2.3. Thuật toán đệ qui (Recursion)

Phương pháp định nghĩa bằng đệ qui: Một đối tượng được định nghĩa trực tiếp hoặc gián tiếp thông qua chính nó được gọi là phép định nghĩa bằng đệ qui.

Thuật toán đệ qui: Thuật toán giải bài toán P trực tiếp hoặc gián tiếp thông qua bài toán P' giống như P được gọi là thuật toán đệ qui. Một hàm được gọi là đệ qui nếu nó được gọi trực tiếp hoặc gián tiếp đến chính nó. Một bài toán giải được bằng đệ qui nếu nó thỏa mãn hai điều kiện:

- **Phân tích được**: *Có thể giải được bài toán P bằng bài toán P' giống như P và chỉ khác P ở dữ liệu đầu vào. Việc giải bài toán P' cũng được thực hiện theo cách phân tích giống như P.*
- **Điều kiện dừng**: *Dãy các bài toán P' giống như P là hữu hạn và sẽ dừng tại một bài toán xác định nào đó.*

Thuật toán đệ qui tổng quát có thể được mô tả như sau:

Thuật toán Recursion (P) {

1. *Nếu P thỏa mãn điều kiện dừng:*

   <Giải P với điều kiện dừng>;

2. Nếu P không thỏa mãn điều kiện dừng:

   Recursion(P').

}

Ví dụ: Tìm tổng của n số tự nhiên bằng phương pháp đệ qui.

Lời giải. Gọi Sn là tổng của n số tự nhiên. Khi đó:

- **Bước phân tích**: Sn = n + S(n-1), n>1.
- **Điều kiện dừng**: s1 = 1 nếu n=1;

Từ đó ta có lời giải của bài toán như sau:

```
int     Tong (int  i ) {
        if (i ==1 ) return(1); //Điều kiện dừng
        else return(i + Tong(i-1)); //Điều kiện phân tích được
}
```

Ví dụ. Tìm n!.

Lời giải. Gọi Sn là n!. Khi đó:

- **Bước phân tích**: Sn = n*(n-1)! nếu n>0;
- Điều kiện dừng: s0=1 nếu n=0.

Từ đó ta có lời giải của bài toán như sau:

```
long    Giaithua (int  i ) {
        if (i ==0 ) return(1); //Điều kiện dừng
        else return(i *Giaithua(i-1)); //Điều kiện phân tích được
}
```

Ví dụ: Tìm ước số chung lớn nhất của a và b bằng phương pháp đệ qui.

Lời giải. Gọi d =USCLN(a,b). Khi đó:

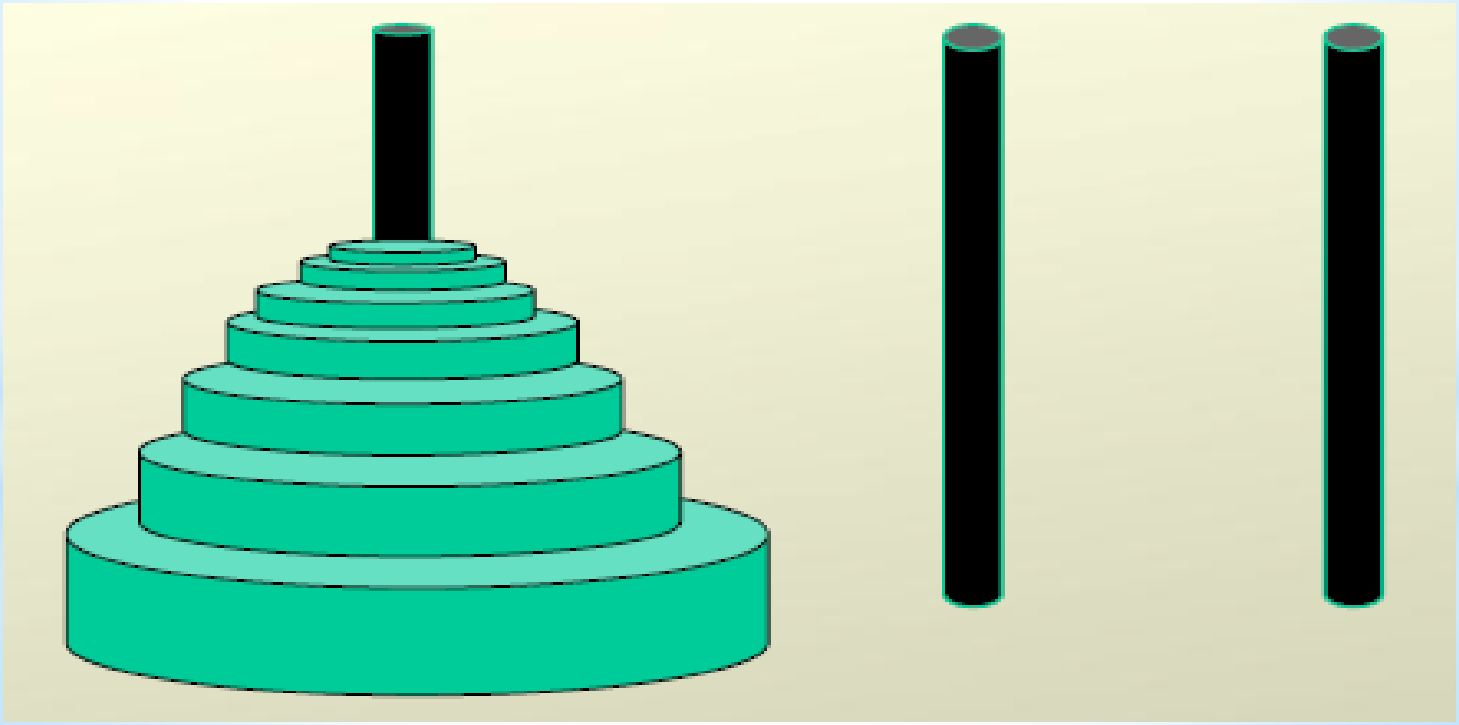
- **Bước phân tích**:
  - d = USCLN(a-b, b) nếu a>b.
  - d = USCLN(a, b-a) nếu a<b.
- **Điều kiện dừng**: d =a hoặc d=b nếu a=b;

