Chương 3

# Bài toán liệt kê (duyệt)

- Phương pháp đệ quy (quay lui)
- Phương pháp lặp (sinh kế tiếp)

# 3.1. Phương pháp đệ quy

- Nguyên lý chung
- Bài toán 8 hậu hòa bình
- Liệt kê các cấu hình tổ hợp cơ bản

# 3.1.1. Nguyên lý chung

Để liệt kê tất cả các cấu hình $S=s_1s_2..s_k$ thuật toán giả sử đã có cấu hình con $s_1s_2..s_{i-1}$. Ở bước thử tìm giá trị cho $s_i$, Try(i), duyệt qua mọi giá trị *j* đề cử được cho $s_i$ và thực hiện các bước sau:

# 3.1.1. Nguyên lý chung

Với mỗi giá trị *j* đề cử được cho $s_i$ thực hiện 4 bước sau:

1) $s_i$=j;
2) <*Thay đổi trạng thái*>;
3) If (i= =k) Print(S); else Try(i+1);
4) <*Trả lại trạng thái cũ*>;

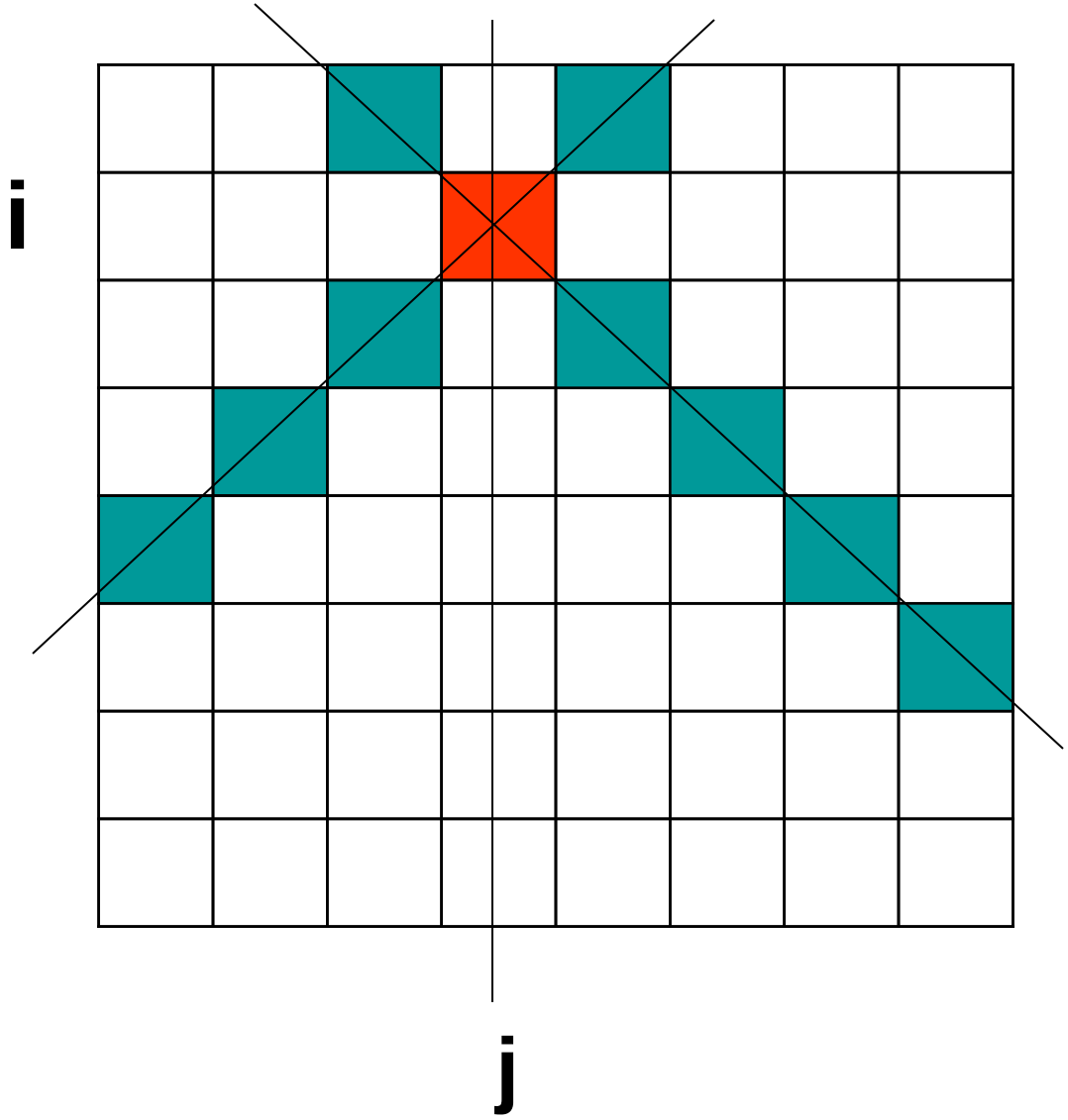
Chương trình chính gọi *Try(1)*

# 3.1.1. Nguyên lý chung

*Lưu ý*: *Một số cấu hình không có hai bước 2) và 4)*

Trong việc mô tả các thuật toán liệt kê các cấu hình tổ hợp cơ bản, luôn giả sử tập nguồn X ={1..n}. Trong trường hợp cụ thể thì lấy các phần tử của X làm chỉ số của mảng. Không dùng S[0]

