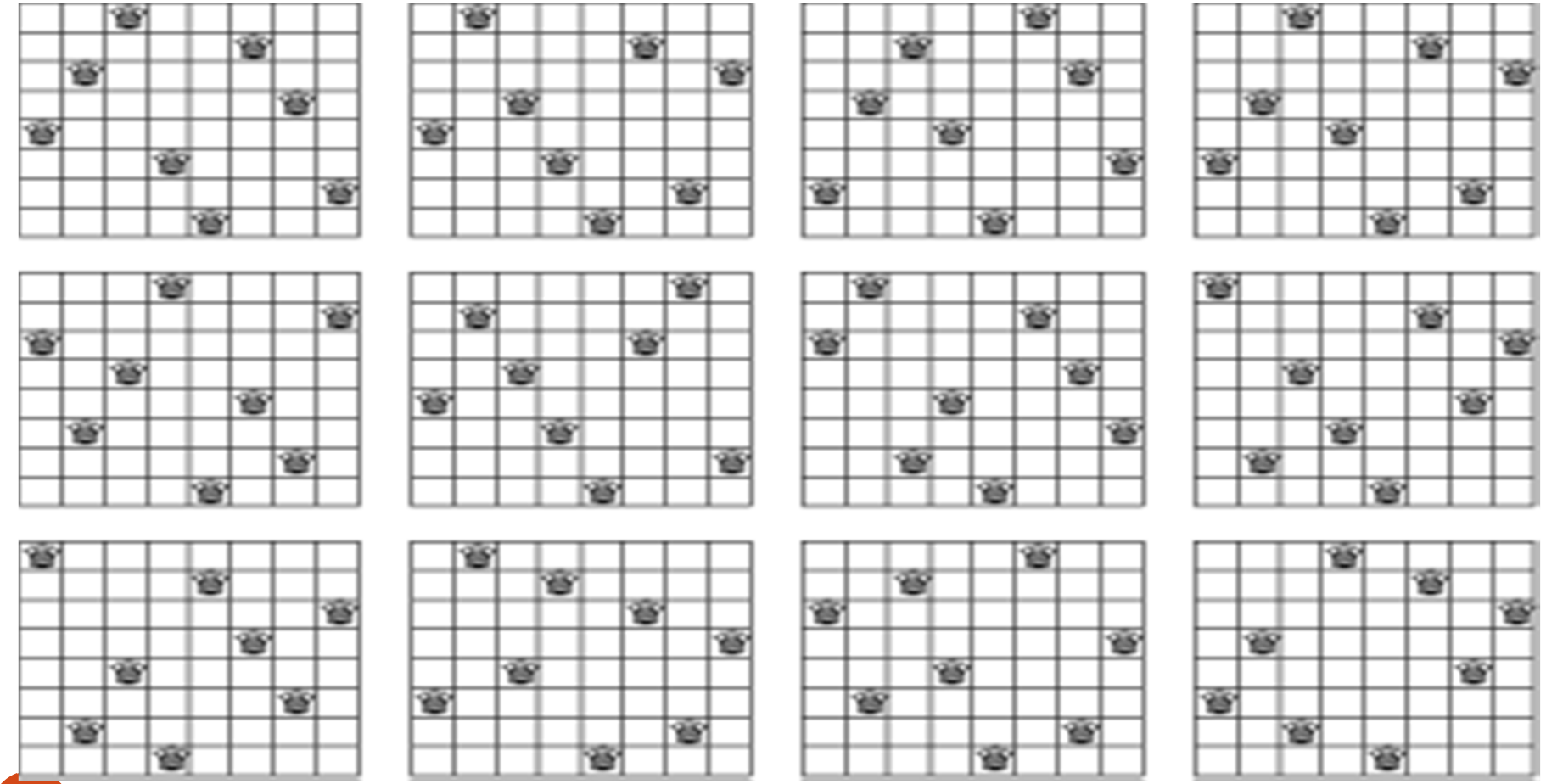


Không phải lời giải

5.4. Giải thuật quay lui. Bài toán sắp hậu (13/15)

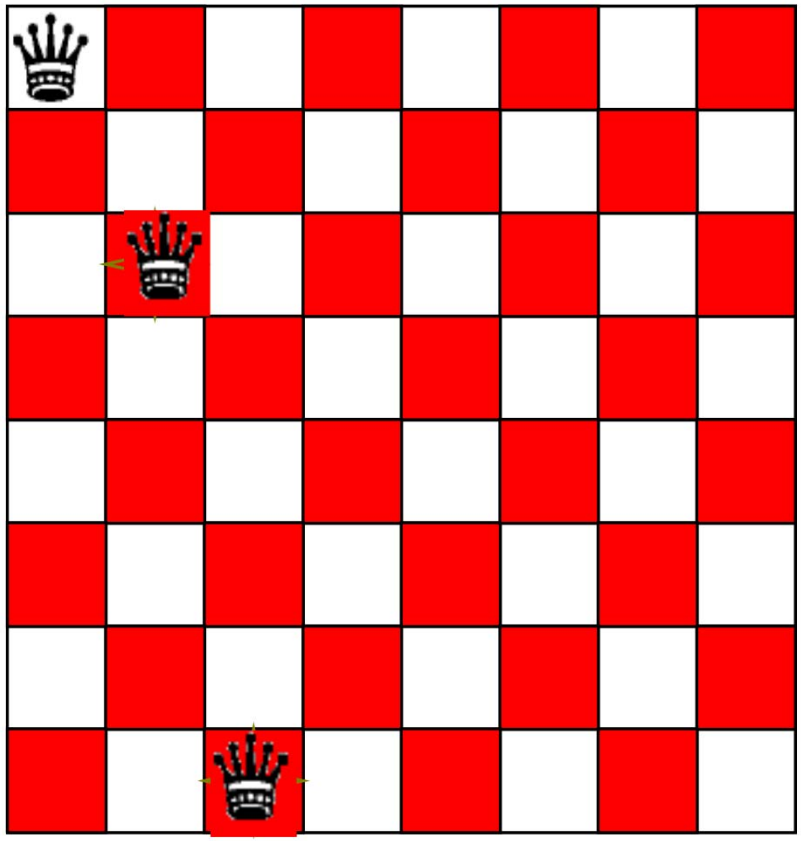
5.4.2. Bài toán sắp hậu (2/4)

Một vài cách đặt 8 con hậu lên bàn cờ (trong số 92 lời giải)



5.4. Giải thuật quay lui. Bài toán sắp hậu (14/15)

5.4.2. Bài toán sắp hậu (3/4)



Thực hiện:

- Trạng thái: Sắp đặt từ 0 đến 8 con hậu lên bàn cờ.
- Trạng thái khởi tạo: không có con hậu nào đặt lên bàn cờ.
- Đặt từng con hậu lên bàn cờ.
- Kiểm tra kết quả: 8 con hậu đã đặt lên bàn cờ, không có con hậu nào ăn được nhau.

→ **64^8 cách đặt 8 con hậu**

5.4. Giải thuật quay lui. Bài toán sắp hậu (15/15)

5.4.2. Bài toán sắp hậu (4/4)

Ý tưởng:

- Mỗi lần đệ quy, tìm cách đặt 1 con hậu lên một cột (hoặc hàng) riêng.
- Với mỗi lần gọi, tình trạng bàn cờ đó biết (các con hậu đó đặt).
- Nếu tại một cột, không đặt được con hậu mới, con hậu ở cột (hàng) đó được bỏ ra khỏi bàn cờ và chuyển xuống cột (hàng) trước đó.