Từ đó ta có lời giải của bài toán như sau:

```
int     USCLN (int a, int b ) {
        if (a ==b ) return(a); //Điều kiện dừng
        else { //Điều kiện phân tích được
                if (a> b) return(USCLN(a-b, b));
                else return(USCLN(a, b-a));
        }
}
```

Ví dụ. *Bài toán tháp Hà Nội*. Có n đĩa bạc có đường kính khác nhau được đặt chồng lên nhau. Các đĩa được đặt chồng lên nhau theo thứ tự giảm dần của đường kính. Có ba vị trí A, B, C có thể đặt đĩa. Chồng đĩa ban đầu đặt ở vị trí A. Một nhà sư chuyển các đĩa từ vị trí A sang vị trí B thỏa mãn điều kiện:

- Khi di chuyển một đĩa phải đặt vào ba vị trí đã cho.
- Mỗi lần chuyển một đĩa và phải là đĩa trên cùng.
- Tại một vị trí đĩa được đặt cũng phải lên trên cùng.
- Đĩa lớn hơn không được phép đặt trên đĩa nhỏ hơn.



Lời giải.

- Nếu n=1 thì ta chỉ cần dịch chuyển một lần là xong.
- Với n>1: ta cần dịch chuyển n-1 đĩa còn lại từ cột A sang cột C, sau đó dịch chuyển một đĩa cuối cùng từ cột A sang cột B. Sau đó ta lại phải chuyển n-1 đĩa từ cột C sang cột A lấy cột B làm cột trung gian.

```
Thuật toán Hanoi-Tower (n, A, B, C) {
    If ( n==1 )
            <Dịch chuyển đĩa cuối cùng từ A sang C>;
    Else {
            Hanoi-Tower(n-1,  A, C, B);
             Hanoi-Tower(n-1,  C, B, A);
    }
}
```

2.4. Thuật toán quay lui (Back track)

Giả sử ta cần xác định bộ X =(x_1, x_2,..,x_n) thỏa mãn một số ràng buộc nào đó. Ứng với mỗi thành phần x_i ta có n_i khả năng cần lựa chọn. Ứng với mỗi khả năng $j \in n_i$ dành cho thành phần x_i ta cần thực hiện:

- *Kiểm tra xem khả năng j có được chấp thuận cho thành phần x_i hay không? Nếu khả năng j được chấp thuận thì nếu i là thành phần cuối cùng (i=n) ta ghi nhận nghiệm của bài toán. Nếu i chưa phải cuối cùng ta xác định thành phần thứ i +1.*
- *Nếu không có khả năng j nào được chấp thuận cho thành phần x_i thì ta quay lại bước trước đó (i-1) để thử lại các khả năng khác.*

```
Thuật toán Back-Track ( int i ) {
    For ( j =<Khả năng 1>; j <=n_i;  j++ ){
        if (<chấp thuận khả năng j>) {
            X[i] = <khả năng j>;
            if ( i ==n) Result();
            else Back-Track(i+1);
        }
    }
```

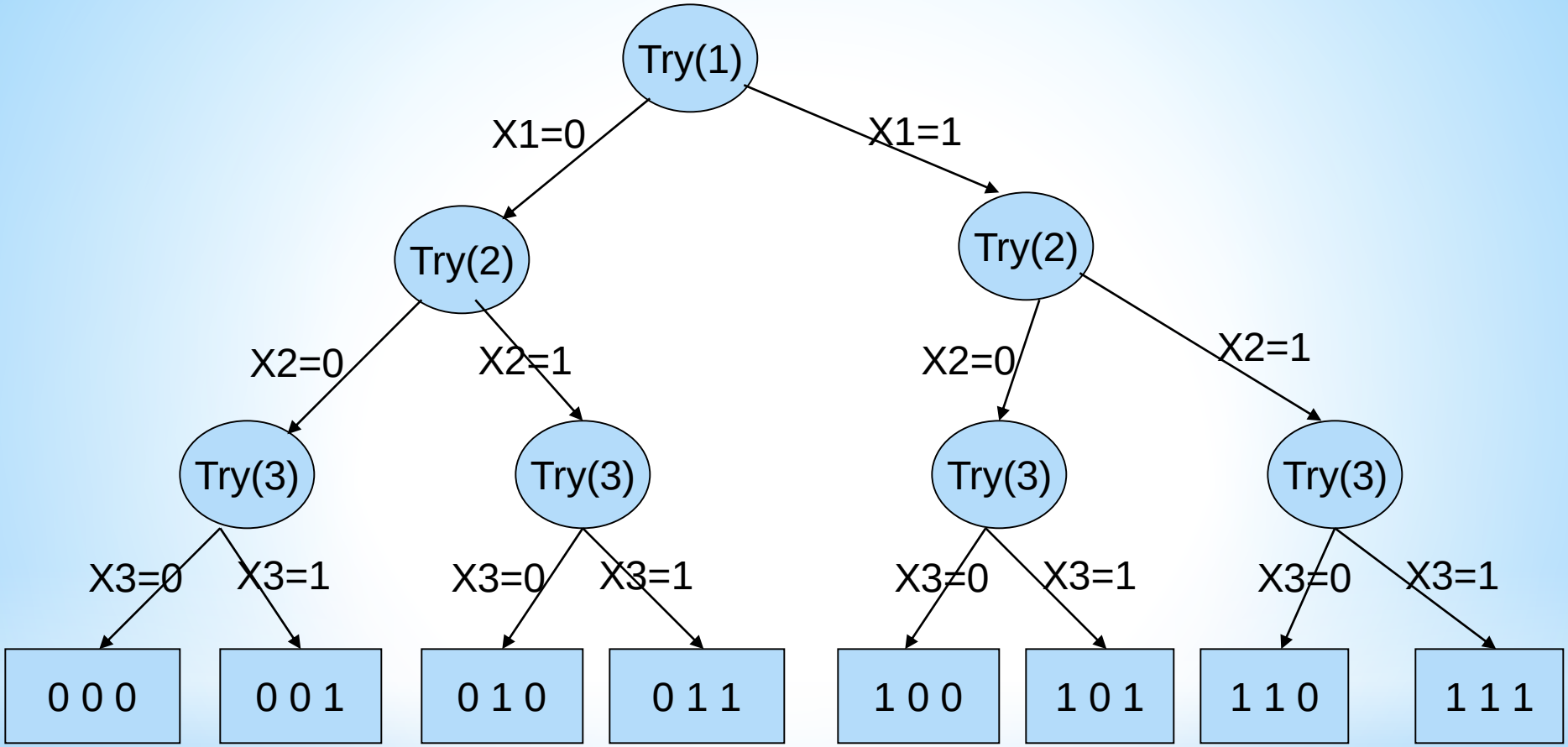
Ví dụ 1. Duyệt các xâu nhị phân có độ dài n.

