# 006. VÒNG SỐ NGUYÊN TỐ

Một vòng tròn chứa 2n vòng tròn nhỏ (Xem hình vẽ). Các vòng tròn nhỏ được đánh số từ 1 đến n theo chiều kim đồng hồ. Cần điền các số tự nhiên từ 1 đến 2n mỗi số vào một vòng tròn nhỏ sao cho tổng của hai số trên hai vòng tròn nhỏ liên tiếp là số nguyên tố. Số điền ở vòng tròn nhỏ 1 luôn là số 1.

****

**Dữ liệu:** Vào từ file văn bản CIRCLE.INP chứa số nguyên dương n (1 < n < 10)

**Kết quả:** Ghi ra file văn bản CIRCLE.OUT:

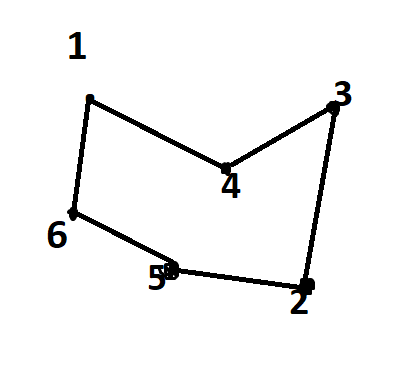
* Dòng đầu tiên ghi số lượng các cách điền số tìm được (k).
* Dòng thứ i trong số k dòng tiếp theo ghi các số trong các vòng tròn nhỏ bắt đầu từ vòng tròn nhỏ 1 đọc theo thứ tự của các vòng tròn nhỏ

**Ví dụ:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CIRCLE.INP | CIRCLE.OUT |  | CIRCLE.INP | CIRCLE.OUT |
| 3 | 2  1 4 3 2 5 6  1 6 5 2 3 4 |  | 4 | 4  1 2 3 8 5 6 7 4  1 2 5 8 3 4 7 6  1 4 7 6 5 8 3 2  1 6 7 4 3 8 5 2 |

**Ý tưởng** : Từ hình vẽ minh hoạ ta nhận thấy vòng tròn 1 có thể đi đến các vòng tròn có giá trị 4,6 Tương tự vòng tròn có giá trị 4 chỉ có thể đi đến vòng tròn giá trị 1 hoặc 3,.. Ta có thể xem như là đường đi giữa các đỉnh đồ thị.

Từ phân tích trên ta được đồ thị:



Ví dụ n = 3 , với 2n vòng tròn nhỏ ( đề bài ) thì ta được 6 đỉnh và bắt đầu từ đỉnh 1. Dễ dàng nhận thấy đồ thị này là đồ thị có hướng nên ta sẽ có được 2 chu trình :

Chu trình 1 : 1 4 3 2 5 6 1

Chu trình 2 : 1 6 5 2 3 4 1

* Tổng của 2 đỉnh kề nhau sẽ là 1 số nguyên tố.

Ví dụ đỉnh 1 và đỉnh 4 thì tổng của chúng bằng 5 là 1 số nguyên tố.

Dựa vào ý tưởng này ta xây dựng một ma trận kề có 2n \* 2n phần tử. Bài này ta sẽ lấy ví dụ n = 3

Bước 1 : Ta xây dựng một hàm kiểm tra số nguyên tố.

Code :

1. bool isPrime(int n) // Hàm kiểm tra các số nguyên tố.
2. {
4. if (n <= 1)
5. **return false;**
6. for (int i = 2; i < sqrt(n) ; i++)
7. if (n % i == 0)
8. return false;
10. **return true;**
11. }

Bước 2 : Khởi tạo ma trận 2n \* 2n với các giá trị mặc định là 0

Code :

void init(int n) // khởi tạo giá trị mặc định của chu trình

{

    for (int i = 1; i <= 2 \* n; i++)

    {

        for (int j = 1; j <= 2 \* n; j++)

        {

            A[i][j] = 0;

        }

    }

}

Kết quả sẽ được một ma trận sau đây.

**0 0 0 0 0 0**

**0 0 0 0 0 0**

**0 0 0 0 0 0**

**0 0 0 0 0 0**

**0 0 0 0 0 0**

**0 0 0 0 0 0**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| j = 1  i = 1 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 |

Bước 3 : Xây dựng ma trận kề với điều kiện i + j sẽ là số nguyên tố và |i - j| cũng là 1 số nguyên tố.  
Code :

void matran(int n) // khởi tạo ma trận kề

{

    for (int i = 1; i <= 2 \* n; i++)

    {

        for (int j = 1; j <= 2 \* n; j++)

        {

            int sum = i + j;

            int diff = abs(i - j);

            if (isPrime(sum))

            {  cout<<"i = "<<i <<"j = "<<j<<endl;

                A[i][j] = 1;

            }

        }

    }

}

Ví dụ i = 1 , j = 4 thì (i + j) = 5 sẽ là 1 số nguyên tố thì sẽ thay đổi giá trị của ma trận tại vị trí (i,j) từ 0 thành 1;

Tương tự ta có

i = 1 ,j = 4

i = 1, j = 6

i = 2, j = 5

i = 4, j = 1

i = 5, j = 2

i = 6, j = 1

Ma trận sau khi đã được thay đổi giá trị :

**0 0 0 1 0 1**

**0 0 0 0 1 0**

**0 0 0 0 0 0**

**1 0 0 0 0 0**

**0 1 0 0 0 0**

**1 0 0 0 0 0**

Bước 4 : Từ ma trận kề với 2n đỉnh duyệt từ đỉnh 1 ta áp dụng chu trình Hamilton :

Mã giả :

void Hamilton( int k) {

*/\* Liệt kê các chu trình Hamilton của đồ thị bằng cách phát triển dãy đỉnh*

*(X[1], X[2],..., X[k-1] ) của đồ thị G = (V, E) \*/*

for y∈Ke(X[k-1]) {

if (k==n+1) and (y == v0) then

Ghinhan(X[1], X[2],..., X[n], v0);

else {

X[k]=y; chuaxet[y] = false;

Hamilton(k+1);

chuaxet[y] = true;

}

}

}

Code :

#define MAX 100

int C[MAX], B[MAX];

int d; // số lượng chu trình

int sodinh ; // số đỉnh = 2n;

void Result(void) // in chu trình

{

    for (int i = sodinh; i > 0; i--)

        cout << B[i] << " ";

    d++; // tăng biến đếm các chu trình có thể tìm được

    cout << endl;

}

void Hamilton(int \*B, int \*C, int i) // duyệt chu trình Hamilton với tham số đầu vào là Ma trận kề A

{

    int j, k;

    for (j = 1; j <= sodinh; j++)

    {

        if (A[B[i - 1]][j] == 1 && C[j] == 0)

        {

            B[i] = j;

            C[j] = 1;

            if (i < sodinh)

                Hamilton(B, C, i + 1);

            else if (B[i] == B[0])

                Result();

            C[j] = 0;

        }

    }

}

Kết quả trả về sẽ là các chu trình xuất phát từ đỉnh có thể đi được trong đồ thị :