- Nếu xét theo cột, tại một cột, tất cả các hàng đã xét, quay trở lại bước trước đó (xét cột trước).
- Nếu con hậu không đặt được lên cột i, khi đó không thử trên cột i+1, quay lại cột i-1, bỏ con hậu đã đi sai.
- Với cách tiếp cận này, có thể giảm bớt phép thử.

5.5. Bài tập chương 5.

Bài 1. Liệt kê tất cả các tập con của tập 1, 2,..,n.

Bài 2. Liệt kê tất cả các xâu nhị phân độ dài n có tổng các bít 1 đúng bằng k≤n.

Bài 3. Liệt kê tất cả các xâu nhị phân độ dài 5 không chứa hai số 0 liên tiếp.

Bài 4. Liệt kê tất cả các phần tử của tập:

$$D = \{x = (x_1, x_2, \cdots, x_n\} : \sum_{j=1}^{n} a_j x_j = b, \quad x_j \in \{0,1\}, j = 1,2,\cdots,n$$

Trong đó $a_1,a_2,..,a_n$, b là các số nguyên dương.

Bài 5. Liệt kê tất cả các phần tử của tập:

$$D = \{x = (x_1, x_2, \cdots, x_n\} : \sum_{j=1}^{n} a_j x_j = b, \quad x_j \in Z_+, j = 1,2,\cdots,n$$

Trong đó $a_1,a_2,..,a_n$, b là các số nguyên dương.