Lời giải. Xâu nhị phân $X = (x_1, x_2,..,x_n)| x_i = 0, 1$. Mỗi $x_i \in X$ có hai lựa chọn $x_i = 0, 1$. Cả hai giá trị này đều được chấp thuận mà không cần có thêm bất kỳ điều kiện gì. Thuật toán được mô tả như sau:

```
Void Try ( int i ) {
        for (int j =0; j<=1; j++){
                X[i] = j;
                if ( i ==n) Result();
                else Try (i+1);
        }
}
```

Khi đó, việc duyệt các xâu nhị phân có độ dài n ta chỉ cần gọi đến thủ tục Try(1). Cây quay lui được mô tả như hình dưới đây.

Cây đệ qui duyệt các xâu nhị phân độ dài n = 3.



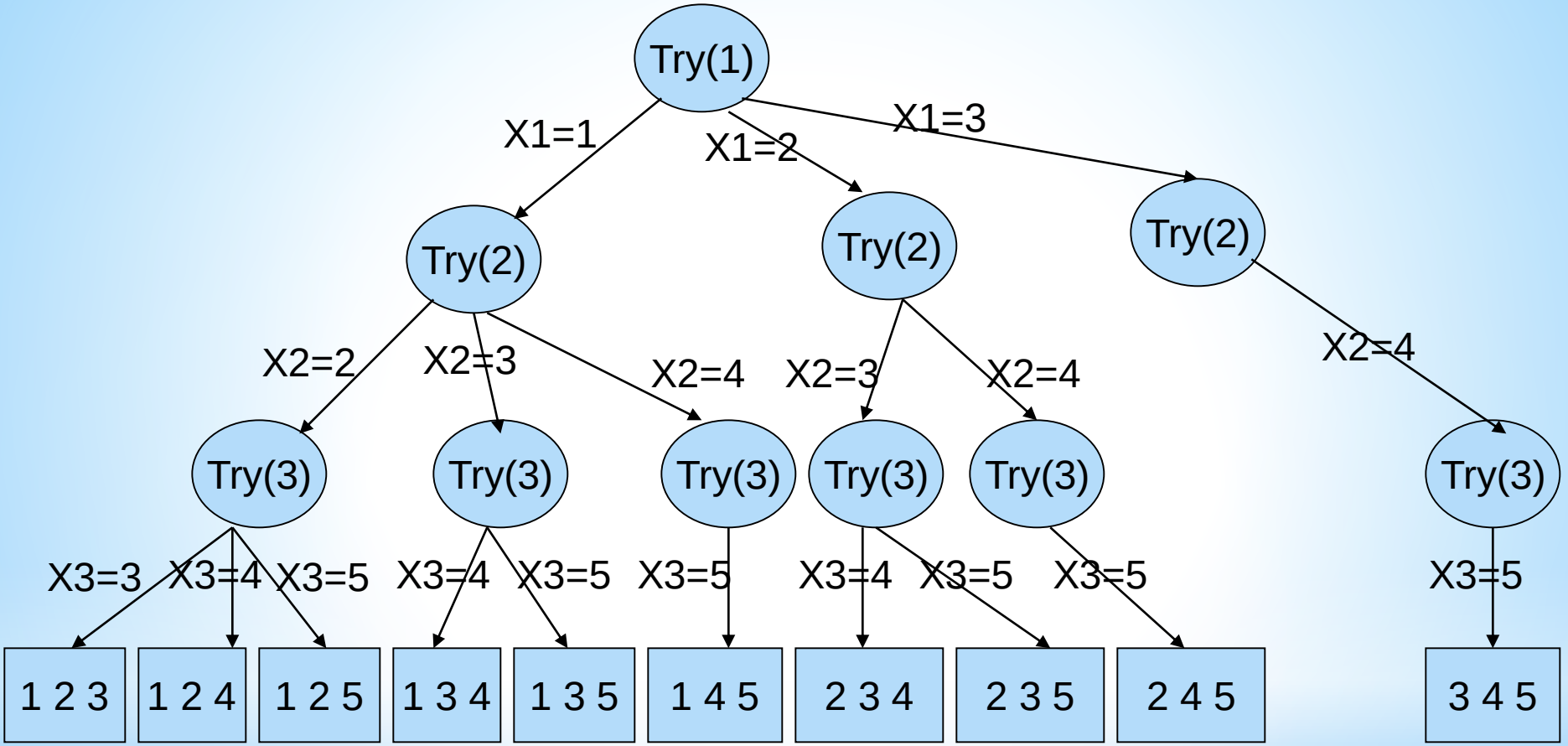
Ví dụ 2. Duyệt các tập con K phần tử của 1, 2, .., N.