# Bài toán 8 hậu



# Bài toán 8 hậu

*Khai báo*

- Bàn cờ: int S[8];
- Các mảng cần đánh dấu
  - Cột: int a [8];
  - Song song chéo chính: int b[15];
  - Song song chéo phụ: int c[15];

*Trước khi gọi Try(0) cần lấp đầy*

*a,b,c giá trị 1* (TRUE)

# Bài toán 8 hậu

```
void Try(int i)
{
  int j;
  for (j=0; j<8; j++) if (a[j]&&b[i-j+7]&&c[i+j]){
    S[i]=j;
    a[j]=0; b[i-j+7]=0; c[i+j]=0;
    if (i==7) print( ); else Try(i+1);
    a[j]=1; b[i-j+7]=1; c[i+j]=1;
  }
}
```

# Bài toán 8 hậu

8Hậu.CPP

# 3.1.2. Liệt kê cấu hình tổ hợp

Trong việc mô tả các thuật toán liệt kê các cấu hình tổ hợp cơ bản, luôn giả sử tập nguồn X ={1..n}. Trong trường hợp cụ thể thì lấy các phần tử của X làm chỉ số của mảng. Không dùng S[0]

# 3.1.2. Liệt kê chỉnh hợp lặp

*Không có hai bước 2) và 4)*

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

# 3.1.2. Liệt kê chỉnh hợp lặp

ChinhHopLap.CPP

# 3.1.3. Liệt kê chỉnh hợp

*Cần mảng a đánh dấu các giá trị j đã dùng*

```
void Try(int i)
{  int j;
  for (j=1; j<=n; j++) if (a[j]){
     S[i]=j;
     a[j]=0;
     if (i==k) print( ); else Try(i+1);
     a[j]=1;
  }
}
```

# 3.1.3. Liệt kê chỉnh hợp

ChinhHop.CPP

- Hoán vị là chỉnh hợp với $k=n$

# 3.1.4. Liệt kê tổ hợp

*Liệt kê theo thứ tự tăng.Giá trị j đề cử được cho $s_i$ là:*

*$1+s_{i-1} \le j \le n-k+i$. Gọi Try(1)=> mảng S có S[0]=0.*

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

# 3.1.4. Liệt kê tổ hợp

ToHop.CPP

# 3.2. Phương pháp lặp

- Nguyên lý chung
- Liệt kê các cấu hình tổ hợp cơ bản

# 3.2.1. Nguyên lý chung

Liệt kê tất cả các cấu hình $S=s_1s_2..s_k$ theo thứ tự từ điển, gồm các bước sau:

- Tìm S nhỏ nhất; Print(S);
- Lặp cho đến khi đủ số cấu hình:

*Tìm S tiếp theo; Print(S);*

*Hay* Trong khi S chưa phải lớn nhất:

*Tìm S tiếp theo; Print(S);*

# 3.2.1. Nguyên lý chung

Bước *Tìm S tiếp theo* S hiện tại thì tùy theo từng cấu hình cụ thể mà thay đổi S như sau:

- *Tìm S[i] bên phải nhất có thể tăng được*
- *Tăng S[i] với giá trị tăng nhỏ nhất*
- *Hạ cấu hình con S[i+1..k] xuống nhỏ nhất*

*Cũng có thể vừa tìm S[i] vừa kiểm tra dừng*

# 3.2.2. Liệt kê chỉnh hợp lặp

```
for (i=1; i<=k; i++) S[i]=1; print(); c=1;
  while (c<=n^k){
    i=k; while (S[i]==n) i--;
    S[i]++;
    for (j=i+1; j<=k; j++) S[j]=1;
    print(); c++;
  }
```

# 3.2.3. Liệt kê tổ hợp

*Tổ hợp là bộ không thứ tự. Liệt kê theo thứ tự tăng: $s_i<s_{i+1}$.Giá trị lớn nhất cho $s_i$ là n-k+i.*

```
for (i=1; i<=k; i++) S[i]=i; print(); c=1;
  while (c<C(n,k)){
    i=k; while (S[i]==n-k+i) i--;
    S[i]++;
    for (j=i+1; j<=k; j++) S[j]=S[j-1]+1;
    print( ); c++;
  }
```

# 3.2.4. Liệt kê hoán vị

*nhỏ nhất theo thứ tự tăng và lớn nhất theo thứ tự giảm.*

```
for (i=1; i<=n; i++) S[i]=i; print( ); c=1;
  while (c<n!) {
    i=n-1; while (S[i]>S[i+1]) i--;
    j=n; while (S[j]<S[i]) j--;
    tam=S[i]; S[i]=S[j]; S[j]=tam;
    j=i+1; k=n; while (j<k){
        tam=S[j]; S[j]=S[k]; S[k]=tam; j++; k--;}
    print( ); c++;
  }
```

# 3.2.4. Liệt kê chỉnh hợp

*Đưa thuật toán sinh hoán vị vào mỗi bước sinh tổ hợp có được thuật toán sinh chỉnh hợp không lặp.*