Dùng mảng Color[] để đánh dấu màu các đỉnh trong quá trình DFS.

Ban đầu với mọi v thì Color[v]=White. Khi một đỉnh u được thăm thì đánh dấu Color[u]=Green, nếu kết thúc mà không có chu trình thì đánh dấu Color[u]=Black, còn trong quá trình xét nếu gặp một đỉnh mà x nào đó mà Color[x]=Green chứng tỏ có 1 chu trình, lúc này kết thúc chương trình và truy vết theo đường đi này.

```
void DFS(int u)
{
 //  color[u] = gray; thay gray =1, white=0, black=-1
     color[u] = 1;
   for (auto Ke: adj[u])
    {
     if (Stop) return;   //Nếu đã tìm được chu trình thì dừng
        int v=Ke.x;
        int cs=Ke.y;
        //Nếu gặp đỉnh v chưa thăm thì đi thăm đỉnh v
        if (color[v] == 0)
         {
              Trace[v]= cs; // Lưu để truy vết
               DFS(v);
        else
          if (color[v] == 1) //Nếu đỉnh v đã thăm trong dây chuyền DFS thì xác định 1 chu trình
          {
             trace[v]=cs;
              LuuV = v;       // Lưu lại đỉnh v để truy vết, đây là đỉnh bắt đầu và kết thúc của chu trình
              Stop=true;              //Đánh dấu có chu trình;
              Return;
          }

 }
   color[u] = -1;  //Đã thăm xong đỉnh u nhưng không có chu trình
}
```

Chương trình chính:

Phần đọc dữ liệu xem ở phía dưới

```
for (∀u ∈ V)
   color[u] = white;
for (∀s ∈ V)
  {
     if (color[s] == 0)       DFS(s);
      if(Stop)     break;
}
//In kết quả và truy vết
if(Stop == false) cout<<"NO";
 else
   {
     cout<<"YES"<<'\n';
```

Phần truy vết xem ở phía dưới

```
}
```

Bài toán yêu cầu in số hiệu các cung theo đúng thứ tự trên chu trình tìm được. Chính vì vậy phải lưu lại được chỉ số của cạnh khi đọc dữ liệu.

Mảng vector **adj[u]** ngoài việc lưu đỉnh kề của u thì cần lưu trữ thêm chỉ số cạnh của cung (u,v)

```
#define x first              // Lưu đỉnh kề
#define y second        //Lưu chỉ số cạnh
typedef pair<long long, long long> Phantu;
vector< Phantu > adj[100001];
pair<long long, long long>  canh[100001];  //Mảng Canh[i] lưu cặp cạnh kề (u,v) thứ i
//-------Đọc dữ liệu----------
            cin>>n>>m;
            for(long long i=1;i<=m;i++)
            {
              cin>>u>>v;
              adj[u].push_back({v,i});   //Đỉnh kề v và chỉ số cạnh là i
             canh[i]={u,v};
            }
```

Truy vết:

Quá trình truy vết: t=LuuV ->trace[t]=i, canh[i] ={u,t} xác định được u, đặt t=u… tiếp tục lặp cho tới khi t==LuuV

```
vector<int > ans;   //Khai báo Vector ans để lưu kết quả là chỉ số các cạnh.
cs= trace[LuuV];      //Lấy chỉ số cạnh đầu tiên
ans.push(cs);   //Đẩy chỉ số cạnh cs vào Vector kết quả
t=canh[cs].x;
while (t!=LuuV)
            {
              cs= trace[t];
              ans.push(cs);   //Tiếp tục đẩy chỉ số cạnh cs vào Vector kết quả
              t=canh[cs].x;
            }
reverse(ans.begin(),ans.end());  //Lật ngược Vector kết quả trước khi in
for(auto cs_canh: ans){
      cout<<cs_canh<<" ";
   }
```