Lời giải. Mỗi tập con K phần tử $X = (x_1, x_2,..,x_K)$ là bộ không tính đến thứ tự K phần tử của 1, 2, .., N. Mỗi $x_i \in X$ có N-K+i lựa chọn. Các giá trị này đều được chấp thuận mà không cần có thêm bất kỳ điều kiện gì. Thuật toán được mô tả như sau:

```
Void Try ( int i ) {
        for (int j =X[i-1]+1; j<=N-K+ i; j++){
                X[i] = j;
                if ( i ==K) Result();
                else Try (i+1);
        }
}
```

Khi đó, việc duyệt các tập con K phần tử của 1, 2, .., N ta chỉ cần gọi đến thủ tục Try(1). Cây quay lui được mô tả như hình dưới đây.

Cây đệ qui duyệt các tập con 3 phần tử của 1, 2, .., 5.



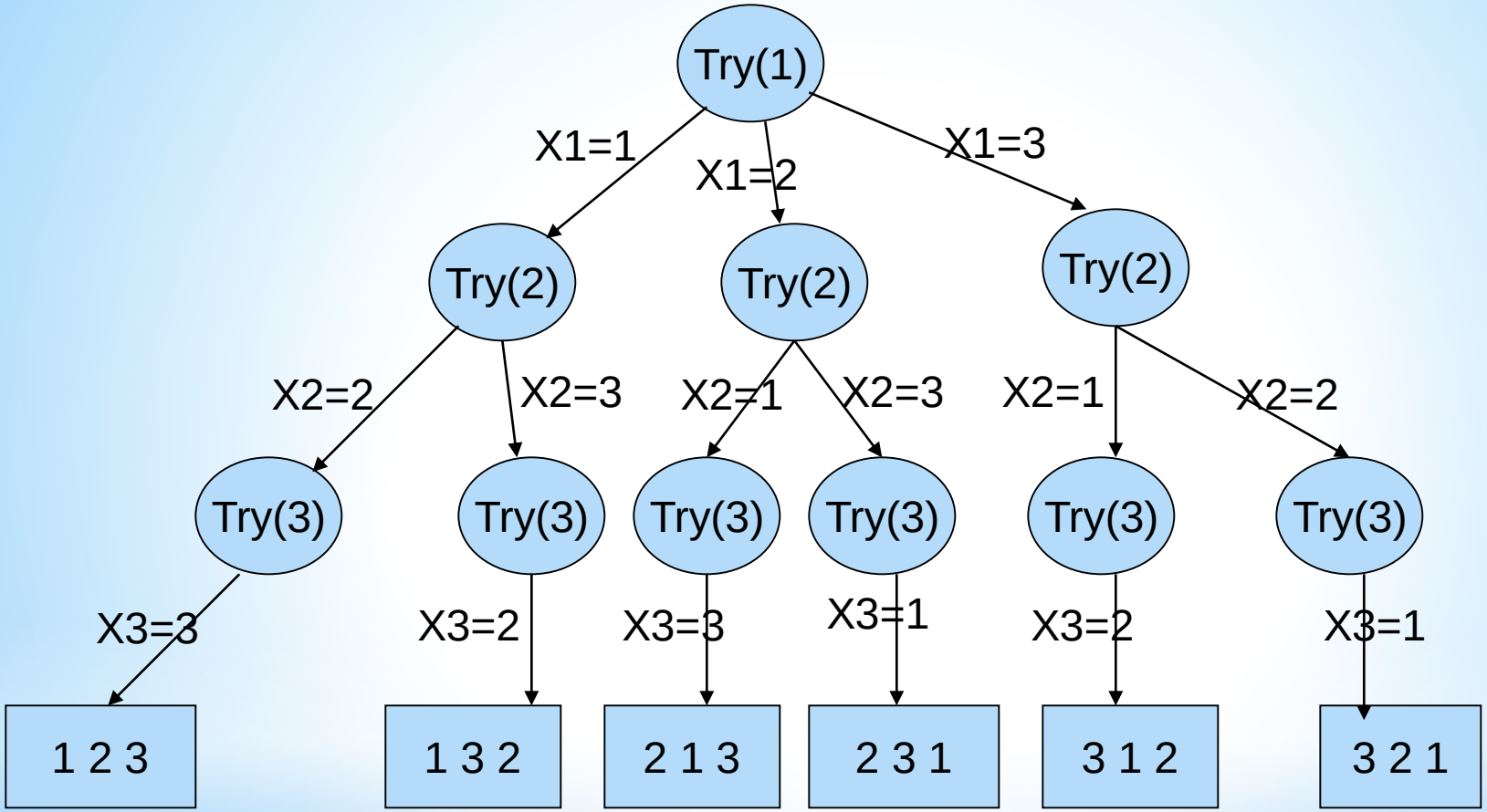
Ví dụ 3. Duyệt các hoán vị của 1, 2, .., N.

Lời giải. Mỗi hoán vị $X = (x_1, x_2,..,x_K)$ là bộ có tính đến thứ tự của 1, 2, .., N. Mỗi $x_i \in X$ có N lựa chọn. Khi $x_i = j$ được lựa chọn thì giá trị này sẽ không được chấp thuận cho các thành phần còn lại. Để ghi nhận điều này, ta sử dụng mảng chuaxet[] gồm N phần tử. Nếu chuaxet[i] = True điều đó có nghĩa giá trị i được chấp thuận và chuaxet[i] = False tương ứng với giá trị i không được phép sử dụng. Thuật toán được mô tả như sau:

```
Void Try ( int i ) {
        for (int j =1; j<=N; j++){
                if (chuaxet[j] ) {
                        X[i] = j;chuaxet[j] = False;
                        if ( i ==N) Result();
                        else Try (i+1);
                        Chuaxet[j] = True;
                }
        }
}
```

Khi đó, việc duyệt các hoán vị của 1, 2, .., N ta chỉ cần gọi đến thủ tục Try(1). Cây quay lui được mô tả như hình dưới đây.

Cây đệ qui duyệt các hoán vị của 1, 2, 3.

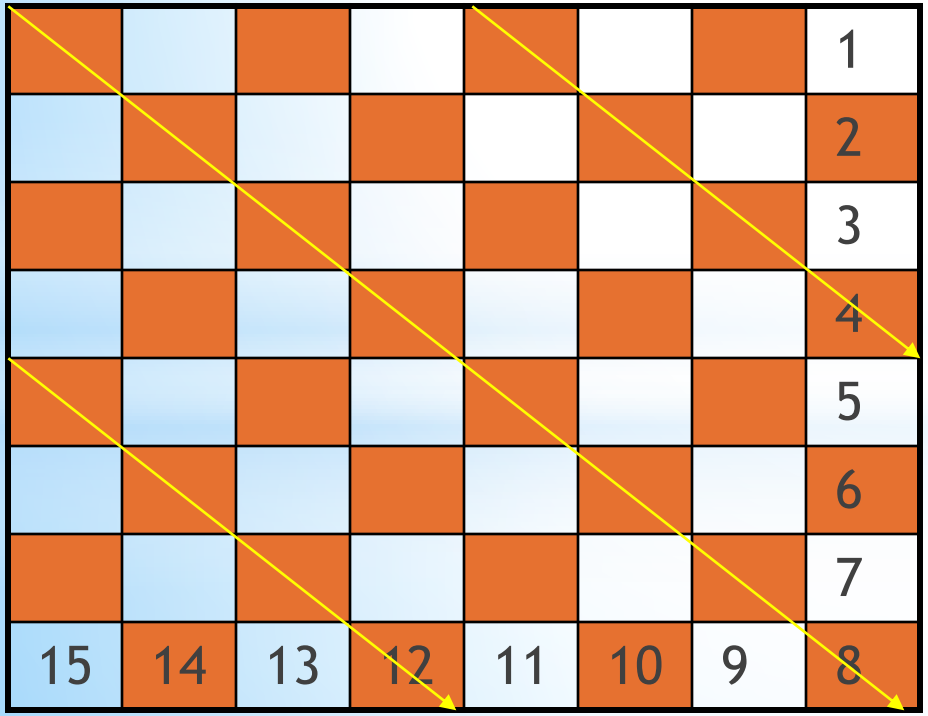


Bài 14. **Bài toán N quân hậu**. Trên bàn cờ kích cỡ N×N, hãy đặt N quân hậu mỗi quân trên 1 hàng sao cho tất cả các quân hậu đều không ăn được lẫn nhau.

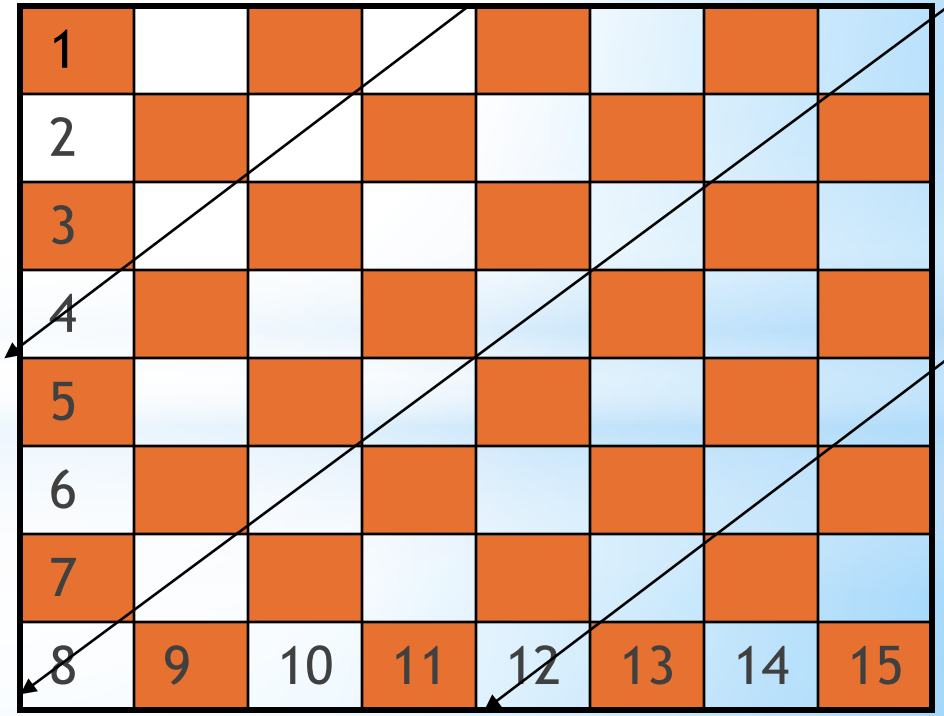
Lời giải. Gọi X =(x_1, x_2,..,x_n) là một nghiệm của bài toán. Khi đó, x_i = j được hiểu là quân hậu hàng thứ i đặt ở cột j. Để các quân hậu khác không thể ăn được, quân hậu thứ i cần không được lấy trùng với bất kỳ cột nào, không được cùng đường chéo xuôi, không được cùng trên đường chéo ngược. Ta có n cột A = (a_1,..a_n), có Xuoi[2*n-1] đường chéo xuôi, Nguoc[2*n-1] đường chéo ngược.

Nếu x_i =j thì A[j] = True, Xuoi[i-j+n] = True, Nguoc[i + j -1] = True.

Đường chéo xuôi: Xuoi [i – j + n]



Đường chéo ngược: Nguoc [i + j -1]



Thuật toán:

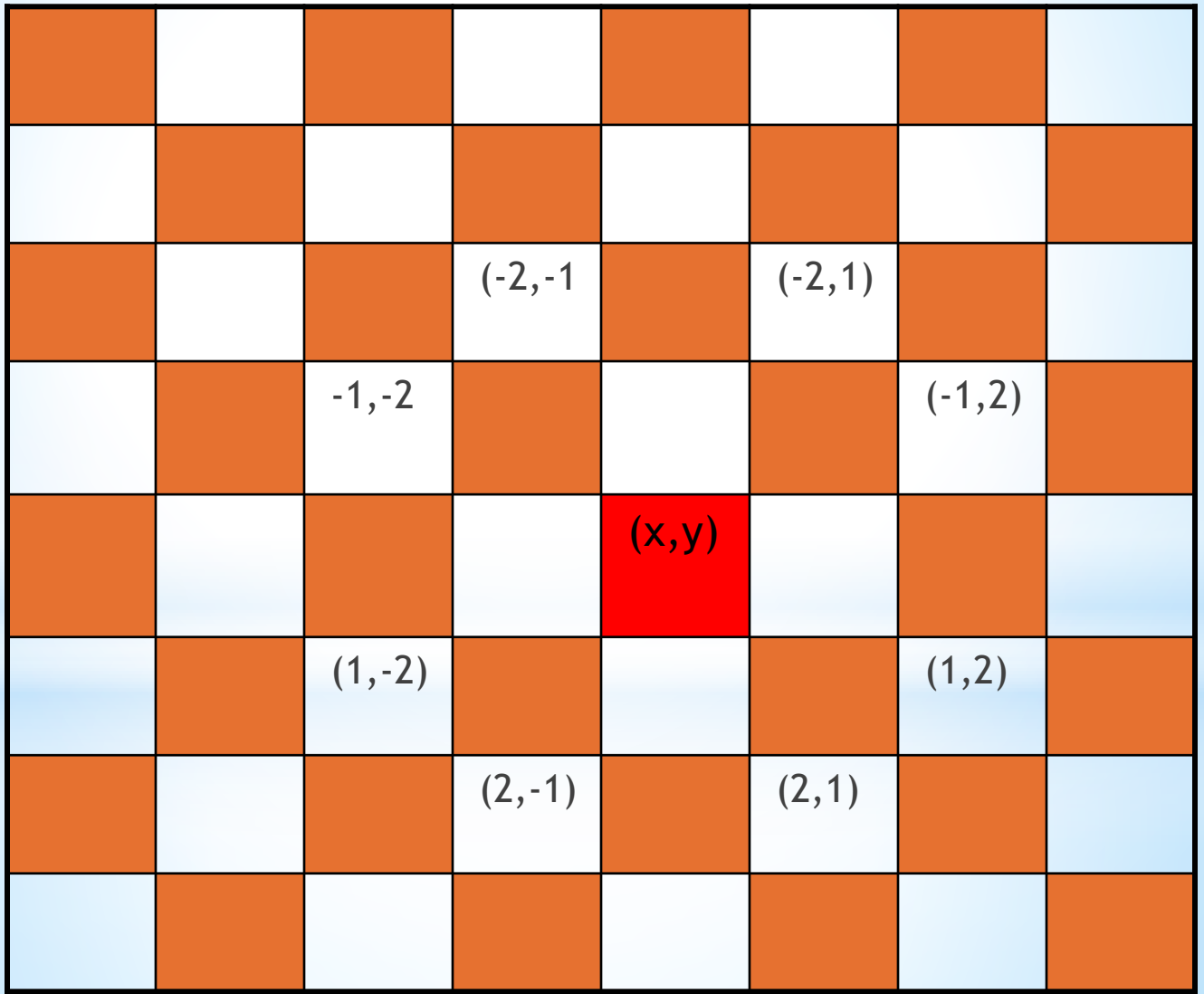
```
void Try (int    i){
            for(int j=1; j<=n; j++){
                        if( A[j] && Xuoi[ i – j + n ] && Nguoc[i + j -1]){
                                    X[i] =j;  A[j]=FALSE;
                                    Xuoi[ i - j + n]=FALSE;
                                    Nguoc[ i + j - 1]=FALSE;
                                    if(i==n) Result();
                                    else Try(i+1);
                                    A[j] = TRUE;
                                    Xuoi[ i - j + n] = TRUE;
                                    Nguoc[ i + j - 1] = TRUE;

                        }
            }
}
```

Bài XX.Mã đi tuần. Trên bàn cờ kích cỡ N×N, hãy tìm một hành trình của quân mã đi qua tất cả các ô trên bàn cờ mỗi ô đúng một lần.

Gọi dx[8] = {2, 1, -1, -2, -2, -1, 1, 2} là vector dịch chuyển theo hang

Gọi dy[8] = {1, 2, 2, 1, -1, -2, -2, 1} là vector dịch chuyển theo cột



Thuật toán:

```
void Try (int    i){
            for(int j=1; j<=n; j++){
                        if( A[j] && Xuoi[ i – j + n ] && Nguoc[i + j -1]){
                                    X[i] =j;  A[j]=FALSE;
                                    Xuoi[ i - j + n]=FALSE;
                                    Nguoc[ i + j - 1]=FALSE;
                                    if(i==n) Result();
                                    else Try(i+1);
                                    A[j] = TRUE;
                                    Xuoi[ i - j + n] = TRUE;
                                    Nguoc[ i + j - 1] = TRUE;

                        }
            }
}
```

BAI 1. TAM GIAC TONG

Cho dãy số A[] gồm n số nguyên dương. Tam giác đặc biệt của dãy số A[] là tam giác được tạo ra bởi n hàng, trong đó hàng thứ n là dãy số A[], hàng i là tổng hai phần tử liên tiếp của hàng i+1 ($1 \le i \le n-1$). Ví dụ A[] = {1, 2, 3, 4, 5}, khi đó tam giác được tạo nên như dưới đây:

[48]
[20, 28]
[8, 12, 16]
[3, 5, 7, 9]
[1, 2, 3, 4, 5]

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào N là số lượng phần tử của dãy số A[]; dòng tiếp theo đưa vào N số của mảng A[].
- T, N, A[i] thỏa mãn ràng buộc: $1 \le T \le 100$; $1 \le N, A[i] \le 10$;

Output: Đưa ra tam giác tổng của mỗi test theo từng dòng. Mỗi dòng của tam giác tổng được bao bởi ký tự [,].

Input	Output
1 5 1 2 3 4 5	[48] [20 28] [8 12 16] [3 5 7 9] [1 2 3 4 5]

BAI 2. HOAN VỊ XAU KY TỰ

Cho xâu ký tự S bao gôm các ký tự in hoa khác nhau. Hãy đưa ra tất cả các hoán vị của xâu ký tự S. Ví dụ S="ABC" ta có kết quả {ABC ACB BAC BCA CAB CBA }.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test là một xâu ký tự S được viết trên 1 dòng.
- T, S thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq length(S) \leq 10$;

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

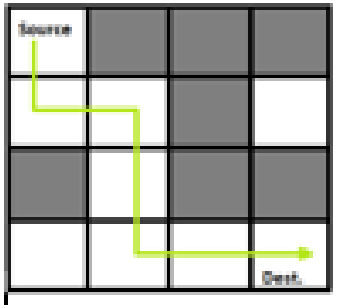
Input	Output
2 AB ABC	AB BA ABC ACB BAC BCA CAB CBA

BÀI 3. DI CHUYỂN TRONG MÊ CUNG 1

Cho một mê cung bao gồm các khối được biểu diễn như một ma trận nhị phân A[N][N]. Một con chuột đi từ ô đầu tiên góc trái (A[0][0]) đến ô cuối cùng góc phải (A[N-1][N-1]) theo nguyên tắc:

- Down (D): Chuột được phép xuống dưới nếu ô dưới nó có giá trị 1.
- Right (R): Chuột được phép sang phải dưới nếu ô bên phải nó có giá trị 1.

Hãy đưa ra một hành trình của con chuột trên mê cung. Đưa ra -1 nếu chuột không thể đi đến đích.



Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là kích cỡ của mê cung; dòng tiếp theo đưa vào ma trận nhị phân A[N][N]. T, N, A[i][j] thỏa mãn ràng buộc: $1 \leq T \leq 10$; $2 \leq N \leq 10$; $0 \leq A[i][j] \leq 1$.

Output: Đưa ra tất cả đường đi của con chuột trong mê cung theo thứ tự từ điển. Đưa ra -1 nếu chuột không đi được đến đích.

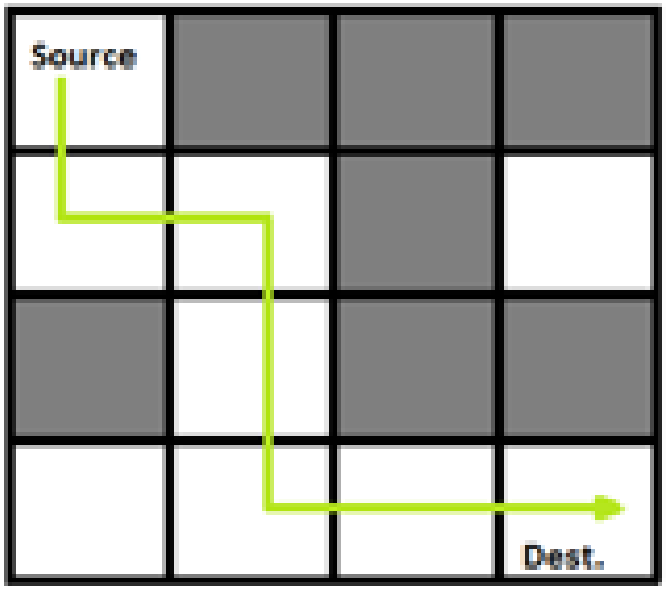
Input	Output
1 4 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	DRDDRR

BÀI 4. DI CHUYỂN TRONG MÊ CUNG 2

Cho một mê cung bao gồm các khối được biểu diễn như một ma trận nhị phân A[N][N]. Một con chuột đi từ ô đầu tiên góc trái (A[0][0]) đến ô cuối cùng góc phải (A[N-1][N-1]) theo nguyên tắc:

- Down (D): Chuột được phép xuống dưới nếu ô dưới nó có giá trị 1.
- Right (R): Chuột được phép sang phải dưới nếu ô bên phải nó có giá trị 1.
- Left (L): Chuột được phép sang trái dưới nếu ô bên trái nó có giá trị 1.
- Up (U): Chuột được phép lên trên nếu ô trên nó có giá trị 1.

Hãy đưa ra tất cả các hành trình của con chuột trên mê cung. Đưa ra -1 nếu chuột không thể đi đến đích.



BÀI 7. ĐỔI CHỖ CÁC KÝ TỰ

Cho số tự nhiên K và xâu ký tự các chữ số S. Nhiệm vụ của bạn là đưa ra số lớn nhất bằng cách thực hiện nhiều nhất K lần đổi chỗ các ký tự trong S. Ví dụ K =3 và S = "1234567" ta được "7654321".

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là số K; dòng tiếp theo là xâu ký tự S.
- T, K, S thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq K \leq 10$; $1 \leq .lenght(S) \leq 7$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
3 4 1234567 3 3435335 2 1034	7654321 5543333 4301

BÀI 8. CHIA MẢNG

Cho mảng các số nguyên A[] gồm N phần tử. Hãy chia mảng số nguyên A[] thành K tập con khác rỗng sao cho tổng các phần tử của mỗi tập con đều bằng nhau. Mỗi phần tử thuộc tập con xuất hiện duy nhất một lần trong tất cả các tập con. Ví dụ với A[] = {2, 1, 4, 5, 6}, K =3 ta có kết quả {2, 4}, {1, 5}, {6}.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số N và K; dòng tiếp theo đưa vào N số của mmảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, K, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, K \leq 20$, $0 \leq A[i] \leq 100$.

Output:

- Đưa ra 1 nếu có thể chia tập con thành K tập thỏa mãn yêu cầu bài toán, ngược lại đưa ra 0.

Input	Output
2 5 3 2 1 4 5 6 5 3 2 1 5 5 6	1 0

BÀI 9. TỔ HỢP SỐ CÓ TỔNG BẰNG X

Cho mảng A[] gồm N số nguyên dương phân biệt và số X. Nhiệm vụ của bạn là tìm phép tổ hợp các số trong mảng A[] có tổng bằng X. Các số trong mảng A[] có thể được sử dụng nhiều lần. Mỗi tổ hợp các số của mảng A[] được in ra theo thứ tự không giảm các số. Ví dụ với A[] = {2, 4, 6, 8}, X = 8 ta có các tổ hợp các số như sau:

[2, 2, 2, 2], [2, 2, 4], [2, 6], [4, 4], [8].

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số N và X; dòng tiếp theo đưa vào N số của mmảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, X, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq X, A[i] \leq 100$. N <= 20.

Output:

- Đưa ra kết quả mỗi test theo từng dòng. Mỗi đường tổ hợp được bao bởi cặp ký tự [,]. Đưa ra -1 nếu không có tổ hợp nào thỏa mãn yêu cầu bài toán.

Input	Output
1 4 8 2 4 6 8	[2 2 2 2] [2 2 4] [2 6] [4 4] [8]

BÀI 10. DI CHUYỂN TRONG MA TRẬN

Cho ma trận A[M][N]. Nhiệm vụ của bạn là đưa ra tất cả các đường đi từ phần tử A[0][0] đến phần tử A[M-1][N-1]. Bạn chỉ được phép dịch chuyển xuống dưới hoặc sang phải phần tử liền kề với vị trí hiện tại.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số M, N tương ứng với số hàng và số cột của ma trận; dòng tiếp theo đưa vào các phần tử của ma trận A[][]; các số được viết cách nhau một vài khoảng trống.
- T, M, N, A[i][j] thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq M, N, A[i][j] \leq 100$.

Output:

- Đưa ra số cách di chuyển của mỗi test theo từng dòng.
- Giải thích test 1: Có 3 cách di chuyển là [1 4 5 6], [1 2 5 6] và [1 2 3 6].

Input	Output
2 2 3 1 2 3 4 5 6 2 2 1 2 3 4	3 2

BAI 12. TỪ ĐIỂN

Cho tập từ ghi trong trừ điển dic[] và một bảng hai chiều A[M][N] các ký tự. Hãy tạo nên tất cả các từ có mặt trong từ điển dic[] bằng cách nối các ký tự kề nhau trong mảng A[][]. Chú ý, phép nối các ký tự kề nhau trong mảng A[][] được thực hiện theo 8 hướng nhưng không có phần tử A[i][j] nào được lặp lại. Ví dụ với từ điển dic[] ={ "GEEKS", "FOR", "QIUZ", "GO"} và mảng A[][] dưới đây sẽ cho ta kết quả: "GEEKS", "QUIZ"

G	I	Z
U	E	K
Q	S	E

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào ba số K, M, N tương ứng với số từ của từ điển dic[], số hàng và số cột của ma trận ký tự A[M][N]; dòng tiếp theo đưa vào K từ của từ điển dic[]; dòng cuối cùng đưa vào các phần tử A[i][j].
- T, K, M, N thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq K \leq 100$; $1 \leq M, N \leq 3$.

Output:

- Đưa ra theo thứ tự tăng dần các từ có mặt trong từ điển dic[] được tạo ra từ ma trận A[][]. Đưa ra -1 nếu không thể tạo ra từ nào thuộc dic[] từ A[][].